ELEKTROTEHNIŠKI VESTNIK 81(3): 94–100, 2014
ORIGINAL SCIENTIFIC PAPER

Multiple-cloud platform monitoring

Jernej Vičič[1,3], **Andrej Brodnik**[1,2]

[1]*University of Primorska, Andrej Marušič Institute, Muzejski trg 2, 6000 Koper, Slovenia*
[2]*University of Ljubljana, Faculty of Computer and Information Sciences, Tržaška cesta 25, 1000 Ljubljana, Slovenia*
[3]*Fran Ramovš Institute of the Slovenian Language, SRC SASA, Novi trg 4, 1000 Ljubljana, Slovenia*

E-mail: jernej.vicic@upr.si, andrej.brodnik@upr.si

Abstract. A research and a pilot implementation of a monitoring architecture for multiple-cloud infrastructures of VMware, HyperV and OpenStack are presented. A standardized set of monitoring attributes is selected and an efficient architecture to support monitoring itself is implemented.
Two different monitoring architectures and their interfaces are described. The pilot is implemented by using the less flexible architecture due to the lack of time. The advantages and disadvantages of both architectures are analyzed.
Furthermore two monitoring systems were implemented. One of them serves only as a proof of concept and the other is aa complete monitoring infrastructure for multiple different clouds. On the top of the monitoring system, a fully operational support for SLAs was implemented.

Keywords: multiple cloud, monitoring, cloud

Nadzor več oblačnih postavitev

Predstavljamo raziskavo in pilotsko postavitev arhitekture sistema za nadzor več oblačnih infrastruktur: VMware, HyperV in OpenStack. Raziskava se osredinja na izbiro standardizirane množice atributov za spremljanje in na izdelavo učinkovite arhitekture za sam nadzor. V prispevku razpravljamo o dveh arhitekturah s potrebnimi vmesniki. Pri izvedbi pilotne postavitve je bila uporabljena manj prilagodljiva arhitektura zaradi časovne omejitve, predstavljene pa so prednosti in slabosti obeh arhitektur. Poleg tega smo izdelali dva sistema za nadzor, kjer je bil namen prvega le dokaz koncepta, drugi pa implementira sistem za nadzor večih oblačnih infrastruktur. Poleg tega sistem implementira polno operativno podporo za SLA.

1 INTRODUCTION

As cloud management is some kind of specialization of management of distributed computing systems, it inherits many techniques from the traditional computer network-management. However, as cloud computing environments are considerably more complex than those of legacy-distributed computing [1], new management methods and tools need to be implemented. Introduction of multiple-cloud platforms (such as VMware, HyperV OpenStack and others) and monitoring crucial aspects from a centralized point are a challenging task. The multiple-cloud monitoring introduces the problem of maintaining compatibility between different attributes in different clouds. Furthermore, not only measures, but also APIs of different clouds are quite different. Indeed, the number of interfaces can be as high as the number of the cloud platforms.

The easiest way to define the quality of service at the SLA (Service Level Agreement [2]) level is to measure the availability of services and quality of their deliverance. It is very difficult to define and measure the exact availability of the resources, such as processing power, I/O read/write speed, etc. The provider provides only the nominal values of availability for such resources. Quality of service (QoS), which defines the availability of sufficient resources to the user, is crucial to the users of cloud services.

There have been some attempts to address the particulars of private and mixed clouds, in particular the solution [3] which presents an implementation of a private cloud monitoring system (PCMONS). The solution is based on agents. The [4] presents a framework called Lattice for monitoring multiple federated private clouds. The general requirements for cloud monitoring are defined in [5], although the multi-tenancy requirement has not much relevance in a private cloud. The current portfolio of the available tools lacks open source, inter-operable cloud management and monitoring tools. To the best of the authors' knowledge, no research has been made public that addresses the specific problems of monitoring multiple private and public clouds.

The paper is structured as follows: Section 2 reviews the related work; it mainly addresses similarities with the basic network management and monitoring. Section 3 presents the methodology, followed by a description

Received 16 December 2013
Accepted 9 May 2014

of implementation and testing in section 4. The final section presents the conclusions and plans for the future work.

2 RELATED WORK

2.1 Network management and monitoring

Network monitoring is defined in [6] and [7] as the use of a system that constantly monitors a computer network for slow or failing components. The system provides several ways of notification for extraordinary events. It is a subset of the functions involved in network management, missing the functions for manipulating network components. Commonly measured metrics are response time, availability, uptime, consistency and reliability metrics.

2.1.1 Open-source monitoring tools: A survey was made by comparing the open-source and free network and system monitoring tools. Some of the observed studies are [8] and [9]. The tools having the biggest user base, biggest number of active installations and the best overall reputation in the community were selected:

- Ganglia monitoring system [10],
- Nagios Core [11] and
- Zabbix [12].

Cacti ([13]) is an open-source, web-based network monitoring and graphing tool designed as a front-end application for the open-source tool. It can be extended using plugins to monitor virtually anything, which makes it a viable solution for a base for a new monitoring tool.

2.1.2 Architecture: Fig. 1 shows the basic architecture of a network management system as presented in [6] and [7]. The architecture for the network monitoring system is the same, only the functionality is limited to monitoring, The basic building blocks are:

- Network Management System (NMS);
- Simple Network Management Protocol (SNMP) [14] presenting a means of standardized communication between the monitoring entities;
- Management Information Base (MIB): monitoring information storage;
- Remote MONitoring (RMON) [15].

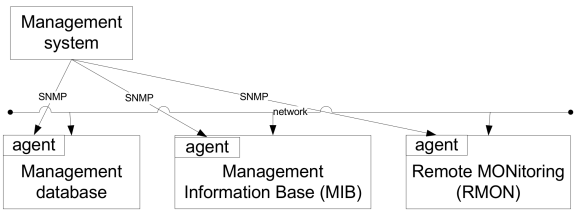

Figure 1. Basic architecture of a network monitoring system.

2.2 Cloud monitoring

The cloud monitoring tools monitor performance of the applications hosted in the cloud or control the public cloud infrastructure, primarily observing the SLA agreement (SLA, [2]). These tools primarily target one cloud platform.

Each cloud deployment model has particular needs and requires different approaches. The most distinct differences are between public and private clouds. The public clouds often have geographically diffuse, large resource pools, which require more investment in monitoring the traffic and ensuring scalability. Service metrics, such as the link availability and connection speed, are an essential information in both the private and public clouds and are subject to the SLA agreement.

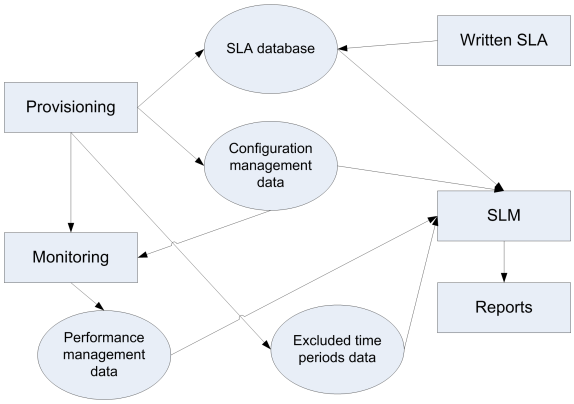

Figure 2. Basic architecture of a cloud system.

The user should be able to control the services remotely. It is important that a ”base-lining” (comparing the current performance to a historical metric or baseline) is created, which represents the optimum. The multiple clouds introduce a new dimension into solving the problem of the monitoring task. The control and monitoring task can be distinguished as two separate tasks:

- External control: all the services are accessible from the outside, including the metrics offered by the service provider. It is necessary to consider also the impact of the communication link from the user to the service provider;
- Internal control: parameters are usually not accessible to the users. The parameters must be controlled by the service provider to detect on the basis of this monitoring the errors and anticipate the trends (accompanied by SLA).

Speaking from the user perspective, the control may also vary according to the type of the cloud (public, private, and hybrid). This work is limited to controlling the IaaS services which include:

- Network control: availability, packet loss, packet

delay and jitter (usually a link between the IaaS platform and the Internet);
- Monitoring the CPU server;
- Storage monitoring, e.g. the API interfaces via the cloud;
- Other types of monitoring: detailed server monitoring and
- Application monitoring.

The interfaces usually used to control the server interface are standard interfaces supported by the operating systems SNMP, WMI, wbem and virtual platform interface, or by a dedicated agent installed on the server that communicates with the control system via a standard or, in most cases, proprietary protocol. Fig. 2 shows the basic control and management architecture of the cloud services.

Int the first phases of the deployment the user or supplier follows the agreed SLA and includes the control services. The control system generates SLA reports (via SLM - Service Level Management module) from the monitoring and SLA data. The SLM module monitors the SLA in real time using an appropriate "dashboard" interface and automatic generation of the SLA reports according to the agreement and compliance penalties.

2.2.1 Accessing the cloud properties: Most of the service providers in the cloud offer their programmable interfaces for monitoring and accessing the functionality. They are usually well documented and publicly accessible (often under a Creative Commons license). As these interfaces differ substantially between individual service providers, they are not inter-operable. Some vendors have adopted interfaces provided by the well established players like Amazon EC2 API [16]. There is also a number of open standards under development, including the Open-Cloud Computing Interface – OCCI consortium and Open-Grid Forum – OGF, which bring together some of the most important players in the field of the cloud technologies. Open standards support independence from services of one manufacturer. The OCCI protocol [17] is a collection of the needs of the IaaS Cloud computing managers and administrators in the form of Use Cases. This document presents the grounds for the monitoring requirements in which the reference feature set (described in Section 3.1) is collected. Configuration Description, Deployment and Lifecycle Management (CDDLM) [18] describes a standard for the management, deployment and configuration life cycle of the Grid services.

3 METHODOLOGY

Our research experiment focused on the following goals:

- definition of a suitable architecture;
- selection of a reference feature set to be monitored for each platform;
- implementation of a pilot monitoring system and deployment into a simulated working environment.

The test multiple-cloud setting consisted of three different cloud infrastructures: VMware, HyperV and OpenStack. Each platform used a virtualization platform: VMware ESX, Microsoft Hyper-V and Linux KVM, respectively. For the monitoring system to function properly, accessing the information about the hosts and virtual machines is needed. The information is delivered through standardized interfaces (installed probes, API calls to the platforms). The selection of the platform set was motivated by the following criterion: select two from the closed-source infrastructures with the largest user base and the most used open-source infrastructure. The selection was agreed by all partners of the KC Class project consortium. Further information about the project is available in Section 6.

3.1 Monitoring-parameters comparison

Three private cloud platforms were selected as the testing environments. Two were closed source, VMware and HyperV and one was open source, OpenStack.

For each of the studied cloud platforms a reference feature set was constructed using a list of the most popular and most important monitoring features. A similar research was already made by [5] and [3], but the results were not conclusive. The most important document our research eas based on, was the OCCI protocol [17] with the record of Use cases. The reference-feature set was constructed according to the research results [4], [19] and [20]. It is presented in Table 1.

Each platform completely supported the reference feature set except for the low-level feature set for the OpenStack (host parameters, marked with an $*$ in Table 1). Therefore, we introduced the Ganglia monitoring system [21] for the OpenStack cloud to support the missing monitoring properties. The selection process is described in Section 3.2.

3.2 Host parameters monitoring

The three final candidates presented in Section 2.1.1 were deployed in a real-life simulated environment to test the ease of installation, scalability and usability (feature set). Based on the testing results the Ganglia monitoring system was selected although Nagios has the largest installation base (most used). The reason for selecting Ganglia was its ease of installation, simple communication protocol and minimalistic design properties. These properties make Ganglia to be most suitable for cluster monitoring which is basically what the system will be used for in our setting. Moreover, Ganglia has also won a good reputation in the community and a big user installation base.

Table 1. The reference feature set. The $*$ denotes the parameters monitored with the third party software.

Parameter/infrastructure	VMware	OStack	Hyper-V
Mixed parameters			
Virtual machine name	X	X	X
Virtual machine status	X	X	X
Host OS parameters			
Last host OS start	X	X	X
Host OS drivers status	X	X	X
Assigned proc. time	X	X	X
Processor load	X	X	X
Processor time limit	X	X	X
Processor time reservation	X	X	X
Memory limit	X	X	X
Memory reservation	X	X	X
Memory usage	X	X	X
Management mem usage	X	X	X
Swap size	X	X	X
Network interface name	X	X	X
Network interface status	X	X	X
IP address	X	X	X
Forwarded packs	X	X	X
Received packs	X	X	X
Sent packs	X	X	X
The assigned disk name	X	X	X
Read speed, nr. of reads	X	X	X
Write speed, nr. of writes	X	X	X
VM disk name	X	X	X
Used disk space	X	X	X
Non-used disk space	X	X	X
Host information			
Server name	X	X$*$	X
Manufacturer name	X	X$*$	X
Proc. model name and nr.	X	X$*$	X
Number of cores	X	X$*$	X
Core speed	X	X$*$	X
Memory size	X	X$*$	X
Actively used memory size	X	X$*$	X
Management memory size	X	X$*$	X
Kernel memory size	X	X$*$	X
Last boot time	X	X$*$	X
Nr. of virtual machines	X	X$*$	X
Nr. of networks	X	X$*$	X
Nr. of network adapters	X	X$*$	X
Amount of all data	X	X$*$	X
Amount of read data	X	X$*$	X
Amount of sent data	X	X$*$	X
Amount of received data	X	X$*$	X
Storage data			
Storage capacity	X	X	X
Available storage capacity	X	X	X
File system type	X	X	X
Image data			
Product name	X	X	X
Product information	X	X	X
Product URL	X	X	X
Manufacturer URL	X	X	X
Image status	X	X	X

3.3 Design overview

The basic design of the cloud architecture is presented in Fig. 2. Monitoring is presented as an independent module. The multiple cloud monitoring architecture complies with the established architecture. Each control system collects data and makes them available to the SLA control system and to the control dashboard. The data is processed either in real-time or on-demand. An important performance aspect are the architecture of the control systems and the interface.

Two architectures are possible. The first is shown on Fig. 3 in which the control system communicates directly with the interface available for a virtual platform. Examples of such interfaces are the VMware SDK [22]; Hyper-V WMI [23], etc. The access to these interfaces is possible via an additional software installed on the control system. In control-system modules can also be implemented as plug-ins. The advantages of such implementation are easier debugging and better error control.

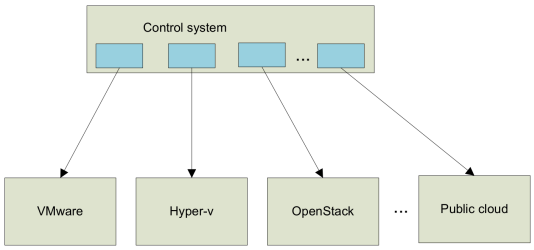

Figure 3. First possibility: the architecture design with direct communication.

In the second possible architecture (see Fig. 4) in which the control system communicates via the same protocol with different virtual platforms, there is no need to install special software on each virtual platform. A gateway (translation interface) is implemented for each virtual platform between the interfaces used by the control system and the virtual platform. The advantage of this implementation is that the controlled systems are presented in the same way which makes it easy to scale the monitoring system. A similar approach is known in the network management (protocols SNMP[14], RMON[15], and in the Ganglia monitoring system [21]).

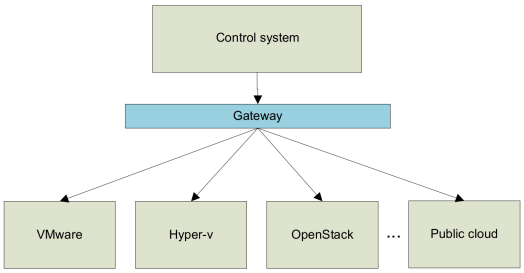

Figure 4. The second possibility: the architecture design with communication through a gateway.

The first possibility was selected for pragmatic reasons because the lack of time and resources to implement the second possibility which is more versatile but was more demanding second architecture, although it presents an important advantage over the first one.

4 IMPLEMENTATION AND VALIDATION

The presented design was implemented on a fully functional multiple-cloud setting. Its details are shown in Fig. 5. The cloud consists of three separate platform implementations: OpenStack, Microsoft Hyper-V, VMware.

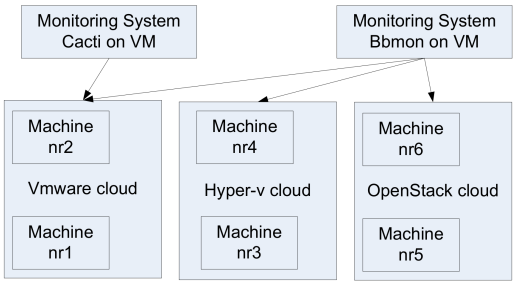

Figure 5. Implemented testi setting.

The proposed architecture and the selected monitoring parameter set were implemented in a fully functional monitoring system, BBMon, that permiting empirical testing of the simulated and real-life test cases. Fig. 6 shows the implemented architecture and the existing control systems. The architecture supports the complete SLA control. Each platform provides a system for monitoring the virtual systems, including ”provisioning” and control. Monitoring can be done through the API interfaces allowing control to the third-party tools. All the data for the OpenStack host parameters gathered by Ganglia are parsed and presented by the BBMon tool.

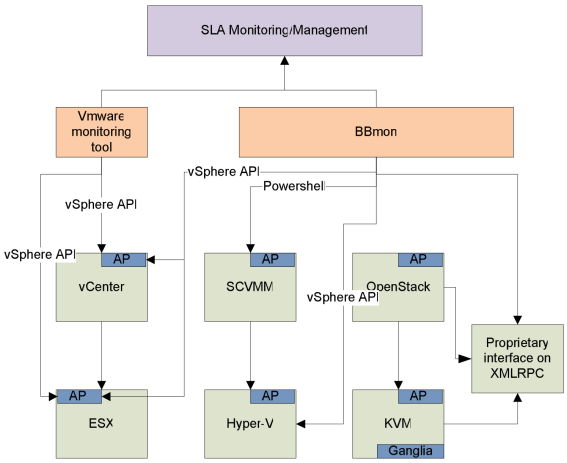

Figure 6. Architecture of the proposed system: two monitoring systems, VMware monitoring tool and BBmon, are deployed on a mixed-cloud environment.

Two control systems were implemented: the VMware monitoring tool and BBmon. The first is used as a proof-of-concept and monitors only the VMware cloud, while the later interacts with the three systems implementing the architecture shown in Fig. 6. ESX (Elastic Sky X) is a hypervisor for the VMware platform, Hyper-v is a hypervisor for the Microsoft platform and KVM is a hypervizor for the OpenStack platform. VSpehere APi is VMware API adopted also by Hyper-v platform and SCVMM (System Center Virtual Machine Manager) is a virtual machine manager tool for the Microsoft solution. XML-RPC is a remote procedure call (RPC) protocol.

4.1 VMware monitoring tool

A single cloud monitoring (VMware Virtualization Monitoring) tool based on the open-source toolkit Cacti was developed as a proof-of-concept. A graphical interface is used to display graphs. The data storage is of a long-term type. A plugin communicating with the virtualization infrastructure through VMware API was introduced to the Cacti, thus enabling monitoring of VMware environment virtualization infrastructure. API consists of a non-trivial set of data structures enabling creation, editing, monitoring and management of most of the infrastructure components. The Cacti framework was used to avoid developing the basic control systems functionality. The tool presents the basic set of functionalities for periodic capturing of the necessary information using API.

4.2 BBmon monitoring system

The BBmon control system is a set of open-source tools and accessories produced by Astec*. One of the most attracting features of the system is the display of the control tests. It supports control of the basic server characteristics presented in Table 1.

Fig. 7 shows the monitoring console of the BBmon control system.

Control can be done in two ways:

- through agents installed on the servers;
- without agents through different protocols or interfaces supported by the controlled appliances or applications.

The system calculates availability of the resources within a specified time period. The calculation is called a test. The test can be modified to tailor specific client’s needs and cover all available monitoring resources with the suitable time-frames. An additional module, called the SLA monitor producing the SLA reports and alarms compliant to the SLA agreement.

4.3 Testing / Validation

The implemented monitoring systems, i.e. VMware monitoring tool and BBmon, were used in a real-life simulation setting.

Stress tests based on artificial traffic and request generators were used to assess the general robustness

*http://www.astec.si

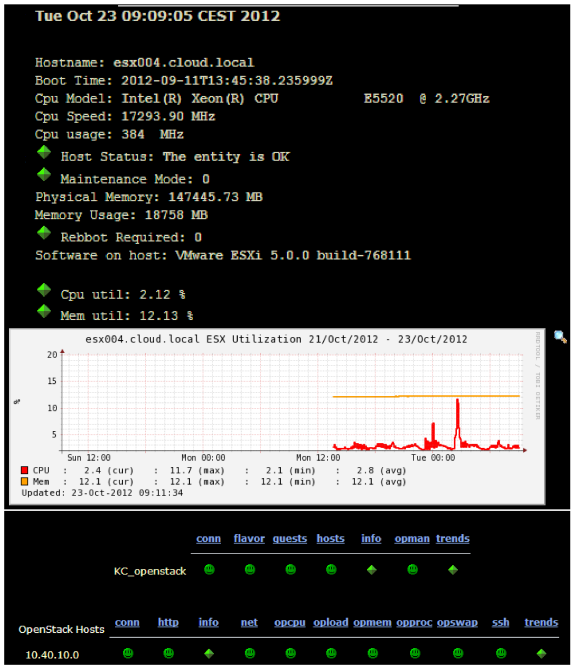

Figure 7. Metrics grouped into the BBmon tests (different states).

of the presented architectures and also of the specific implementation:

- A set of three Linux flavors (a generic off-the-shelf Ubuntu server, Debian server, CentOS server) and one windows flavor (a Windows Server 2008) were prepared. A flavor is a cloud-ready installation of an operating system used to produce ”live” images;
- Each platform was loaded with 10 images, 6 Linux and 4 Windows flavors. The total number of images was 30;
- A script was prepared to start and stop the installed images as planned:
  - at the first hour, all the images were started;
  - at the next hour, 50% of the (randomly selected) images were shut down;
  - at the last hour, all the images were shut down;
  - the iteration was repeated for the whole week;
- The setting (all images) was subjected to a heavy load for a random amount of time: a set of traffic generators generated requests to all the live images.

The testing results are presented in Table 2.

The monitoring systems performed normally: the latency of the monitoring functions was not increased or only marginally increased in the heavy-load state. No data was lost. Some minor flaws in the functionality were fixed ad-hoc, but overall the proposed architecture design shows no flaws. The monitoring systems were installed on VMs in the cloud with the possibility of having the setting changed with no impact to the generality.

Table 2. The stress-test results: no data lost, latency only neglectably increased.

Started images	Locally collected data (bytes)	Received data (bytes)	Latency
Normal load			
100 %	126.290	126.290	0m0.568s
50 %	74.670	74.670	0m0.574s
0 %	23.050	23.050	0m0.559s
Stress load			
100 %	126.290	126.290	0m0.642s
50 %	74.670	74.670	0m0.664s
0 %	23.050	23.050	0m0.634s

5 Conclusion and Discussion

A reference-feature set for cloud monitoring to be used as a basis for new comparisons is constructed following the results of previous research. The selected platforms monitoring features are checked against a reference feature set; each of the three platforms enables the reference feature set to be monitored. A pilot implementation of the monitoring system proves that the prepared solution is satisfactory but needing further evaluation.

The future plans include reimplementation of a standardized architecture solution with the monitoring system communicating with the virtual platforms through a gateway as shown in Fig. 4. The presented methods and architectures should be further evaluated on a large mixed-cloud environment using the presented criteria and the evaluation method should be further researched as the latency and data consistency of the monitoring system do not represent the whole monitoring system performance.

6 Acknowledgements

We want to thank Janez Bostič and Borut Žnidar from ASTEC d.o.o and Andraž Piletič from NIL d.o.o who greatly participated in the implementation of the presented work. The work presented in this paper was supported by the European Union, European Regional Fund, within the scope of the framework of the Operational Programme for Strengthening Regional Development Potentials for the Period 2007-2013, contracts No. 3211-10-000467 (KC Class).

References

[1] V. H. Nguyen, D. Tran, I. Les, M. Cedex, D. N. Inria, and K. Nantes, “Autonomic virtual resource management for service hosting platforms ∗,” in *CLOUD ’09 Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009, pp. 1–8.

[2] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 27–33, 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm ?arnumber= 5474674

[3] C. B. W. Shirlei Aparecida de Chaves, Rafael Brundo Uriarte, "Toward an Architecture for Monitoring Private Clouds," *IEEE Communications Magazine*, vol. 49, no. December, pp. 130–137, 2011.

[4] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-merino, L. M. Vaquero, K. Nagin, and B. Rochwerger, "Monitoring Service Clouds in the Future Internet," in *Towards the Future Internet - Emerging Trends from European Research*. IOS Press, 2010, pp. 115–126.

[5] P. Hasselmeyer and N. D'Heureuse, "Towards holistic multi-tenant monitoring for virtual data centers," in *2010 IEEE/IFIP Network Operations and Management Symposium Workshops*. IEEE, 2010, pp. 350–356. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp ?arnumber= 5486528&contentType=Conference+Publications

[6] A. Clemm, *Network Management Fundamentals*. Cisco Press, 2006. [Online]. Available: http://www.ciscopress.com/store/network-management-fundamentals-9781587201370

[7] Cisco, "Network Management System: Best Practices White Paper," 2007.

[8] S. Zanikolas and R. Sakellariou, "A taxonomy of grid monitoring systems," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 163–188, 2005.

[9] TopTenReviews, "Monitoring software review," TopTenReviews, Tech. Rep., 2012. [Online]. Available: http://monitoring-software-review.toptenreviews.com/

[10] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, Jul. 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167819104000535

[11] "Nagios Core", "Nagios Core Version 3.x Documentation," Nagios Core Development Team and Community, Tech. Rep., 2010. [Online]. Available: httpd://nagios.sourceforge.net/docs/nagios-3.pdf

[12] A. Vladishev, C. Collins, and S. Marriott, "ZABBIX Manual," Zabbix SIA, Tech. Rep., 2008. [Online]. Available: http://www.docstoc.com/docs/ 97829643/ZABBIX-Reference-Manual

[13] I. Berry, T. Roman, L. Adams, J. P. Pasnak, J. Conner, R. Scheck, and A. Braun, "The Cacti Manual," The Cacti Group, Tech. Rep., 2012. [Online]. Available: http://www.cacti.net/downloads/docs/text/manual.txt

[14] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition)*. Addison-Wesley Professional, 1999. [Online]. Available: http://www.amazon.com/SNMP-SNMPv2-SNMPv3-RMON-Edition/dp/0201485346

[15] M. Erlinger, "RMON - From Concept to Specification," in *IFIP TC6/WG6.6*, Apr. 1993, pp. 73–80. [Online]. Available: http://dl.acm.org/citation.cfm?id=647599.732059

[16] Amazon, "API Reference Amazon Elastic Compute Cloud: API Reference," Amazon, Tech. Rep., 2012.

[17] T. Metsch, "Open Cloud Computing Interface - Use cases and requirements for a Cloud API," Sun Mictosystems, Tech. Rep., 2009.

[18] P. Goldsack, "Configuration Description, Deployment, and Lifecycle Management SmartFrog-Based Language Specification," Hewlett Packard Labs, Tech. Rep., 2005.

[19] CloudTweaks, "Top 10 Cloud Computing Load Test and Performance Monitoring Companies," cloudtweaks.com, Tech. Rep., 2012. [Online]. Available: cloudtweaks.com

[20] T. Harris, "Cloud Computing Services – A comparison," THBS, Tech. Rep., 2012. [Online]. Available: http://www.thbs.com/pdfs/Comparison of Cloud computing services.pdf

[21] Ganglia, "Ganglia Monitoring System," 2012.

[22] VmWare, "Virtual Disk API Programming Guide," vmWare, Tech. Rep., 2012.

[23] "Hyper-V WMI Provider", "Hyper-V WMI Provider," p. 34, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh850074(v=vs.85).aspx

Jernej Vičič received his B.Sc., M.Sc. and Ph.D. degrees in Computer Science from the University of Ljubljana, Slovenia, in 1999, 2003 and 2012, respectively. He is an assistant professor and researcher at the University of Primorska, Slovenia, and a researcher at the Fran Ramovš Institute of Slovenian Language SRC SASA, Slovenia.

Andrej Brodnik got his PhD degree in 1995 from the University of Waterloo, Ontario, Canada. After graduation he worked as head of research and CTO in industry (IskraSistemi and ActiveTools). In 2002 he joined the University of Primorska and also worked as a researcher and adjoined professor at the University of Technology in Luleå, Sweden. He has written authored several tens of various scientific papers and is an author and co-author of patents in Sweden and USA. The CiteSeer and ACM Digital Library lists over 200 citations of his works. He is also a recipient of a number of awards (National award for exceptional achievements in higher-education teaching; Boris Kidriča award; Fulbright scholarship; IBM Faculty Award, 2004; Golden Plaque of the University of Primorska; etc.). Currently he holds a teaching position at the University of Primorska and the University of Ljubljana.