Monitoring Service Clouds in the Future Internet

Stuart CLAYMAN [a], Alex GALIS [a], Clovis CHAPMAN [b], Giovanni TOFFETTI [c],
Luis RODERO-MERINO [d], Luis M. VAQUERO [d], Kenneth NAGIN [e],
Benny ROCHWERGER [e]
e-mail: sclayman@ee.ucl.ac.uk e-mail: a.galis@ee.ucl.ac.uk
[a] *Dept. of Electronic Engineering, University College London (UCL)*
e-mail: c.chapman@cs.ucl.ac.uk [b] *Dept. of Computer Science, UCL*
e-mail: toffettg@lu.unisi.ch [c] *University of Lugano*
e-mail: rodero@tid.es, lmvg@tid.es [d] *Telefonica I+D, Madrid*
e-mail: nagin@il.ibm.com, rochwer@il.ibm.com [e] *IBM Research, Haifa*

Abstract. Service Clouds are a key emerging feature of the Future Internet which will provide a platform to execute virtualized services. To effectively operate a service cloud there needs to be a monitoring system which provides data on the actual usage and changes in resources of the cloud and of the services running in the cloud. We present the main aspects of Lattice, a new monitoring framework, which has been specially designed for monitoring resources and services in virtualized environments. Finally, we discuss the issues related to federation of service clouds, and how this affects monitoring in particular.

Keywords. monitoring, cloud computing, service cloud, federation, service management, SLA

Introduction

The Future Internet will have many of its core features and functions based on virtualized resources. The virtualization of resources has been in use for some time now, and there are several projects and initiatives working towards the virtualization of networks in the Future Internet, such as [1] [2] [3] [4]. The virtual resources which will be part of the Future Internet offerings also include virtual machines which will execute either virtual routers or virtualized service elements. These virtual resources will be combined together to create *service clouds*. In order to manage these virtual resources, there needs to be a monitoring system which can collect and report on the behaviour of these resources.

This paper presents the Lattice monitoring framework which can be used as a fundamental part of the management of an Internet scale deployment of Service Clouds. Lattice has been designed for monitoring resources and services in virtualized environments. The monitoring of virtualized resources and services has many features, requirements, and criteria which are not relevant for monitoring systems that are used for traditional fixed network resources.

Traditionally, the architecture for monitoring has needed to adapt to an arbitrary network structure, but in a Service Cloud environment the infrastructure is structured as a federation of large clusters of distributed hosts. Consequently, we can design and build a monitoring system that factors in this major difference.

In the work presented in this paper we use the Lattice monitoring framework as a real-time feed for the management of a service cloud. Monitoring is a fundamental aspect of Future Internet elements, and in particular for service clouds, where it is used for both the infrastructure and service management. We first present the issues relating to the management of service clouds, discussing the key design requirements and how these are addressed in the RESERVOIR project [5]. Then, in section 2 we present the Lattice monitoring framework, discussing its main features and also giving and overview of its design and implementation, together with a presentation of its use within RESERVOIR. This is followed, in section 3, by a presentation of the issues related to federation within service clouds and how this affects monitoring in particular. Finally, we state our conclusions and suggest further work.

1. SERVICE CLOUDS

The emerging *cloud computing* paradigm [6] [7] [8] [9] for hosting Internet-based services in virtualized environments, as exemplified by the Amazon Elastic Compute Cloud (EC2) [10] or Google's AppEngine [11], aims to facilitate the creation of innovative Internet scale services without worrying about the computational infrastructure needed to support such services. At present, no single hosting company can create a seemingly infinite infrastructure capable of serving the increasing number of on-line services, each having massive amounts of users and access at all times, from all locations. To cater to the needs of service providers, it is inevitable that the *service cloud* is going to be composed of a federation of sites from various infrastructure providers [12]. Only by partnering and federating with each other, can infrastructure providers take advantage of the diversity factor and achieve the economies of scale needed to provide a seemingly infinite compute utility.

Service clouds are just the latest incarnation of a concept that has been around since the 1960's, namely the manifestation of a general-purpose public computing utility [13]. Throughout the history of computing we have seen such utilities appear in one form or another. Even though some success stories exist, such as in in the area of high performance scientific computing, where grid computing [14] [15] [16] made significant progress over the past decade, none of these attempts materialized into a true general purpose compute utility that is accessible by anyone, at any time, from anywhere. Now however, the advent of new approaches utilizing an always-on Internet and virtualization [2], has brought about system designs which will enable the desired progress [17]. An example of such a system design is the RESERVOIR Compute Cloud, which is described in the next section.

1.1. RESERVOIR

RESERVOIR [5] is a *service cloud* which has a new and unique approach to Service-Oriented cloud computing [18]. In the RESERVOIR model there is a clear separation between service providers and infrastructure providers. Service providers are the entities that understand the needs of particular business and offer service applications to address those needs. Service providers do not need to own the computational resources needed by these service applications, instead, they lease resources from an infrastructure provider. The infrastructure provider owns or leases a computing cluster, which supplies the service provider with a seemingly infinite pool of computational resources. The cluster is presented as a service cloud site which is capable of allocating resources to many service providers.

The high-level objective of RESERVOIR is to significantly increase the effectiveness of the compute utility enabling the deployment of complex services on a service cloud that spans infrastructure providers and even geographies, while ensuring QoS and security guarantees. In doing so, RESERVOIR will provide a foundation where resources and services are transparently and flexibly provisioned and managed like utilities.

1.2. RESERVOIR Architecture

The essence of the RESERVOIR service cloud is to effectively manage a service which has been specified as a collection of virtual execution environments (VEEs). A service cloud, such as RESERVOIR [12] [17], operates by acting as a platform for running virtualized applications in VEEs, which have been deployed on behalf of a service provider. The service provider defines the details and requirements of the application in a Service Definition Manifest. This is done by specifying which virtual machine images are required to run, as well as specifications for (i) Elasticity Rules, which determine how the application will scale across a cloud, and (ii) Service Level Agreement (SLA) Rules [19], which determine if the cloud site is providing the right level of service to the application. Within each service cloud site there is a Service Manager (SM) and a VEE Manager (VEEM) which provide all the necessary management functionality for both the services and the infrastructure. These management components of a cloud system are presented in more detail here.

The Service Manager (SM) is responsible for the instantiation of the service application by requesting the creation and configuration of VEEs for each service component in the manifest. In addition, it is the Service Manager that is responsible for (i) evaluating and executing the elasticity rules and (ii) ensuring SLA compliance [20], by monitoring the execution of the service applications in real-time. Elasticity of a service is done by adjusting the application capacity, either by adding or removing service components and/or changing the resource requirements of a particular component according to the load and measurable application behaviour.

The Virtual Execution Environment Manager (VEEM) is responsible for the placement of VEEs into VEE hosts (VEEHs). The VEEM receives requests from the Service Manager to create VEEs, to resize VEEs, and to also finds the best placement for these VEEs in order to satisfy a given set of constraints. The role of the VEEM is to optimize a site and its task is to place and move the VEEs anywhere, even on remote sites, as long as the placement is done within the constraints set in the Manifest, including specifications of VEE affinity, VEE anti-affinity, security, and cost. In addition to serving local requests, the VEEM is the component in the system that is responsible for the federation of VEEs to and from remote sites.

The Virtual Execution Environment Host (VEEH) is a resource that can host a certain type of VEEs. For example one type of a VEEH can be a physical machine with the Xen hypervisor [21] controlling it, whereas another type can be a machine with the

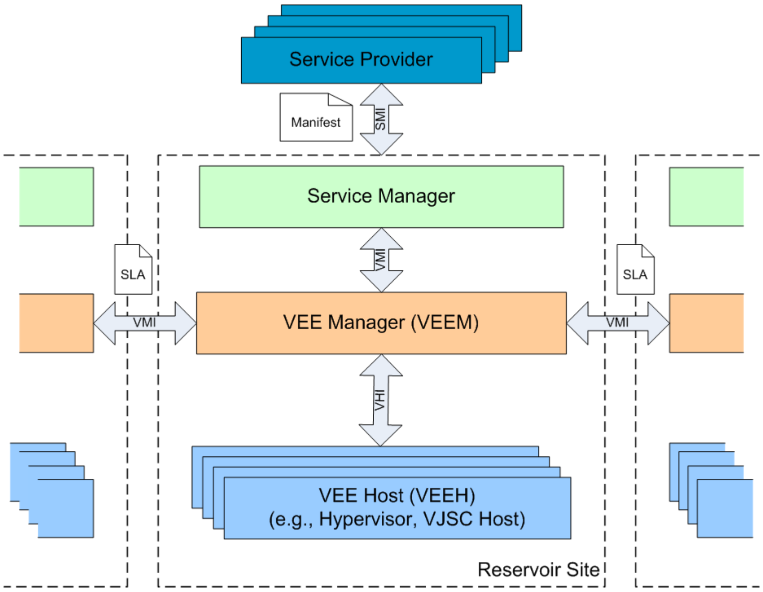

Figure 1. RESERVOIR Service Cloud Architecture

kvm hypervisor [22]. In a service cloud hosting site there is likely to be a considerable number of VEEHs organised as a cluster.

These three main components of the service cloud architecture interact with each other using well defined interfaces, namely SMI, VMI, and VHI, within a site and also use the VMI interface for site-to-site federation via the VEEM. In figure 1 the relationship between these components and interfaces is shown.

As can be seen in figure 1, the RESERVOIR platform has a Service Provider specifying the details and requirements of his application in the Service Definition *Manifest*. The Manifest also has the specifications of Elasticity Rules, which determine how the application will scale across the cloud, and the Service Level Agreement (SLA) Rules [20], which determine if the platform is providing the right level of service to the application.

Within each site the resource utilization is monitored and the placement of VEEs is constantly updated to achieve optimal resource utilization. Similarly, the execution of the service applications is monitored and their capacity is constantly adjusted to meet the requirements specified in the manifest. These on-going optimizations are done without human intervention by the management components. Within the Service Manager there are components which dynamically evaluate Elasticity and SLA rules and implement their actions in order to maintain effective running of the application. For these rules to be evaluated A service cloud needs a monitoring sub-system which sends measurement data, collected from the whole system, to be fed into the rules [23]. This measurement data can come from the following sources:

1. raw data gathered from probes in the underlying infrastructure,
2. raw data gathered from probes attached to the virtual machines,
3. data gathered from probes embedded in the application,
4. data which is the combination of the raw data into composed Key Performance Indicators (KPIs), and

5. data derived from the analysis of historical raw data and KPI data held in a data store.

In a large distributed system, such as a RESERVOIR service cloud, there may be hundreds or thousands of measurement probes which can generate data. It would not be effective to have all of these probes sending data all of the time, so a mechanism is needed that controls and manages the relevant probes. Therefore, a service cloud requires a monitoring system that has a minimal runtime footprint and is not intrusive, so as not to adversely affect the performance of the cloud itself or the running service applications. As a consequence, we need to ensure that components such as the VEEM, and Service Manager elements such as the Elasticity Rules Engine and the SLA Rules Engine only receive data that is of relevance.

2. REQUIREMENTS FOR MONITORING SERVICE CLOUDS

Existing monitoring systems such as Ganglia [24], Nagios [25], MonaLisa [26], and GridICE [27] have addressed monitoring of large distributed systems, but they have not addressed a rapidly changing and dynamic infrastructure seen in service clouds.

The monitoring system needs to be designed and built to be fit for the purpose of infrastructure and service monitoring. It needs to be for the whole of service management, and so it should cover SLAs, elasticity, QoS, etc. It is important to recognise that it is the monitoring mechanism that closes the loop from the initial deployment, through execution, and back to the Service Manager. The monitoring system is there to gather data from all the components within a cloud architecture and so monitoring is a fundamental aspect of a service cloud that is used by the infrastructure and for service management.

The monitoring system for a service cloud needs to feed data into the Service Manager so that it can manage the services deployed on the cloud. Over time it is expected that the management capabilities of the Service Manager will expand to include new functions. As a consequence we need the monitoring system to be adaptable, flexible, and extensible in order to support the expanding functionality.

To address all of the requirements and functionality of the service cloud environment we have determined that the main features for monitoring which need to be taken account of are:

- *scalability* - to ensure that the monitoring can cope with a large numbers of probes
- *elasticity* - so that virtual resources created and destroyed by expanding and contracting networks are monitored correctly
- *migration* - so that any virtual resource which moves from one physical host to another is monitored correctly
- *adaptability* - so that the monitoring framework can adapt to varying computational and network loads in order to not be invasive
- *autonomic* - so that the monitoring framework can keep running without intervention and reconfiguration
- *federation* - so that any virtual resource which reside on another domain is monitored correctly

To establish such features in a monitoring framework requires careful architecture and design. In the following section presents the Lattice Monitoring Framework which we have designed and built for the purpose of monitoring environments such as service clouds.

2.1. *The Lattice Monitoring Framework*

A management system for service cloud requires a monitoring system that can collect all the relevant data in an effective way. The monitoring system has to have a minimal runtime footprint and not be intrusive, so as not to adversely affect the performance of the network itself or the running service applications. As a consequence, we need to ensure that the management components only receive data that is of relevance. In a large distributed system there may be hundreds or thousands of measurement probes which can generate data. It would not be effective to have all of these probes sending data all of the time, so a mechanism is needed that controls and manages the relevant probes.

2.1.1. *Producers and Consumers*

The monitoring system itself is designed around the concept of producers and consumers. That is there are producers of monitoring data, which collect data from probes in the system, and there are consumers of monitoring data, which read the monitoring data. The producers and the consumers are connected via a network which can distribute the measurements collected.

The collection of the data and the distribution of data are dealt with by different elements of the monitoring system so that it is possible to change the distribution framework without changing all the producers and consumers. For example, the distribution framework can change over time, say from IP multicast, to an event bus, or a publish / subscribe framework. Making this change should not affect other parts of the system.

2.1.2. *Data Sources and Probes*

In many monitoring systems probes are used to collect data for system management [28] [24]. In this regard, Lattice follows suit. However, to increase the power and flexibility of the monitoring we introduce the concept of a data source. A data source represents an interaction and control point within the system that encapsulates one or more probes. A probe sends a well defined set of attributes and values to the consumers. This can be done by transmitting the data out at a predefined interval, or transmitting when some change has occured.

The measurement data itself is sent via a distribution framework. These measurements are encoded to be a small as possible in order to maximise the network utilization. Consequently, the measurement meta-data is not transmitted each time, but is kept separately in an information model. This information model can be updated at key points in the lifecycle of a probe and can be accessed as required by consumers.

The goal for the monitoring system is to have fully dynamic data sources, in which each can have multiple probes, with each probe returning its own data. The data sources are able to turn on and turn off probes, or change their sending rate on demand.

2.1.3. *Data Distribution Mechanism*

In order to distribute the measurements collected by probes within the monitoring system, it is necessary to use a mechanism that fits well into a distributed architecture such as a service cloud. We need a mechanism that allows for multiple submitters and multiple receivers of data without having vast numbers of network connections. For example, having many TCP connections from each producer to all of the consumers of the data for that producer would create a combinatorial explosion of connections. Solutions to this include IP multicast, Event Service Bus, or publish/subscribe mechanism. In each of these, a producer of data only needs to send one copy of a measurement onto the network, and each of the consumers will be able to collect the same packet of data concurrently from the network. Such an approach works well for a service cloud because we have both multiple producers and multiple consumers.

2.2. Utilization of Lattice within RESERVOIR

Using the Lattice framework we have built a monitoring system suitable for service cloud management as part of the RESERVOIR project. As stated earlier, measurement data can come from raw data gathered from probes in the underlying infrastructure and from raw data gathered from probes attached to the virtual machines. Consequently, we have written probes for monitoring physical resources and for monitoring virtual resources. Furthermore, we have embedded probes inside the applications themselves in order to gather application specific data. These probes have been deployed into a testbed which implements the RESERVOIR functionality and are evaluated here.

2.2.1. Monitoring Physical Resources

To monitor the underlying infrastructure of the testbed we have created three different probes. The probes monitor CPU usage, memory usage, and network usage of each VEEH in the testbed. First, the CPU probe collects data on CPU usage for each core of a server. It runs regularly in order to collect this data in real time. Second, the memory probe collects data on memory usage on the server. It too runs regularly to collect data in real time. Third, the network probe collects data on the network interface of a server. By running at a regular interval it can determine the amount of traffic on the network interface for each sample.

2.2.2. Monitoring Virtual Resources

To monitor VEEs we have created some probes that collect data on the virtual machines running on a particular host. These probes interact with the hypervisor to collect data on each VEE executing under hypervisor control. Each of these probes runs on a regular basis, collecting data on CPU usage, memory usage, and network usage of each VEE.

2.2.3. Monitoring Service Applications

To monitor the Services that are deployed onto a service cloud it is necessary to embed probes into the environment of the applications that are executing inside each VEE. The data from these probes are sent from the VEE, via the infrastructure network, into the Service Manager. As an example of such a probe, we developed and installed one that collects the length of the job queue for a virtualized Sun Grid Engine application [29]. The queue length was used directly by one of the elasticity rules for this application, so that (a) when the queue length was high new virtual machines were allocated automatically, or (b) when queue length went low existing virtual machines were shutdown automatically. By adapting to the queue length the Service Manager and the VEEM optimized the running of the whole Sun Grid Engine application on RESERVOIR.

Summary of the Lattice Monitoring Framework

The Lattice framework provides a platform from which various monitoring systems can be built. It provides the building blocks and the glue from which specific monitoring elements can devised.

We can write probes which collect data for any specific purpose and then write consumers which process that data in any way necessary. Lattice itself does not provide a pre-defined library of probes, data sources, or consumers. Rather, it allows the probes, data sources, and consumers to be written and deployed as needed. For different applications, we can write a completely different monitoring system that is based on the Lattice monitoring framework. As an example of this, we have written a monitoring system for the virtual network project AutoI [30] for which we have written separate probes.

3. MONITORING FEDERATED SERVICE CLOUDS

When Service Clouds are federated to accept each other's workload there needs to be a consideration of how monitoring will behave in the presence of the federated VEEs. Monitoring needs to continue to work correctly and reliably when a service executes across federated clouds. When some VEEs are migrated to another cloud, the data distribution mechanism will need to cross cloud boundaries. It is essential that the interfaces and formats between clouds is standardised in order that federation monitoring should work in heterogeneous environments. This will ensure that that the monitoring data for all the VEEs of a service will be connected, whether local or remote.

Monitoring in a service cloud presents us with some interesting issues. Although the service cloud infrastructure is a distributed system, it is structured in a very particular way, with one large grid of machines acting as one cloud. Most of the monitoring data stays within the site, as all of the consumers are within the site. The exception is for federated VEEs. With many monitoring systems the sources and consumers are often distributed arbitrarily across a network, and so the paths of data flow and the patterns of interaction are different. Within a service cloud the pattern is more predictable, and so we need to design and build for this. In figure 2 we can see how the monitoring for the VEEs in one service are connected by the data distribution mechanism within each cloud, and how these are connected together to form a single service.

When a VEE is migrated from one site to a remote site, the monitoring data from that VEE still needs to be collected by the Service Manager. However, by migrating to a remote site the originating site loses direct control of the VEE. However, in order to ensure a continuous monitoring capability, some fundamental issues need to considered. They are:

- how does monitoring behave when a VEE is sent to a remote cloud
- what operations need to be done in each of the clouds to activate this federation
- what components and objects need to be created and/or managed when federation occurs.

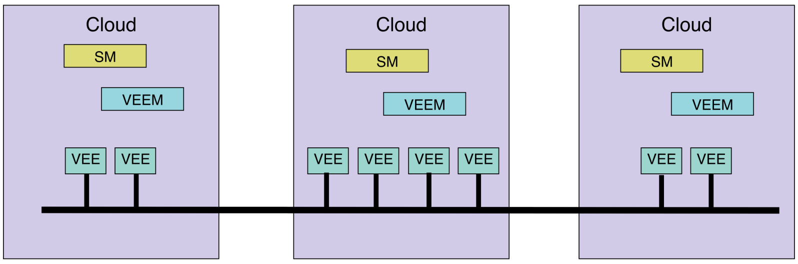

Figure 2. The Federation of VEEs for a Single Service

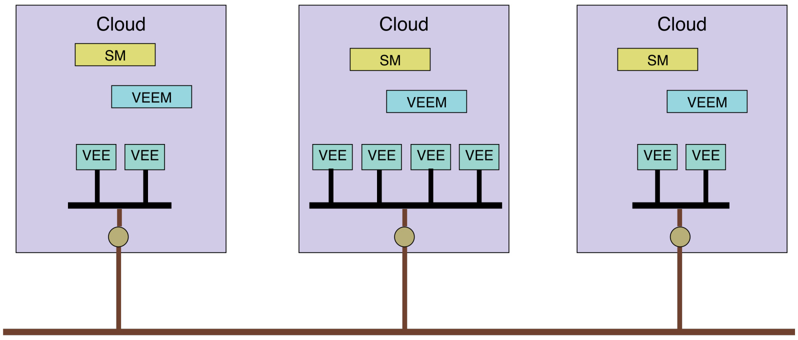

Figure 3. The Federation of VEEs on Different Clouds

As the remote VEEs have crossed another cloud's boundaries, there is not really a direct connection between the VEEs as seen in figure 2. Each cloud will be on its own network, hence the VEEs residing within it will not have direct contact with VEEs in another cloud. There is a need for a component in each cloud, such as a gateway or a broker, to act on behalf of the VEEs. Such a configuration is shown in figure 3, where all of the VEEs are still connected, but using this intermediate component.

For federated monitoring, there needs to be consideration all of the operations, components, and objects that may be involved in federation of VEEs. It is important to note that the data distribution mechanism crosses cloud boundaries as VEEs migrate. Although the data distribution mechanism for monitoring connects all of the VEEs, this cannot happen for free. There are many issues to consider. For this process to work correctly the system needs to:

- address the setup of the federation in the home and remote cloud
- address the setup of the remote migration
- address the connection for sending measurements back to the home cloud
- address the tear-down of remoting in both the the home and remote cloud
- ensure that remote and connected VEEs must be kept separate from other services.

For each of these steps we need to control the operations that need to be done in each of the clouds to activate this federation, and interact with the components and objects need to be created and/or managed when federation occurs.

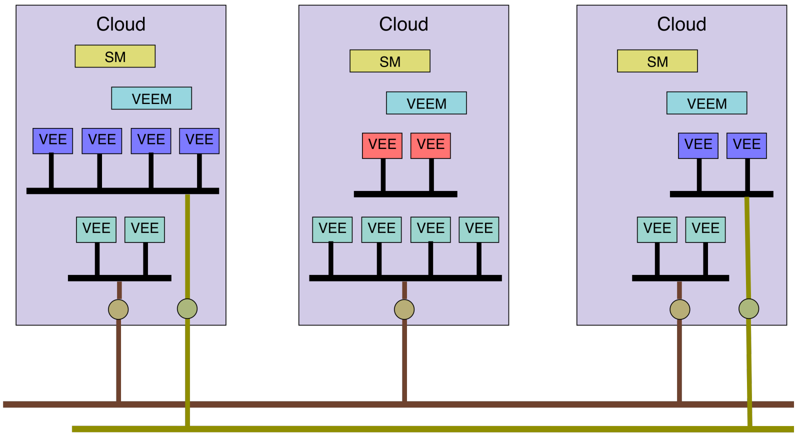

Figure 4. The Federation of VEEs for Multiple Services

Furthermore, as a service cloud will be used for more than one service, we need to also consider how we address the federation of multiple services. Each service needs to be kept separate and secure from all of the other services. As a consequence the process of splitting data distribution into multiple segments and having each segment connected using a component onto an external network will have to be done *per service*. Such a scenario is shown in figure 4 where there are multiple services across the thee clouds.

4. CONCLUSIONS

Monitoring in service clouds, such as RESERVOIR, presents us with some interesting issues. Although the service cloud infrastructure is a distributed system, it is structured in a very particular way, with one large cluster of machines acting as one cloud site. Most of the monitoring data stays within the site, as all of the consumers are within the site. The only exception to this is for federated VEEs. We see that the monitoring system is pervasive to a service cloud as:

1. it is required by most of the components of the service cloud;
2. it cuts across the layers of the service cloud creating vertical paths; and
3. it spans out across all the clouds in a federation in order to link all the VEEs of a service.

With many monitoring systems the sources and consumers are often distributed arbitrarily across a network, and so the paths of data flow and the patterns of interaction are different. Within a service cloud the pattern is more predictable, and so we have designed and built for this situation.

The use of the Lattice framework allowed us to build a monitoring infrastructure that collects, processes, and disseminates network and system information from/to the entities at real-time, acting as an enabler for service management functionality. We have seen that monitoring is a fundamental feature of any service cloud management system. We have also shown that Lattice can provide such monitoring for systems in which the virtualized resources are highly volatile as they can be started, stopped, or migrated at any time by the management.

The paper has presented the Lattice framework, which has been successfully used in the RESERVOIR project [5] for monitoring service cloud components. We can build many different monitoring systems using the framework, where different implementations of probes, data sources, and consumers are envisaged. Such an approach has been undertaken within the AutoI project [31] to allow the monitoring of virtual networks.

4.1. Further Work

There is much further work to do in the areas of service cloud management and monitoring.

Within RESERVOIR, there is a full monitoring system lifecycle integration within the Service Manager, from the initial Service Definition Manifest presentation, through the first service deployment, and then the continual execution of the service. To support fully dynamic on-the-fly functionality these lifecycle and control functions will need to use a control plane of the monitoring framework. Using this control plane the Service Manager will be able to activate, deactivate, and set the data rate of probes, as necessary.

Furthermore, to ensure the continuous operation of the monitoring system, it needs to be self-aware of its operation and its effect on the whole RESERVOIR system. In order to make it this way, the monitoring system needs its own adaptable controller with policies which can control the whole of the monitoring system. All of these elements are considered as part of the further work.

A further goal for the Lattice monitoring system is to have the ability to add new probes to a data source at run-time. By using this approach we will be able to instrument components of the service cloud without having to restart them in order to get new information.

Acknowledgment

This work is partially supported by the European Union through the RESERVOIR project [5] of the 7th Framework Program. We would like to thank the members of the RESERVOIR consortium for their helpful comments.

References

[1] "Future Internet Assembly (FIA)." http://www.future-internet.eu/.

[2] A. Galis, H. Abramowicz, M. Brunner, D. Raz, P. Chemouil, J. Butler, C. Polychronopoulos, S. Clayman, *et al.*, "Management and service-aware networking architectures for future internet position paper: System functions, capabilities and requirements," in *IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09)*, August 2009. Invited paper.

[3] "GENI design principles," *Computer*, vol. 39, pp. 102–105, 2006.

[4] "Future Internet Design (FIND) program." http://www.nets-find.net/.

[5] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, *et al.*, "The RESERVOIR model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, 2009.

[6] N. Carr, *The Big Switch - Rewiring the World from Edisson to Google*. W. W. Norton, January 2008.

[7] "Open cirrus - open cloud-computing testbed." www.opencirrus.org.

[8] P. Wallis, "Understanding cloud computing, keystones and rivets." http://www.keystonesandrivets.com/kar/2008/02/cloud-computing.html, February 2008.

[9] "Above the clouds: A berkeley view of cloud computing." http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[10] "The Amazon Elastic Compute Cloud (Amazon EC2) web site." http://www.amazon.com /gp/browse.html?node=201590011.

[11] "What is the Google App Engine." http://code.google.com/appengine/docs/whatisgoogleappengine.html.

[12] B. Rochwerger, D. Breitgand, D. Hadas, I. Llorente, R. Montero, P. Massonet, E. Levy, A. Galis, M. Villari, Y. Wolfsthal, E. Elmroth, J. Caceres, C. Vazquez, and J. Tordsson, "An architecture for federated cloud computing," *Cloud Computing*, 2010.

[13] R. M. Fano, "The MAC system: The computer utility approach," *IEEE Spectrum*, vol. 2, p. 5644, January 1965.

[14] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International J. Supercomputer Applications*, vol. 15, no. 3, 2001.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "Grid services for distributed system integration," *Computer*, vol. 35, no. 6, 2002.

[16] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. V. Reich, "The Open Grid Services Architecture, Version 1.0," tech. rep., Open Grid Forum (OGF), January 2005.

[17] B. Rochwerger, A. Galis, D. Breitgand, E. Levy, J. A. Caceres, I. M. Llorente, Y. Wolfsthal, M. Wusthoff, S. Clayman, W. Emmerich, E. Elmroth, and R. S. Montero, "Design for future internet service infrastructures," in *Towards the Future Internet - A European Research Perspective*, p. 350, IOS Press, April 2009.

[18] B. Rochwerger, A. Galis, E. Levy, J. A. Caceres, D. Breitgand, Y. Wolfsthal, I. M. Llorente, M. Wusthoff, R. S. Montero, and E. Elmroth, "Management technologies and requirements for next generation service oriented infrastructures," in *11th IFIP/IEEE International Symposium on Integrated Management*, June 2009.

[19] J. Skene, D. Lamanna, and W. Emmerich, "Precise Service Level Agreements," in *Proc. of the* 26^{th} *Int. Conference on Software Engineering, Edinburgh, UK*, pp. 179–188, IEEE Computer Society Press, May 2004.

[20] A. Sahai, S. Graupner, *et al.*, "Specifying and monitoring guarantees in commercial grids through sla," in *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'03)*, pp. 292–299, 2002.

[21] P. Barham, B. Dragovic, K. Fraser, S. Hand, *et al.*, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.

[22] A. Kivity, "kvm: the linux virtual machine monitor," in *OLS '07: The 2007 Ottawa Linux Symposium*, pp. 225–230, July 2007.

[23] J. Skene, A. Skene, J. Crampton, and W. Emmerich, "The Monitorability of Service-Level Agreements for Application-Service Provision," in *Proc. of the* 6^{th} *Int. Workshop on Software and Performance (WOSP), Buenos Aires, Argentina*, pp. 3–14, ACM Press, Feb. 2007.

[24] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: Design, implementation and experience," *Parallel Computing*, vol. 30, p. 2004, 2003.

[25] "Nagios." http://www.nagios.org/.

[26] H. Newman, I. Legrand, P. Galvez, R. Voicu, and C. Cirstoiu, "MonALISA : A distributed monitoring service architecture," in *Proceedings of CHEP03, La Jolla, California*, 2003.

[27] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, G. Tortone, and M. C. Vistoli, "GridICE: A monitoring service for grid systems," *Future Gener. Comput. Syst.*, vol. 21, no. 4, pp. 559–571, 2005.

[28] A. Cooke, A. J. G. Gray, L. Ma, W. Nutt, *et al.*, "R-GMA: An information integration system for grid monitoring," in *Proceedings of the 11th International Conference on Cooperative Information Systems*, pp. 462–481, 2003.

[29] "Guide to Sun Grid Engine 6.2 Installation and Configuration," white paper, Sun Microsystems, September 2008.

[30] A. Galis, S. Denazis, A. Bassi, P. Giacomin, *et al.*, *Management Architecture and Systems for Future Internet Networks*. IOS Press, http://www.iospress.nl, ISBN 978-1-60750-007-0, April 2009.

[31] "Autonomic Internet (AutoI) Project." http://www.ist-autoi.eu/, 2008-2010.