Dynamic, Secure Resource Control in the Cloud

Viswanath Nandina, Edward J. Nava, José Marcio Luna, Christopher C. Lamb,
Gregory L. Heileman, Chaouki T. Abdallah
University of New Mexico
Department of Electrical and Computer Engineering
Albuquerque, NM 87131-0001
{vishu, ejnava, marcio, cclamb, heileman, chaouki}@ece.unm.edu

ABSTRACT

In this paper we describe the development of a system that provides security and performance controls over content in a cloud environment. Using artifacts that are classed at different sensitivity levels associated with service level agreements (SLAs) describing how and where they can be used, we are able to successfully provision resources in a hybrid cloud environment. These provisioned resources are created to match both performance characteristics as well as specific sensitivity restrictions specified within associated SLAs.

Categories and Subject Descriptors

D.2.11 [**Software**]: Software Architectures—*Domain-specific Architectures*

General Terms

Design, Performance, Security

Keywords

Access Control, Interoperability, DRM, Usage Management

1. INTRODUCTION

With the advent and widespread use of cloud computing, those responsible for a given usage managed resource are almost never those responsible for the computing systems, except at edge devices like mobile phones or other small profile computing devices. Resources are regularly moved across national boundaries and regional areas without either the content owner's or creator's knowledge. Furthermore, this kind of transfer is generally according to pre-established algorithms or data routing protocols over which users have no control. Managing these issues requires new usage management capabilities that can run on platforms ranging from small, hand-held devices to nodes in large data centers.

Herein, we define usage management as the ability to control actions over resources and data across and within computing environments. More than access control or digital rights management, usage management addresses with fine-grained control of all aspects of how a given digital resource is used. As digital environments become more open over time, the need for usage management for resources that span utility computational environments (e.g. cloud provider systems) will become increasingly important [17, 18].

One of the central problems with respect to usage management application in cloud environments is the ability to specify and enforce specific levels of usage management over resources. The essential mobility of information within private, public, and hybrid cloud environments makes this management virtually impossible currently, leaving system developers left with the sole option of designing proprietary information management capabilities. These capabilities become more and more expensive to develop as computational environments become increasingly distributed, and are generally a nightmare to manage, both due to the proliferation of disparate technology throughout the available public and private cloudosphere.

Furthermore, cloud computing is emerging as the future of utility systems hosting for consumer-facing applications. In these kinds of systems, components, applications, and hardware are provided as utilities over the Internet with associated pricing schemes pegged by system demand. Users accept specific Quality-of-Service (QoS) guidelines that providers use to provision and eventually allocate resources. These guidelines become the basis over which providers charge for services.

This work focuses on developing integrated SLAs that address both traditional performance measures as well as security and usage management directives. We feel that cloud monitoring capabilities have over the past year reached the point at which commercial providers supply enough performance monitoring and system management primitives to begin to implement cloud-scale automatic management systems addressing data processing suitability from both performance and security perspectives. For example, Amazon now supplies information through their CloudWatch product and provides a robust control interface via their Elastic Compute Cloud (EC2). Likewise, private cloud infrastructure-as-a-service (Iaas) offerings like OpenStack and Eucalyptus provide similar control interfaces, simplifying control of hybrid systems.

In this paper we first develop the mechanisms to enable

CCSW' 12, October 19, 2012, Raleigh, NC, USA.

control based on SLA established performance and security measures. Then, we introduce a prototypical system architecture based on widely used open-source tooling to establish a simple proof of concept. Finally we discuss preliminary observations from running our prototypical system in a hybrid cloud enfvironment.

2. CONTRIBUTING WORK

Current policy-centric systems are being forced to move to cloud environments and incorporate much more open systems. Driven by both cost savings and efficiency requirements, this migration will result in a loss of control of computing resources by involved organizations as they attempt to exploit economies of scale and utility computing.

Robust usage management will become an even more important issue in these environments. Federal organizations poised to benefit from this migration include agencies like the National Security Agency (NSA) and the Department of Defense (DoD), both of whom have large installed bases of compartmentalized and classified data. The DoD realizes the scope of this effort, understanding that such technical change must incorporate effectively sharing needed data with other federal agencies, foreign governments, and international organizations [2]. Likewise, the NSA is focused on using cloud-centric systems to facilitate information dissemination and sharing [4].

Cloud systems certainly exhibit economic incentives for use, providing cost savings and flexibility, but they also have distinct disadvantages as well. Specifically, the are not intrinsically as private as some current systems, generally can be less secure than department-level solutions, and have extensive trust and privacy issues [22].

How to address these issues is an open research question. Organizations ranging from cloud service providers to governments are exploring how to engineer solutions to these problems, and to more clearly understand the trade-offs required between selected system architectures [3]. The problems themselves are wide ranging, appearing in a variety of different systems. Healtcare and government systems are clearly impacted by these kinds of trust and security issues, and they also have clear information sensitivity problems. This, coupled with the fact that these organizations have been dealing with these issues in one form or another for decades make them very well suited for prototypical implementation and study.

Over the past few years multiple service-based paradigms such as web services, cluster computing and grid computing have contributed to the development of what we now call cloud computing [9]. Cloud computing distinctly differentiates itself from other service-based computing paradigms via a collective set of distinguishing characteristics: market orientation, virtualization, dynamic provisioning of resources, and service composition via multiple service providers [10]. This implies that in cloud computing, a cloud-service consumer's data and applications reside inside that cloud provider's infrastructure for a finite amount of time. Partitions of this data can in fact be handled by multiple cloud services, and these partitions may be stored, processed and routed through geographically distributed cloud infrastructures. These activities occur within a cloud, giving the cloud consumer an impression of a single virtual system. These operational characteristics of cloud computing can raise concerns regarding the manner in which cloud consumer's data and applications are managed within a given cloud. Unlike other computing paradigms with a specific computing task focus, cloud systems enable cloud consumers to host entire applications on the cloud (i.e. software as a service) or to compose services from different providers to build a single system. As consumers aggressively start exploiting these advantages to transition IT services to external utility computing systems, the manner in which data and applications are handled within those systems by various cloud services will become a matter of serious concern [14].

A growing body of research has begun to appear over the past two years applying control theory to tuning computer systems. These range from controlling network infrastructure [7] to controlling virtualized infrastructure and specific computer systems [23], [15] to exploring feedforward solutions based on predictive modeling [5]. Significant open questions remain within this field [25], [13].

To address these issues, we first applied the principles of system design to develop a framework for usage management in open, distributed environments that supports interoperability. These principles have been used by researchers in large network design to create a balance between interoperability and open, flexible architectures [6, 8, 11], without sacrificing innovation. Initially we standardized certain features of the framework operational semantics, and left free of standards features that necessitate choice and innovation.

Usage management incorporates specific characteristics of traditional access control and digital rights management incorporating encryption mechanisms, trust management, and trusted computing platforms [14]. In order to be effective, it must be flexible enough to provide users with opportunities for differentiation and extension, but interoperable enough to provide services across widely diverging computational environments.

3. USAGE MANAGEMENT & ACCESS CONTROL

Usage management is the concept of managing resources within computational ecosystems. These resources are controlled by images. Each image has policies associated with it that describe the circumstances in which the image can be used. Likewise, each user has an identity associated with them that describe their credentials as well as environment information, also known as context. Usage management occurs when a user's identity information is compared with a resources' policies to determine if that user is qualified to use that particular resource.

3.1 Access Control

Access control is utility to manage controlled access resources. The various components of our access control model are given below:

Roles: The roles table is used to record user's roles. There is an entry for each user. It consists of an id, which is a unique identifier that defines different type of roles, and another field called parent, which can be used to define inheritance. Resources: The resource table is used to identify each resource whose access is being controlled. Each entry consists of an id, which is a unique identifier, and another field for the resource descriptor. A resource can be anything from a file to a service. RolesResources: The RolesResouces is the access control mapping between the roles and resources. It consists of 3 fields. The first entry in the column identifies the user by Roleid. The second entry contains the id of a resource to which a user is authorized access. The third field defines the user's privileges for that resource.

3.2 Cloud Computing

There are different models for cloud computing ranging from software as a service (SaaS) to infrastructure as a service (IaaS). For our implementation we are only concerned with IaaS. Cloud computing systems include both public and private systems.

3.3 System Specifications

3.3.1 System Overview

Our primary goal is to develop and implement a secure, robust, and inter-operable role-based usage management system for data transactions in cloud computing. This system will merge traditional usage management systems with modern cloud computing technologies [14].

The usage management system envisioned consists of an ACL containing a usage hierarchy where every person must adopt one or more of two explicit roles: admin or user. An admin sits atop the hierarchy and therefore has the most privileged access to resources. A user, generally, has limited access only to those resources approved by the usage management policy for that user. Similar to the hierarchal structure of an object-oriented programming language, every admin is a user, but every user is not an admin. Based on an individual's role they are assigned an identification number specific to the role. If someone requires multiple roles then unique identifications must be created for each of those roles. This system access policy is called Role-Based Access Control (RBAC).

Every user and admin has access to certain resources defined by the user and admin's context of access control. To formalize the confidentiality, integrity, and authenticity of data resources we have defined the context of access control by the tuple S = {C, I, A}, where C is a bit indicating the confidentiality of the data resource, I is a bit indicating the integrity of the data resource, and A is a bit indicating the authenticity of the data resource. That is, each bit defines the high or low status of confidentiality, integrity, and authenticity and is syntactically represented by the letters H or L. Hence, there are 8 possible list orderings of access control. These orderings indicate the degree of access admin's and user's have within the system. Therefore, if a user wishes to access a resource with some or all of the 8 possible degrees of security, the user must have the appropriate authoritative credentials and clearance rights. These usage policy management rights are determined by the private party and then strictly controlled in the system's infrastructure to ensure the party complete security and control over whom is able to view specific data resources.

Each user's access to any data resources is controlled through a database. The database is designed to control access by assigning a unique identification number to each role, that is, a number corresponding to each role, not each user. Based on the credentials of the user, the user is presented a list of resources the user is allowed to access and is informed there exists a private encryption key for each resource. If the user sufficiently meets the qualifications for viewing the data resource, our custom Cloud Manager software provisions the data based on the degree of security particular to the data. The Cloud Manager instantiates the Virtual Machine using the corresponding private key to the specific Virtual Machine. Since there are 8 degrees of security as defined by the ordered tuple attributes, our software selects one of 8 image instances and allows the data to be accessed through one of 8 Virtual Machines. Each image instance and Virtual Machine only handles data with the appropriate degree of security it is specified to manage. When a user decides to launch an image, the Cloud Manager will query a resource table for the three attributes of the image intended to facilitate that resource. For example, if all the three security attributes of the image are SC {H,H,H}, then the image can be executed in a Virtual Machine that also has all of these attributes set to SC {H,H,H}.

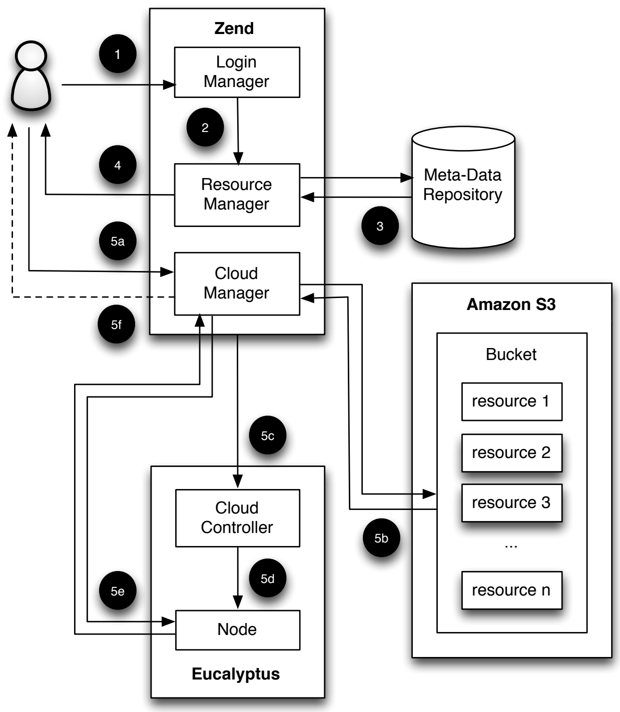
Resources are typically stored in a cloud computing repository such as Amazon Simple Storage service (Amazon S3). Amazon S3 can be used to store and retrieve data in a very flexible manner via the web. It provides redundant storage and the management is transparent to the user.

3.3.2 System Architecture/Model

1. User logs into Login Manager
2. Request & forward credentials to Resource Manager
3. List of available resources returned to the user
4. The Resource Manager retrieves user information from Meta-Data repository

5.a. User selects a resource
5.b. Cloud Manager retrieves resource from Amazon S3
5.c. Cloud Manager creates an appropriate Virtual Machine instance
5.d. Instance created
5.e. Control monitoring & processing
5.f. Returns results of processing

The operational details of our usage management system are visualized in Figure 1. By means of a web browser user interface, the user is presented with a login dialog box. The user enters user credentials to log into the system. The credentials are forwarded to the Resource Manager. The Resource Manager queries a Meta-Data repository to get information about the user. Once the user's identification has been authenticated, a list of available resources is presented on the user's browser. This list of resources varies depending on user context. If the user selects a resource, the Cloud Manager software retrieves the desired resource from a cloud computing repository like Amazon S3. The Cloud Manager software then instantiates an appropriate Virtual Machine specified to handle resources with the same security context. The Cloud Manager also monitors the instance and returns references about the performance of the node to the Cloud Manager. Using the approach explained in Section [?], we



1. User logs into Login Manager
2. Request & forward credentials to Resource Manager
3. List of available resources returned to the user
4. The Resource Manager retrieves user information from Meta-Data repository
5.a. User selects a resource
5.b. Cloud Manager retrieves resource from Amazon S3
5.c. Cloud Manager creates an appropriate Virtual Machine instance
5.d. Instance created
5.e. Control monitoring & processing
5.f. Returns results of processing

Figure 1: Component Diagram of Usage Management for a Cloud-Based System

regulate the performance measures of the system that are covered by SLA requirements.

3.3.3 Implementation

The implementation of the system is shown in Figure 2. The application is written using Zend framework and we use MySQL for the database services. When a user logs in, we use Zend to authenticate and determine the role of the user. Once a user is authenticated we use ACL to decide what resources the user can access. Based on the resources available to the user we present the user with a list of images the user can run on a Virtual Machine. It is important to note the actual resources are stored in Amazon S3 and some of the features of the Eucalyptus cloud system have been omitted from our diagram for the sake of brevity. If the user selects a resource, we query the resource table to get the appropriate keys and then execute the image in the virtual machine specified to manage resources with the appropriate security categories. With the help of our software we monitor the VMs for performance as defined in the SLA requirements and control the VM appropriately.

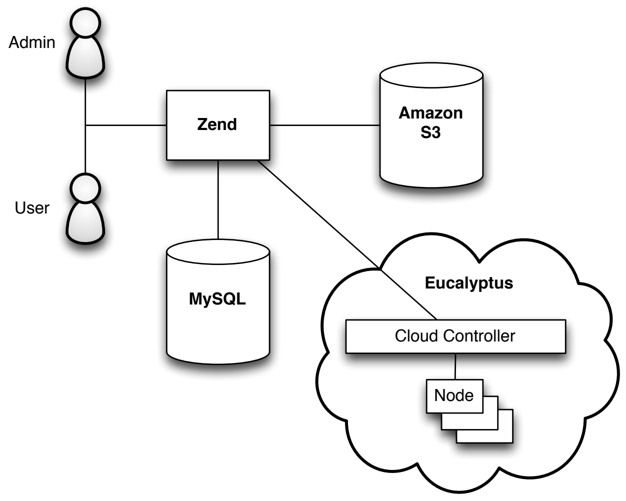


Figure 2: Technology Architecture

4. SECURITY CATEGORY ANALYSIS

4.1 Security Categories

Our plan is to label resources (information) and systems using the NIST standards for security classification of Federal Information and Information Systems [1].

The security objectives for both the information and the information systems are: confidentiality, integrity, and availability. For both the information and the information systems, the impact of the compromise of each of these objectives is categorized as: low impact, moderate impact, or high impact. A generalized format for expressing an information type security category as follows:

$$SC = \{confidentiality, integrity, availability\}$$

An overall system category is the high water mark value of the three individual objectives. For example, a system with moderate impact for confidentiality and low impact for both integrity and availability is given an overall rating as a moderate impact system.

A security policy-based control system will contain a table with information system security category information that will be used as a basis for allowing or not allowing a operation on an individual cloud computing system.

4.2 Protection Measures for Security Objectives

For each security category, we can use a number of mechanisms to achieve the necessary level of protection, integrity assurance, or availability. The mechanisms that may be used for each level are proposed below. For all transactions with data, we assume that a usage mechanism is available to control access as appropriate. Once the user has been granted access to the data, the mechanisms below would be incorporated into an operating system image that will be loaded on newly instantiated VMs.

Additional protection can be achieved by using private or community cloud computing systems. This approach would serve to eliminate the sharing of the base hardware with untrusted users, but also forces organizations to pay to install and maintain the systems. This approach has its merits but also negates some of the cost savings than an organization could get by renting what they need, only when they need it. As a result, we are focusing on measures that can be used in public cloud computing systems. The specific measures we propose are shown below.

Level	Measures Or Mechanisms
H	Data encrypted while at rest, Data transmitted in encrypted form (HTTPS, SSL, Tunnels)
M	Data transmitted in encrypted form (HTTPS, SSL, Tunnels)
L	Data handled and transmitted in the clear, Use standard OS file protection measures

Table 1: Confidentiality

Level	Measures Or Mechanisms
H	Data are signed with a secure hash [2], Data transmitted in encrypted form (HTTPS, SSL, Tunnels)
M	Data transmitted in encrypted form (HTTPS, SSL, Tunnels), Data are verified with a Message Authentication Code [3]
L	Data handled and transmitted in the clear, Use standard OS file protection measures, Standard Parity checks used

Table 2: Integrity

Level	Measures Or Mechanisms
H	Data stored using Simple Storage System (S3), (Use built-in redundancy)
M	Data stored using S3, (Use built-in redundancy)
L	Data are stored on local machines, Data stored using S3

Table 3: Availability

4.3 Implementation

VMs are instantiated to meet computational needs; the specific configuration of each can be controlled. With no security controls, a user selects one from a set of images that can be used to launch the VM. The image contains the operating system and other application software that is loaded into the VM when it is instantiated.

Images can be configured with mechanisms to implement the functions required for each security category. Images that process data with security triples of SC{H,X,X}, SC{M,X,X}, SC{X,H,X}, and SC{X,M,X} (X - denotes don't care) should be configured to include intrusion detection system software, firewall software, virus checking software, and operations logging. In addition, the operating systems should be configured to prevent the user from reconfiguring the operating system and application software. (As good computer security practice, all images should include virus checking software.)

No.	Conf.	Integ.	Avail.	Potential Examples
1	H	H	H	Medical Records, credit card transactions, financial management
2	H	H	L	Applications with non-urgent sensitive info, Social Security administration
3	H	L	H	Can disregard?
4	H	L	L	Can disregard?
5	L	H	H	Breaking News, Emergency health guidelines
6	L	H	L	General News, Official government documents
7	L	L	H	Weather warnings
8	L	L	L	General interest and Non-critical information

Table 4: Reduced Set of Security Categories

4.4 Implementation Considerations

The security category triple yields a set of 3^n possible combinations. With $n = 3$, this results in 27 different security categories to consider in a system design. As a first step in simplifying the problem to make it more tractable, we will reduce each security objective to two values, low or high. This reduces the set of possibilities to 2^n, or 8 total.

For demonstration purposes, we will use two VM images, one with SC {H,H,H} and the other with SC {L,L,L}. In later development spirals, we will add more VM images that are configured for the other combinations shown above and potentially combinations with intermediate security levels. As protection mechanisms impose computational burdens, there is motivation for using images with relaxed security requirements. Images configured for high consequence security categories can be used with systems with low consequences, but the converse is not true. So, there is motivation to have VM images that are tailored for the security requirements. By using the simplification shown above, configuring a set of images to handle each of the combinations is manageable.

The experimental system configuration is shown in the figure below.

5. SYSTEMS AND CONTROL APPROACH

In a cloud computing environment each application may have different performance measures, such as, response time, throughput or even number of instructions per second in the case of compute applications. Furthermore, since our main goal is to control more than one performance measure by controlling one or more inputs of the system, we consider the cloud model to be a Multiple-Input-Multiple-Output (MIMO) system.

The control method we use in our approach rely upon the availability of a mathematical model relating virtual resources and performance measures of the system. Based on [21], given an input vector $\mathbf{u}(n) = (u_1(n), \ldots, u_q(n))^T$ and a state vector $\mathbf{y}(n) = (y_1(n), \ldots, y_p(n))^T$ of the cloud, the general MIMO model of the cloud interactions is given by a set of nonlinear difference equations. The existent identification techniques for nonlinear systems are not suitable to be implemented in real-time, furthermore the amount of required learning data for the identification algorithm turns out to be impractical except for the simple cases.

In previous works on data centers and cloud computing such as, [21],[24] and [20] a linear time-invariant realization of the system is accurate enough as long as the control inputs of the system are constrained to a relatively "small" interval. In the following section we provide a detailed description of the system and a first candidate for the controller.

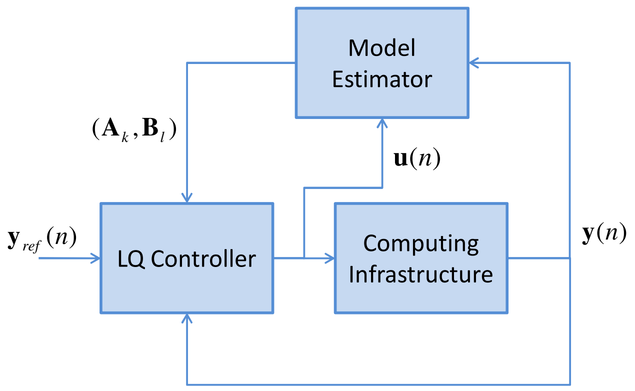


Figure 3: Component Diagram of Usage Management for a Cloud-Based System

5.1 Dynamic Model

Our model is approximated by the following difference equations,

$$\mathbf{y}(n) = \sum_{k=1}^{N} A_k \mathbf{y}(n-k) + \sum_{l=0}^{M-1} B_l \mathbf{u}(n-l) + \mathbf{w}(n), \qquad (1)$$

where $\mathbf{y} \in \Re^p$ is the state vector, $\mathbf{u} \in \Re^q$ is the input vector, $\mathbf{w} \in \Re^q$ is a zero-mean white-noise vector, and the system matrices $A_k \in \Re^{p\times p}$ and $B_l \in \Re^{p\times q}$ are to be estimated.

Now, by defining the matrix X and the vector ϕ as follows,

$$X = (B_0, B_1, \ldots, B_{M-1}, A_1, A_2, \ldots A_N), \qquad (2)$$

$$\phi(n) = (\mathbf{u}(n)^T, \ldots, \mathbf{u}(n-M+1)^T, \mathbf{y}(n-1)^T, \ldots, \mathbf{y}(n-N)^T)^T,$$

then we can reformulate the difference equation (1) as,

$$\mathbf{y}(n) = X\phi(n) + \mathbf{w}(n).$$

Therefore, by getting an estimate $\hat{X}$ of the matrix X we get a linear realization of the model based on the inputs and outputs of the system.

The following iterative equations, based on [16], conform our Recursive Least Square (RLS) estimator,

$$\begin{aligned}
\varepsilon(n+1) &= \mathbf{u}(n+1) - \hat{X}(n)\phi(n), \\
\hat{X}(n+1) &= \hat{X}(n) + \frac{\varepsilon(n+1)(\phi)^T(n)P(n-1)}{\lambda + (\phi)^T(n)P(n-1)\phi(n)}, \\
P^{-1}(n) &= P^{-1}(n-1) + \phi(n)(\phi)^T(n) \\
&\quad + (\lambda - 1)\frac{(\phi)^T P(n-1)\phi(n)}{[(\phi)^T(n)\phi(n)]^2}\phi(n)(\phi)^T(n).
\end{aligned}$$

where P is the covariance matrix, $\lambda \in (0,1)$ is the forgetting factor and $\hat{X}$ is the estimate of X. Notice from (2) that the estimate $\hat{X}$ contains the desired estimate of the parameters of the systems.

5.2 Linear Quadratic Control

With the estimate of the MIMO system of the cloud, we proceed to design a Linear Quadratic control that minimizes the following cost function,

$$J = (\mathbf{y}(k) - \mathbf{y}_{ref}(k))^T Q(\mathbf{y}(k) - \mathbf{y}_{ref}(k)) + u^T(k)Ru(k), \quad (3)$$

where the matrix $Q \in \Re^{p\times p}$ is positive semi-definite and determines the level of accuracy of the regulated states, and $R \in \Re^{q\times q}$ is a positive definite matrix that penalizes the control effort [12].

By assuming no effect from the noise vector $\mathbf{w}$ we rewrite the system as,

$$\mathbf{y}(n) = \mathbf{y}(n-1) + B_0 u(n)$$

then, from [19] we get the following control law that minimizes the cost function (3), while guaranteeing the stability of the system,

$$\mathbf{u}(n) = \left((W\hat{B}_0)^T W\hat{B}_0 + Q^T Q\right)^{-1} \times$$
$$\left[(W\hat{B}_0)^T W\left(\mathbf{y}_{ref}(n+1) - \hat{X}(n)\tilde{\phi}(n)\right) + Q^T Q\mathbf{u}^T(n-1)\right].$$

Through this approach, we should be able to regulate the performance measures of the system that are covered by the SLA and implemented through our policy based usage management of the cloud. One of the main concerns in this approach is the implementation of our linear quadratic control to an IaaS environment. Specifically, in the EC2 service, the variations within Virtual Machines (VMs) instances is limited to some discrete values, namely, small, large and extra-large intances. Therefore, the discrete-state and discrete-time properties of the system imply some considerations when the SLA imposes references of the performance measures that are not reachable given the discrete-state constrain of the system. However, to overcome this particular issue, we restrict the regulation problem to satisfy only feasible values for all performance measures to avoid oscillations or overshoots on the states.

6. CONCLUSIONS AND FUTURE WORKS

Usage management is a common problem set with features embodied in domains ranging from security systems to video games to music production and retail. The ability to provide management of resources with regard to authorized subjects is being addressed in multiple different forums, many of which are taking remarkably different approaches. Common features however generally include the need for either ubiquitous rights expression language acceptance or for extensive translation between all supported rights languages.

In this paper, we first described the primitives and approaches currently available that use used to enable simple SLA-centric control over information distribution and processing based on performance and security sensitivity attributes. Thereafter, we described in some detail a system architecture currently realizable with modern open-source tools that enables this kind of dynamic information control. Finally, we discussed our experiences with a prototypical implementation of our proposed system architecture.

In the future, we currently plan to move away from our current web-centric model, examining more data-centric tooling, though we expect to remain committed to open source tools. We will also incorporate more standards, like the eXtensible Access Control Markup Language (XACML), to describe policies and controls, and work to establish a clear model behind our work in order to more deeply understand the intrinsic limitations of this problem domain.

7. REFERENCES

[1] Standards for security categorization of federal information and information systems. http://purl.access.gpo.gov/GPO/LPS75866, February 2004.

[2] DoD Information Sharing Strategy. http://cio-nii.defense.gov/docs/InfoSharingStrategy.pdf, May 2007.

[3] Assured Information Sharing in Clouds. http://www.zyn.com/sbir/sbres/sttr/dod/af/af11-bt30.htm, August 2011.

[4] NSA Pursues Intelligence-Sharing Architecture. http://www.informationweek.com/news/government/cloud-saas/229401646, April 2011.

[5] S. Abdelwahed, J. Bai, R. Su, and N. Kandasamy. On the application of predictive control techniques for adaptive performance management of computing systems. *Network and Service Management, IEEE Transactions on*, 6(4):212 –225, 2009.

[6] H. Alverstrand. The role of the standards process in shaping the internet. *Proceeding of the IEEE*, 92(9):1371–1374, 2004.

[7] Y. Ariba, F. Gouaisbaut, and Y. Labit. Feedback control for router management and tcp/ip network stability. *Network and Service Management, IEEE Transactions on*, 6(4):255 –266, 2009.

[8] M. S. Blumenthal and D. D. Clark. Rethinking the design of the Internet: The end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology*, 1(1):70–109, Aug. 2001.

[9] R. Buyya. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 1–, Washington, DC, USA, 2009. IEEE Computer Society.

[10] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[11] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: Defining tomorrow's internet. In *SIGCOMM*, pages 347–356, Pittsburg, Pennsylvania, USA, Aug. 2002.

[12] P. Dorato, C. T. Abdallah, and V. Cerone. *Linear Quadratic Control: An Introduction*. Krieger Publishing Company, 2000.

[13] J. Hellerstein, S. Singhal, and Q. Wang. Research challenges in control engineering of computing systems. *Network and Service Management, IEEE Transactions on*, 6(4):206 –211, 2009.

[14] P. A. Jamkhedkar, G. L. Heileman, and C. C. Lamb. An interoperable usage management framework. In *Proceedings of the tenth annual ACM workshop on Digital rights management*, DRM '10, pages 73–88, New York, NY, USA, 2010. ACM.

[15] M. Kjaer, M. Kihl, and A. Robertsson. Resource allocation and disturbance rejection in web servers using slas and virtualized servers. *Network and Service Management, IEEE Transactions on*, 6(4):226 –239, 2009.

[16] R. Kulhavý. Restricted exponential forgetting in real-time identification. *Automatica*, 25(5):589–600, 1987.

[17] C. C. Lamb, P. A. Jamkhedkar, G. L. Heileman, and C. T. Abdallah. Managed control of composite cloud systems. In *6th IEEE International Conference on System of Systems Engineering (SOSE)*. IEEE.

[18] C. C. Lamb, P. A. Jamkhedkar, G. L. Heileman, and C. T. Abdallah. Managed control of composite cloud systems. In *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pages 167 –172, june 2011.

[19] X. Liu, X. Zhu, P. Pradeep, Z. Wang, and S. Sharad. Optimal multivarite control for differentiated services on a shared hosting platform. In *IEEE Conference on Decision and Control*, pages 3792–3799, New Orleans, LA, December.

[20] J. M. Luna and C. T. Abdallah. Control in computing systems: Part ii. In *IEEE Multi-Conference on Systems and Control*, Denver, CO, September 2011.

[21] R. Nathuji, A. Kansal, and A. Ghaffarkhah. Q-clouds: Managing performance interference effects for qos-aware clouds. In *Proceedings of the ACM European Society in Systems Conference 2010*, pages 237–250, Paris, France, April 2010.

[22] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693 –702, 30 2010-dec. 3 2010.

[23] Z. Wang, Y. Chen, D. Gmach, S. Singhal, B. Watson, W. Rivera, X. Zhu, and C. Hyser. Appraise: application-level performance management in virtualized server environments. *Network and Service Management, IEEE Transactions on*, 6(4):240 –254, 2009.

[24] J. Yao, X. Liu, X. Chen, X. Wang, and J. Li. Online decentralized adaptive optimal controller design of cpu utilization for distributed real-time embedded systems. In *Proceedings of the 2010 American Control Conference (ACC'10)*, pages 283–288, Baltimore, MD, June 2010.

[25] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin. What does control theory bring to systems research? *SIGOPS Oper. Syst. Rev.*, 43:62–69, January 2009.