Managed Control of Composite Cloud Systems

Christopher Lamb **Pramod Jamkhedkar** **Greg Heileman** **Chaouki Abdullah**
Department of Electrical and Computer Engineering
University of New Mexico
{cclamb, pramod54, heileman, chaouki}@ece.unm.edu

Abstract - *Cloud providers have just begun to provide primitive functionality enabling users to configure and easily provision resources, and primarily in the Infrastructure as a service domain at that. In order to effectively manage cloud resources in an automated fashion, systems must automate quality of service metric measurement as a part of a larger usage management strategy. Collected metrics can then be used within control loops to manage and provision cloud resources when needed. This basic approach can be scaled to monitor the use of system artifacts as well as simple quality-of-service parameters, and can also address the needs of large systems spanning the boundaries of single service providers.*

Keywords: Usage Management, Cloud Computing, System of Systems.

1 Introduction

Cloud computing services as a computational paradigm are more market oriented than previous attempts at commodity computing. Furthermore, they are in many cases designed to be composed into larger, more powerful customer facing systems. These kinds of aggregate systems fit neatly into one of the more commonly used definitions of a system of systems as well [1], [2]. With so much data in the hands of different providers in an aggregate system, system developers and users are hard-pressed to effectively monitor and control the use of sensitive content by various composite systems. Some of this information can be contained in Service Level Agreements (SLAs), but they have thus far been focused on quality-of-service (QoS) metrics rather than addressing issues like data flow or physical application residency. For the most part SLAs are simply not sufficient for addressing usage management concerns [3], [4], [5], [6].

Effective usage management monitoring coupled with feedback processing creates an event loop suitable for applying control theoretic concepts to cloud infrastructures.

Usage policies specified at a fine-grained level provides cloud service users with more reliability of the use of their data within cloud centric systems. For example, data routing, caching, or hosting can be a sensitive issue for some systems in that specific users may want to restrict the countries that can access that data. The ability to specify and control where specifically that data travels and resides gives those kinds of sensitive users confidence to use could-centric computing resources. Furthermore, this kind of control will also facilitate cost profiles for services that more closely match demand, giving providers better control over their infrastructure and additional areas for product differentiation.

Herein, we will elaborate the idea of applying usage management to single and distributed could systems. In this brief analysis, we will touch on the application of common system design principles and standards [7], [8], [9], application of usage control concepts [10], [11], policy language application [12], digital rights management (DRM) systems [13], and interoperability [14], [15], [16], [17], [18]. We will apply these ideas toward a controllable feedback-enabled system suitable for cloud system control.

In Section 2 this paper first addresses how to create a controllable system with feedback suitable for system evaluation from the perspective of a single provider. Here, we will address the constraints and advantages of such an approach and how providers could begin to offer these kinds of services. In this first example, we will focus on QoS data specifically. Next in Section 3 we will extend our single provider system to provide control over attributes more specific to the usage management domain, with examples and associated analysis. Finally in Section 4 we extend this single provider model to a more realistic system deployed to multiple cloud providers in a realistic system-of-systems scenario.

1.1 Previous Work

Could computing is emerging as the future of utility systems hosting for consumer-facing systems. In these kinds of systems, components, applications, and hardware are pro-

vided as utilities over the Internet with associated pricing schemes pegged by system demand. Users accept specific QoS guidelines that providers use to provision and eventually allocate resources. These guidelines become the basis over which providers charge for services.

Over the past few years multiple service-based paradigms like web-services, cluster computing and grid computing have contributed to the development of what we now call cloud computing [19]. Cloud computing distictly differentiates itself from other service-based computing paradigms by via a collective set of distinguishing characteristics: market orientation, visualization, dynamic provisioning of resources, and service composition via multiple service providers [20]. 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 of 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). 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.

2 Single Provider Feedback System

Controllable cloud systems enable providers to supply more closely targeted, cost effective services while at the same time providing service consumers with the confidence that the data and other artifacts their systems use are protected. With that as our eventual goal, we first begin with a simple system managing currently accepted QoS parameters - system attributes like bandwidth, system memory allocation, and the like. Specifically, we intend to provide the ability to monitor and control a virtual system hosted on a cloud infrastructure so that response time for a hosted application falls within a specific range of accepted parameters.

In order to manipulate a system to meet a preselected threshold of performance metrics, we must have access to measurement information with respect to factors effecting those metrics and we need to be able to adjust system performance in response to those measurements. We are going to focus on system response time measured at the edge of the cloud provider's infrastructure as the metric we wish to control. We are going to adjust system performance to meet that metric by manipulating the number of processing nodes, bandwidth available to those nodes, and node RAM allocations.

We have identified the attributes we wish to control. This leads us to a group of requirements we can use to assemble a logical system architecture. Requirements we know we need to address include:

- *Performance:* We will be adjusting a system within specific soft real-time frames. Ergo, we need to be able to collect feedback measurements, process those measurements, and make decisions about how to respond to those measurements quickly in order to avoid falling out of compliance with any performance parameters to which we must adhere.

- *Accessibility:* In order to control component systems, we must be able to access those systems. In order to do so within time constraints, we must be able to access those systems electronically as well; physical access requirements simply will not scale into this performance domain.

- *Controllability:* We must be able to access the appropriate control primitives on the systems we need to tune. This will include accessing compute node generation and termination capabilities. It would help if we could access node performance information and tune those nodes as well, though this is not required; we can emulate this by terminating nodes in one configuration and creating nodes with another to more adequately address performance needs.

These system attributes lead us to a system architecture that is beginning to look like a traditional feedback-centric controllable system.

[insert UML diagram]

The above system has four primary components, one of which has three subcomponents. These components work together to provide cloud services to consumers in a hypothetical Infrastructure-as-a-Service scenario. This particular view addresses logical, functional components of this kind of a system rather than specific technologies used in an implementation, although some components are loosely modeled on popular open-source cloud environments (i.e. Eucalyptus).

Essentially, we have a cloud controller element that initiates provisioning of compute nodes within a given cluster. The cluster itself is managed by a cluster controller, which in turn controls storage controllers, node controllers, and by

extension, nodes. The nodes and node controllers themselves are monitored by a resource allocation controller which refers to a set of QoS requirements.

- *Cloud Controller:* Provides an initial interface to administrative users to control the cloud.
- *Cluster Controller:* Managed by the cloud controller, the cluster controller manages the resources of a single cluster. A given cloud may contain multiple clusters.
- *Storage Controller:* Provides storage of system images and for other general storage needs. This controller component is highly I/O sensitive.
- *Node Controller:* Responsible for allocating, delivering, and managing individual compute nodes upon which client software runs. This is the primary computational resource accessed by users accessing managed cloud resources.
- *Node:* The compute node delivering services to end users and managed by the cluster's control infrastructure.
- *QoS:* Quality of service terms the cloud provider has agreed to honor for the cloud customer with respect to system delivery, provisioning, and overall performance.
- *Resource Allocation Controller:* The component responsible for real-time tuning of the cloud system to maintain defined quality of service.

Operationally, the initial commands required to initialize the cloud are delivered from the *Could Controller* to the *Cluster Controller*, who then propagates another, related set of commands provisioning an initial set of resources from the *Node Controllers* and the *Storage Controllers*. At this point, the initial system has been configured and is running, serving hosted software to it's customer base.

Once the system is running, state data describing the performance metrics of interest is dispatched from *Nodes* on the *Node Controllers* and the *Storage Controller* and delivered to the *Resource Allocation Controller*. The *Resource Allocation Controller* then processes this new event data in the context of the defined *QoS* parameters. This processing, in this model, is likely to be simple processing over the current event package or perhaps the current and the otherwise most recent event package. This evaluation is very performance sensitive; we need to process the state of the current system quickly and adjust resource allocation accordingly. Because of this soft real-time requirement, we do not have the luxury of spending significant time reviewing trending or providing sophisticated analysis over delivered event information. Note that extension of this system into the feedforward domain would allow this kind of more robust system management, allowing us to employ more complex and powerful machine learning or neural systems to predict system needs.

Finally, if needed the *Resource Allocation Controller* will dispatch messages to the *Cluster Controller*, *Nodes*, and *Storage Controller* adjusting system profiles to ensure they remain within acceptable performance ranges.

[cast in control theoretic terms, include block diagram]

This control infrastructure as it has strict timing requirements with respect to event collection, analysis, and control, likely needs to be hosted in close physical proximity to the controlled systems. Otherwise, the systems themselves can be located just about anywhere accessible to the Internet. These logical components are not necessarily all hosted on physically distinct systems either, though generally at least the *Storage Controller* and the *Node Controller* are as they have remarkably different requirements with respect to processing power and I/O throughput.

Clearly, both the cloud service consumer and providers are impacted by this kind of infrastructure. Consumers have systems performing within required performance bounds, and providers are no longer required to maintain as strict administrative over-watch of managed systems. This kinds of systems may also impact system developers, as this kind of dynamic node control and allocation imparts new requirements with respect to intra-system data handing and processing. Generally however, accepted service development guidelines with respect to statelessness and allowing running processes to terminate prior to node shutdown will alleviate these issues.

When implemented, this kind of system will provide dynamic runtime control of cloud systems enhancing provider and customer confidence in the hosted infrastructure's performance potential. This can also be extended into the usage management realm with more specific requirements with respect to how customer artifacts are managed, not just delivered.

3 Single Provider Feedback System with Usage Management

Now that we have developed a cloud system capable of fairly granular control via a feedback control loop using QoS parameters, we will being to incorporate specific usage management parameters. Usage management, as a field, extends simple access control and digital rights management perspectives with regard to resources. While access control

is concerned with controlling access to a specific protected resource, usage management incorporates control over that initial access and extends into how that resource is used after that initial access. Likewise, while digital rights management technologies do exert some level of control over the use of artifacts, they have generally been applied in strict media domains like music, film, or digitized images. In this system we will apply usage management to specific non-traditional artifacts to control how they are used and where they may be stored.

In order to provide control over customer data artifacts in a cloud environment, we need to first establish a basic mechanism for doing so. This system must fit within the functional confines of the QoS system from Section 2 while extending the QoS functionality to artifacts not generally controlled via traditional QoS metrics. For these purposes, a good example of an artifact not generally controlled via QoS parameters could be streaming network data. While bandwidth throttling is clearly in the QoS domain, more specific uses of that data stream like caching and routing are not.

Using a data stream as an example, we recognize some situations we clearly need to be able to control. In this example, we will limit ourselves to a data stream emitted from a *Node* on a *Node Controller* which is routed to a user as a result of a user request. Here, we have control over stream creation. We want to limit the ability to update that data stream, we certainly don't want that stream deleted, and we want to limit who may read that stream. In fact, we can safely assume in this scenario that update and deletion are operations we want to completely forbid, while we may want to limit stream readability, leading us to the primary new requirement when adding usage management over a network stream in this case:

- *Accessibility:* Data streamed through the cloud system must be able to be monitored and the accessibility of that stream needs to be dynamically tunable. This implies that we need to be able to control routing and caching of all streaming data according to user specified conditions. This also implies that we need to be able to control exactly which *Node Controllers* are able to spawn which *Nodes*.

The addition of these attributes and requirements give us this logical system:

[insert UML diagram w/ UM & network components]

(what; present UML diagram of single system with QoS) The new system has new associations and components required to implement the degree of control required to limit the accessibility of the network stream. We have added *Cache Controllers* and *Switches*, defined as:

- *Cache Controllers:* Streaming network data, specifically media-centric streams, can and are cached by strategically located cache systems. In order to control the read access of network data, we must be able to exercise explicit control over any caching systems in our infrastructure.

- *Switch:* Really any kind of hardware that controls the delivery of network data. This component includes switches and routers primarily. In order to control how data is accessed we must be able to control the locations to which it is delivered.

We have also added a new relationship to enable control over the *Cloud Controller*. To ensure that we can control where data is at any given time, we must also be able to control the geographic areas from which data is generated, especially if the virtual compute cloud spans national boundaries.

This again forms a controllable feedback loop, though one that is more complex than the simple QoS case. The addition of new controllable items and relationships increases the responsibility and complexity of the *Resource Controller* as new logic and capabilities are added to facilitate measurement and control of the larger system.

4 Scaling to Multiple Providers

[scale to multiple providers accessing common QoS]

5 Conclusion

In this paper we introduced the notion of usage management in cloud computing environment. Cloud computing exhibits a unique set of characteristics that will require usage management of users' data according to user concerns and expectations. We analyzed the challenges involved in the design and development of a framework for usage management in cloud environments. We showed that such a framework needs to be open to leverage existing security technologies and SLA frameworks. The framework needs to exhibit features such as support for multiple policy languages, existence of a common cloud ontology, dynamic interpretation and data transformations. Finally a preliminary framework that supports multiple policy languages was introduced that will provide a platform upon which such a framework can be built.

Future works involves efforts towards the development of a common extensible cloud ontology that will provide the vocabulary underlying this framework and enable interoperability. Such an ontology needs be developed by taking into consideration the requirements of different cloud services. In addition, it is necessary to standardize interfaces for other

security mechanisms mentioned in the paper that can be incorporated within the framework. Finally, the framework allows use of multiple policy languages, to this effect, existing policy languages can be modified to be incorporated within the framework, and news ones need to designed to address the specific needs of different cloud services.

References

[1] A. P. Sage and C. D. Cuppan, "On the systems engineering and management of systems of systems and federations of systems," *Inf. Knowl. Syst. Manag.*, vol. 2, pp. 325–345, December 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=1234195.1234200

[2] "IEEE SMC 2005 Conference Theme," http://ieeesmc2005.unm.edu/SoSE_Defn.htm, March 2011.

[3] *Web Services Agreement Specification*, Open Grid Forum, Mar. 2007. [Online]. Available: http://www.ogf.org/documents/GFD.107.pdf

[4] *Web Service Level Agreement (WSLA) Language Specification*, IBM, Jan. 2003. [Online]. Available: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[5] *Web Services Policy 1.5 Framework*, World Wide Web Consortium, Sep. 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-ws-policy-20070904/

[6] P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in Cloud Computing," in *Proceedings of the Workshop on Best Practices in Cloud Computing: Implementation and Operational Implications for the Cloud at ACM International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Orlando, FL, Oct. 2009.

[7] 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*, vol. 1, no. 1, pp. 70–109, Aug. 2001.

[8] D. D. Clark, "The design philosophy of the DARPA internet protocols," in *ACM SIGCOMM*, Stanford, CA, Aug. 1988, pp. 106–114.

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

[10] J. Park and R. Sandhu, "The UCON$_{ABC}$ usage control model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128–174, 2004.

[11] P. A. Jamkhedkar, G. L. Heileman, and C. Lamb, "An Interoperable Usage Management Framework," in *Proceedings of the Tenth ACM Workshop on Digital Rights Management*, Chicago, Oct. 2010.

[12] P. A. Jamkhedkar, G. L. Heileman, and I. Martinez-Ortiz, "The problem with rights expression languages," in *Proceedings of the Sixth ACM Workshop on Digital Rights Management*, Alexandria, VA, Nov. 2006, pp. 59–67.

[13] P. A. Jamkhedkar and G. L. Heileman, "Digital Rights Management Architectures," *Computers Electrical Engineering*, vol. 35, no. 2, pp. 376–394, 2009.

[14] ——, "DRM as a Layered System," in *Proceedings of the Fourth ACM Workshop on Digital Rights Management*, Washington, DC, Oct. 2004, pp. 11–21.

[15] G. L. Heileman and P. A. Jamkhedkar, "DRM interoperability analysis from the perspective of a layered framework," in *Proceedings of the Fifth ACM Workshop on Digital Rights Management*, Alexandria, VA, Nov. 2005, pp. 17–26.

[16] R. H. Koenen, J. Lacy, M. MacKay, and S. Mitchell, "The long march to interoperable digital rights management." *Proceedings of the IEEE*, vol. 92, no. 6, pp. 883–897, 2004.

[17] "Coral consortium whitepaper," Tech. Rep., Feb. 2006. [Online]. Available: www.coral-interop.org/main/news/Coral.whitepaper.pdf

[18] "Marlin architecture overview," Tech. Rep., 2006. [Online]. Available: www.marlin-community.com

[19] 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*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.97

[20] 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*, vol. 25, no. 6, pp. 599–616, 2009.