Toward an Architecture for Monitoring Private Clouds

Shirlei Aparecida de Chaves, Rafael Brundo Uriarte, Carlos Becker Westphall
Post Graduation Program in Computer Science (PPGCC — UFSC)
Federal University of Santa Catarina
Florianópolis, Santa Catarina, Brazil

ABSTRACT

Cloud computing is rapidly emerging as a new model for service delivery, including for telecommunications services (cloud telephony). Although many solutions are now available, cloud management and monitoring technology has not kept pace, partially because of the lack of open source solutions. To address this limitation, this article describes our experience with a private cloud, and discusses the design and implementation of a private cloud monitoring system (PCMONS) and its application via a case study for the proposed architecture. An important finding of this article is that is possible to deploy a private cloud within the organization using only open source solutions and integrating with traditional tools like Nagios. However, there is significant development work to be done while integrating these tools. With PCMONS we took the first steps toward this goal, opening paths for new development opportunities as well as making PCMONS itself an open-source tool.

INTRODUCTION

Cloud computing has rapidly emerged as a method for service delivery over TCP/IP networks such as the Internet. It disrupts the traditional IT computing environment by providing organizations with an option to outsource the hosting and operations of their mission-critical business applications. Subscribers typically access cloud services using desktop-hosted web browsers as clients or, in the case of cloud telephony, VoIP hardphones or softphones. In addition to financial benefits such as minimal capital and operational expenditure (CAPEX and OPEX) costs, the cloud model provides several technical benefits including flexible hardware and software allocation, elasticity. and performance isolation.

Providing cloud computing services, however, may require sophisticated management procedures to ensure performance, robustness, dependability, and security. In particular, cloud telephony providers should closely manage their services. Because of the real-time low-latency nature of telephony, users may consider slightly degraded performance to be unusable. Cloud management may be viewed as a specialization of distributed computing management and therefore inherits many techniques from traditional computer network management. However, as cloud computing environments are considerably more complex than those of legacy distributed computing [1], they demand the development of new management methods and tools. To date, there are no widely accepted standards or open source reference implementations for cloud management applications. Progress in this area may have been hindered by early cloud computing implementations' dependence on particular technological solutions and tight coupling to specific cloud infrastructures.

Accordingly, this article proposes a generic cloud monitoring architecture for private clouds. The intent of this article is to:

- Provide insight into how traditional tools and methods for managing network and distributed systems can be reused in cloud computing management
- Introduce a private cloud monitoring system (PCMONS) we developed to validate this architecture, which we intend to open source
- Help future adopters of cloud computing make good decisions on building their monitoring system in the cloud

To that end, we describe our development and implementation, including some lessons learned during the process.

We argue that because of significant differences, there is no generic cloud management solution that may be applied to cover the three major cloud service models: infrastructure-as-a-service (IaaS), software-as-a-service (SaaS), and platform-as-a-service (PaaS). We chose to address management for IaaS because of its flexibility, as one can emulate SaaS and PaaS models over an IaaS base. We chose to address private clouds because they enable enterprises to reap cloud benefits while keeping their mission-critical data and software under their control and under the governance of their security policies. Among the services we expect private

0163-6804/11/$25.00 © 2011 IEEE

clouds to host is enterprise telephony. Solutions from leading enterprise telephony vendors and open source solutions such as Asterisk are now being offered in virtualized environments that may readily be mapped to a cloud environment.

This article is organized as follows. We provide background on cloud computing. We present the PCMONS architecture. We then describe the testbed environment and present our case study and results. We discuss related work. We expose lessons learned while developing the system. We encourage future potential users of open source PCMONS to study this section closely.

Background

The term *cloud computing* was created as a metaphor for the Internet as it is often depicted in network diagrams: as a cloud, abstracting the infrastructure and complexity it encompasses. "In essence, cloud computing is a construct that allows you to access applications that actually reside at a location other than your computer or other Internet-connected device; most often, this will be a distant datacenter [2]."

Cloud Computing Service Models

Regarding service models, there are three major ones in common use, presented below according to NIST definitions [3]:

- Software-as-a-service (SaaS): The consumer uses the provider's applications, which are hosted in the cloud.
- Platform-as-a-service (PaaS): Consumers deploy their own applications (home-grown or acquired) into the cloud infrastructure. Programming languages and application development tools used must be supported by the provider.
- Infrastructure-as-a-service (IaaS): Consumers are able to provision storage, network, processing, and other resources, and deploy and operate arbitrary software, ranging from applications to operating systems. This article focuses on this model.

For each service model, consumers have different degrees of control over the infrastructure management. In the SaaS model, control is typically limited to user-specific application configuration settings. PaaS gives control over the deployed applications, and possibly application hosting environment configurations. IaaS provides control over operating systems, storage, deployed applications, and possibly limited control of select networking components [3].

Cloud Computing Deployment Models

Another relevant concept is the cloud deployment models. The most well known are the following four, but it is important to note that there are other models that derive from these.

Public: Resources are available to the general public over the Internet. In this case, "public" characterizes the scope of interface accessibility, not whether or not resource usage is charged. This environment emphasizes the benefits of scalability, rationalization (since one can use only the required resources and pay just for their use "by the drink"), and operational simplicity (since the environment is hosted by a third party, i.e., the cloud provider). The main issue is security, since the environment is shared and managed by the cloud provider, and, accordingly, the consumer/subscriber has little control over it.

Private: Resources are accessible within a private organization. This environment emphasizes the benefits of scalability, integration, and optimization of hardware investments. The main issue is operational complexity, since the environment is hosted and managed by internal resources. Security is not a major concern when compared to a public cloud as the resources are accessible only through internal interfaces and are interconnected by private networks protected by network firewalls.

Community: Resources on this model are shared by several organizations with a common mission. It may be managed by one of the organizations or a third party [3].

Hybrid: This model mixes the techniques from public and private clouds. A private cloud can have its local infrastructure supplemented by computer capacity from a public cloud [4]. The benefits and challenges are a combination of the items above.

Regarding monitoring, different cloud deployment models have different needs and require different approaches. The differences are most distinct between public and private clouds. Public clouds often have geographically diffuse, large resource pools, which require more investment in monitoring traffic and ensuring scalability. Private clouds, even if scalable, are limited to the resources owned by the operating organization. Regarding security, in private clouds the data is under the organization's control, whereas public clouds require continual surveillance across multiple cyber attack vectors. Service metrics such as link availability and connection speed are essential information in public clouds but are of little value in private clouds. In public clouds, firewall settings may limit what can be monitored between cloud providers. Finally, public clouds need to provide monitoring information for clients, which requires more flexibility, customizability, and security.

Cloud Computing Standards

Several organizations are developing cloud standards to promote adoption. The efforts range from security to benchmarks, and emphasize interoperability. Vendor lock-in is one cloud computing issue that concerns potential users and slows uptake; consequently, interoperability plays an important role in cloud computing adoption.

Some of the standardization initiatives most relevant to the present work include:

The Open Cloud Computing Interface (OCCI) Working Group: This Open Grid Forum group has a focus on specifications for interfacing "*aaS" cloud computing facilities. As of the writing of this article, it is a work in progress with two deliverables: a use case, with an associated requirements document, and an OCCI protocol [5]. The latter document "records the needs of IaaS Cloud computing managers and administrators in the form of Use Cases." We found the document to be most useful as a guide

Several organizations are developing cloud standards to promote adoption. The efforts range from security to benchmarks, and emphasize interoperability. Vendor lock-in is one cloud computing issue that concerns potential users and slows uptake; consequently, interoperability plays an important role in cloud computing adoption.

Cloud solutions usually focus on one service model, and sometimes on specific technologies or products. In private clouds, for example, enterprises usually use their own (proprietary) data center to take advantage of cloud computing while inside their firewalls.

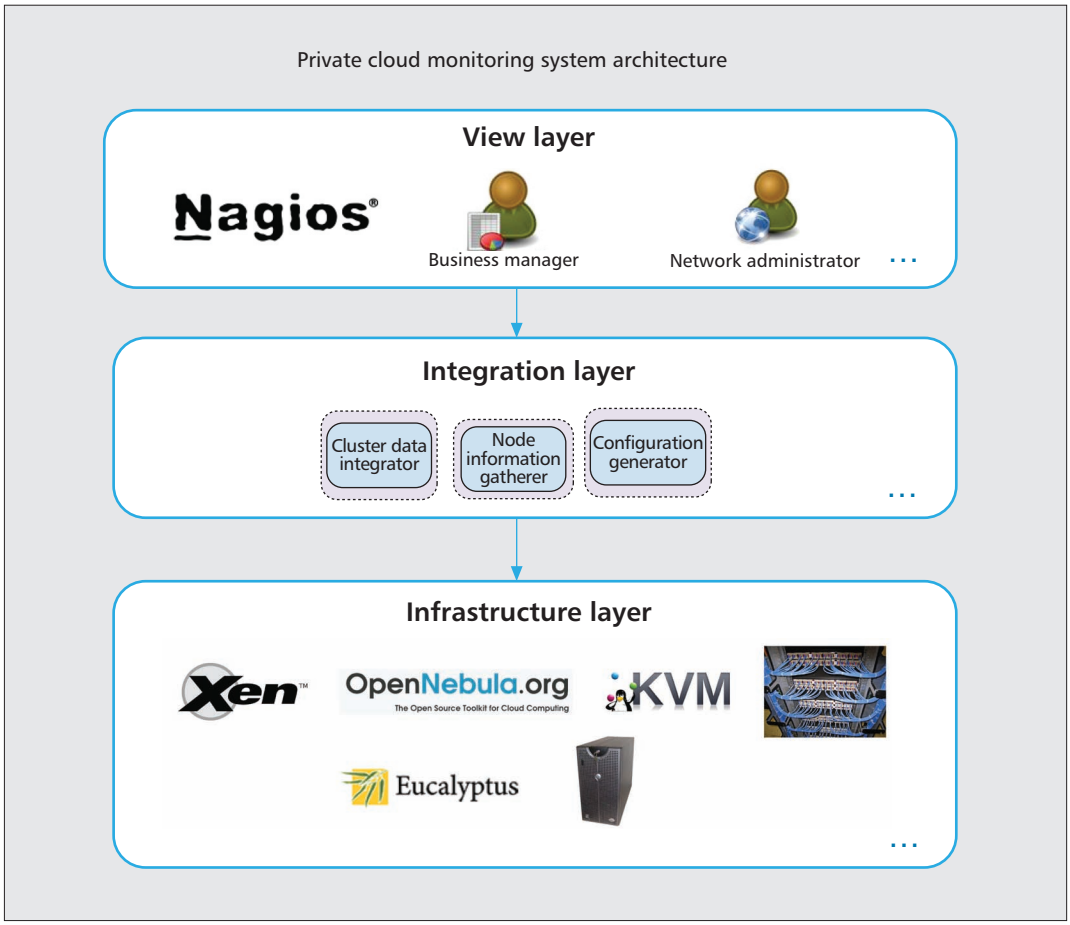

Figure 1. *Private cloud monitoring system architecture.*

for developing tools addressing IaaS managing systems.

Open Cloud Standards Incubator: This initiative, from the Distributed Management Task Force (DMTF), focuses on interactions between cloud environments, their consumers, and developers. The DMTF document "Use Cases and Interactions for Managing Clouds" describes a Monitor Service Resources use case. In the use case, "a cloud consumer configures a monitor for a deployed service instance and resources that support the service instance." Other useful DMTF documents include "Architecture for Managing Clouds," which provides a cloud service reference architecture.

Monitoring Architecture and PCMONS

This section presents an abstract general-purpose monitoring architecture and an associated framework implementation.

Architecture

Cloud solutions usually focus on one service model, and sometimes on specific technologies or products. In private clouds, for example, enterprises usually use their own (proprietary) data center to take advantage of cloud computing while inside their firewalls. For another example, IaaS heavily relies on virtualization technology. There are many available virtualization tools, including the hypervisors Xen, Kernel-Based Virtual Machine (KVM), and others. This means that general-purpose tools for cloud computing may be difficult to build, and causes the open tools market to focus on orchestration technologies, at the expense of open monitoring technologies.

After analyzing the aforementioned characteristics, we designed an abstract general-purpose monitoring architecture. The three-layer architecture of Fig. 1 addresses the monitoring needs of a private cloud, and is extensible, modular, and simple. The middle integration layer provides a clear separation, via abstraction, between infrastructure details and the monitoring information required by cloud users.

Infrastructure — This layer consists of:
- Basic facilities, services, and installations, such as hardware and network
- Available software: operating system, applications, licenses, hypervisors, and so on

Integration — The infrastructure layer is mainly composed of heterogeneous resources, and thus requires a common interface for access. A typical situation is a user requesting a VM instantiation. This type of request could be managed by different hypervisors like Xen or KVM, depending on how the infrastructure layer is deployed. Another possible scenario is the pres-

ence of multiple platforms for cloud computing, like Eucalyptus and OpenNebula. Hence, the monitoring actions to be performed in the infrastructure layer must be systematized before being passed to the appropriate service running in the infrastructure layer. Therefore, the integration layer is responsible for abstracting any infrastructure details.

View — This layer presents as the monitoring interface through which information, such as the fulfillment of organizational policies and service level agreements (SLAs), can be analyzed. Users of this layer are mainly interested in checking VM images and available service levels. Note that this layer may implement different views, according to the enterprise's needs. For example, business managers need different data than network administrators; for example, generic overviews with charts and summarized information are more useful to management than, say, the current state of a particular VM, which is critical information for administrators.

Implementation

Considering the lack of effective generic open source solutions for cloud monitoring and management, as well as the goal of validating the proposed architecture, we developed an extensible modular monitoring framework called PCMONS.

As mentioned earlier, many tools and management practices from distributed computing may be applied in cloud management. During the development of our monitoring framework we tried to incorporate those tools and practices, in order to:

- Ensure that PCMONS integrates seamlessly into organizations' existing management infrastructure and operations
- Leverage the installed software and hardware base as well as the skills and experience of IT administrators

Another consideration is that generally, IaaS tools have three or four high-level components: cloud manager, node controller (or local hypervisor manager), and storage controller, and, when the cloud is divided into clusters, a fourth component: cluster controller. These components were considered when developing the PCMONS, as well as the fact that control in cloud environments is normally centralized [6], making a client/server model suitable for the monitoring purposes.

The current PCMONS version acts principally on the integration layer, by retrieving, gathering, and preparing relevant information for the visualization layer. Monitoring features are organized in modules that consider specific phases of a VM life cycle. The system is divided into the modules listed below, for the purpose of simplifying future adaptations to specific tools and facilitating its study and application:

Node Information Gatherer: This module is responsible for gathering local information on a cloud node. Information gathered might differ according to specific needs, but in the current version it gathers information about local VMs and sends it to the Cluster Data Integrator.

Cluster Data Integrator: As most cloud tools organize nodes into clusters [6], there is a specific agent that gathers and prepares the data for the next layer. This agent avoids unnecessary data transfers from each node to the Monitoring Data Integrator.

Monitoring Data Integrator: Gathers and stores cloud data in the database for historical purposes, and provides such data to the Configuration Generator.

VM Monitor: This module injects scripts into the VMs that send useful data from the VM to the monitoring system. Examples of this data are processor load and memory usage.

Configuration Generator: Retrieves information from the database, for example, to generate the necessary configuration files for visualization tools being used in the view layer.

Monitoring Tool Server: This module is responsible for receiving monitoring data from different resources (e.g., the VM Monitor). The current version it is not fully developed and some shortcuts were taken to pass some monitoring information directly to Nagios. However, its purpose is to receive monitoring information and take actions such as storing it in the Database module for historical purposes.

User Interface: Most monitoring tools have their own user interface. Specific ones can be developed depending on needs, but in our case the Nagios interface is sufficient.

Database: Stores data needed by Configuration Generator and the Monitoring Data Integrator.

Figure 2 depicts a typical private cloud scenario deploying the PCMONS monitoring tool.

The first release of PCMONS is compatible with Eucalyptus at the Infrastructure layer and Nagios at the View layer. However, its development facilitates integration with other cloud toolkits, through the development of extensions.

Regarding the monitoring tool server, currently it generates a configuration file that allows Nagios to monitor and display the monitoring information in its standard interface. The Nagios choice was due to its popularity, documentation, openness and flexibility (easily extended by the development of plug-ins). In addition, Eucalyptus provides a simple Nagios script for basic monitoring of Eucalyptus components.

The Nagios choice was due to its popularity, documentation, openness, and flexibility (easily extended by the development of plug-ins). In addition, Eucalyptus provides a simple Nagios script for basic monitoring of Eucalyptus components.

Case Study

Figure 3 depicts the testbed environment. OpenSUSE was chosen as the operating system of the physical machines, due to its easy installation and configuration, especially of Xen and network, considerably simplified by using the installation and configuration tool YaST. Eucalyptus is an open software solution for cloud computing that offers an interface compatible with Amazon's EC2, widely used and considered by some the de facto standard when it comes to services for the cloud.

Exemplifying a practical situation for the developed tool and strategy, we built an environment where VM images are available for users that instantiate a web server, thus simulating web hosting service provision. That is, the instantiated VMs are Linux servers providing a basic set of tools, acting as web hosting servers. The

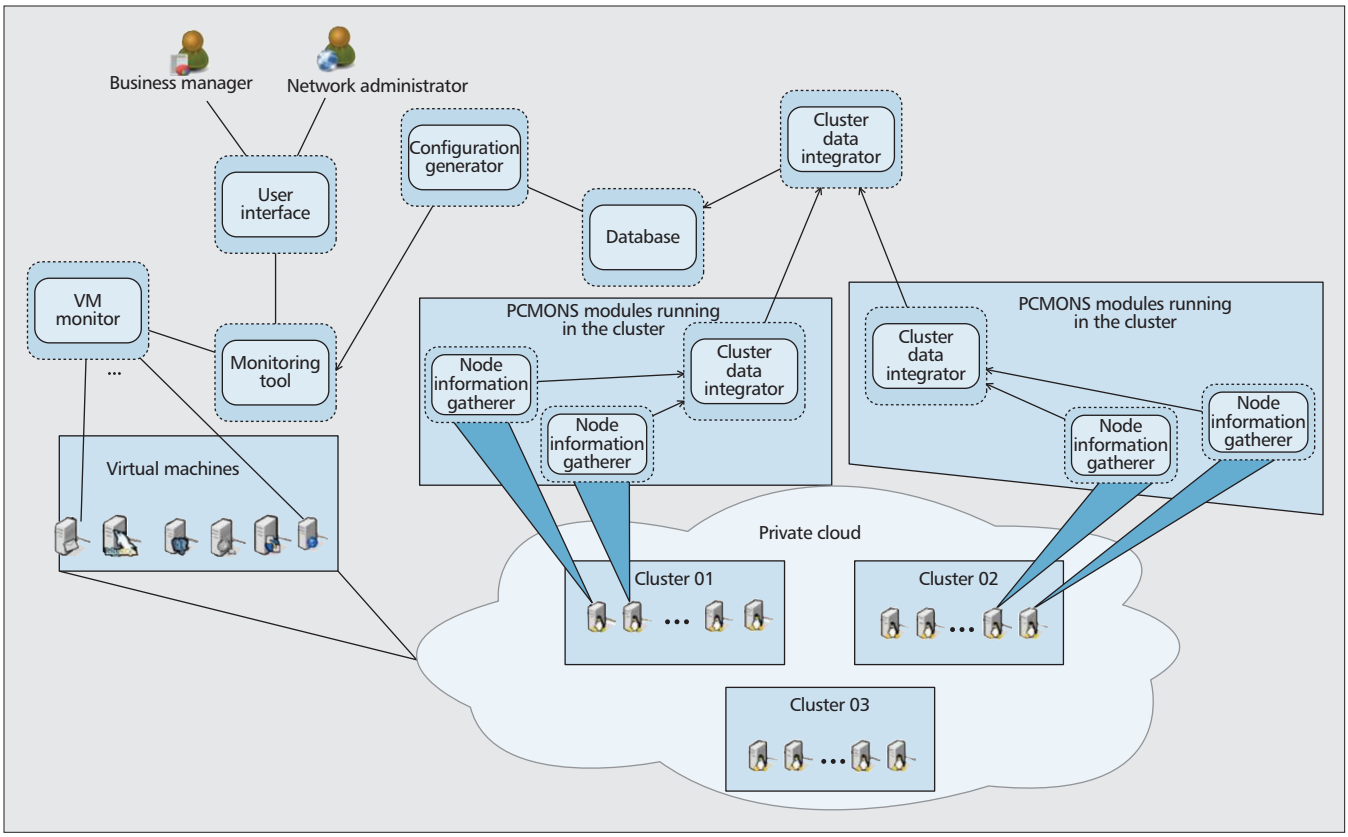

Figure 2. *A typical deployment scenario for PCMONS.*

tools are the Apache Web Server, PHP language interpreter (enabling the use of PHP language for web page development), and SQLite for data persistence. The associated VM images were downloaded from the Eucalyptus website. These images provide a minimal operating system (OS) installation, suitable for different purposes as one may install any application available for Linux distribution. In order to allow specific data monitoring, the VM Monitor module is injected into the VM during boot. The VM Monitor module has an initial set of monitoring plug-ins, responsible for monitoring services such as processor and memory usage. This information could help the administrator automate the instantiation of new or more powerful VMs, depending on resource usage. In addition, data such as ping time, SSH/TLS usage, and HTTP connections are monitored as well.

Figure 4 depicts an overview of hosts and host groups being monitored in our study case. Nagios is used to display monitoring information. The first column shows object names (VM, physical machines, routers, etc.). VM names are an aggregation of user name (who instantiated the VM), VM ID, and the name (or IP address) of the physical machine where the VM is running. The other two columns show service names and their status (OK, Warning, Critical). Figure 4 also shows an overview of host groups created by PCMONS. Another important feature of PCMONS is the physical machine/VM mapping. Groups of running VMs in each physical machine were created to enable faster failure and/or problem root cause identification.

Another deployment scenario for PCMONS is cloud telephony. Recently, many companies are leveraging their telephony solutions to enter the cloud telephony market. As defined by one of these companies, CallFire, cloud telephony is on-demand voice services provided by a hosted, dynamically scalable, often virtualized deployment of VoIP infrastructure. Many of these services may be categorized into the SaaS cloud computing model because they offer tools for developing phone applications. One example is the Cloudvox application programming interface (API), built on the Asterisk open source phone server platform. Most definitions for cloud computing include elasticity and scalability as requirements for any solution to fit into this paradigm; and cloud telephony is no different. Hence, cloud telephony requires a scalable IaaS framework for handling high-demand services like high-volume call setup requests and short message service (SMS) requests. Although PCMONS' main focus is IaaS and not SaaS or PaaS (and many cloud services may be categorized as SaaS), the infrastructure layer of PCMONS' architecture may support monitoring of PaaS and SaaS models. Thus, for example, PCMONS may be used to monitor an Asterisk server delivering SaaS cloud telephony. Furthermore, PCMONS organizes and centralizes monitoring information in forms readily consumed by human administrators, an essential feature for improving management of complex technological infrastructures underlying cloud telephony and cloud computing in general.

Related Work

Grid Monitoring

As of this writing, related work on cloud monitoring is scarce. Because cloud computing makes use of related technologies like utility computing, grid computing, and others, these technologies were considered in the conception of the architecture and PCMONS. Grid computing, for example, has considerable research and implementation efforts and mature results.

Reference [7] introduces the three-layer Grid Resource Information Monitoring (GRIM): the query layer, mediation layer, and information provider layer. The query layer design purpose is similar to PCMONS' visualization layer, i.e., users with different expectations are involved. In the GRIM query layer, users request information about resources, whereas in the PCMONS visualization layer the focus is on enabling different software interfaces for the monitoring data.

The GRIM mediation layer mediates the queries from users and information updates from hosts, whereas in PCMONS the integration layer integrates communication and access to heterogeneous infrastructure. The GRIM information provider layer consists of monitored hosts. A GRIM host is any resource providing grid services that is enabled with sensors to detect its state [7]. In PCMONS architecture, the infrastructure layer has similar behavior. Sensors are added to virtual machines in PCMONS during their boot processes.

Several design issues that should be considered when constructing a Grid Monitoring System (GMS) are presented in [8]. We have selected some and correlated them with PCMONS.

High adaptability: Easy integration and interoperability with existing grid and cloud platforms are necessary. PCMONS is modular, allowing plug-ins deployment for a specific platform.

Heterogeneity: Another characteristic that both grid and cloud environments share is the heterogeneity of resources and their status information. The resultant issue is that a unified logical view of resources and their status is necessary. Heterogeneity is one of the characteristics that drove our design decision to split our solution into layers and modules. The heterogeneity of the infrastructure layer is handled in the integration layer. Users in the visualization layer see only the logical view of the resources and their status, as shown in Fig. 4.

Deployment Transparency: Grid nodes may join and leave quickly, which is incompatible with non-time-consuming error-prone monitoring deployment processes; hence, the need for a transparent monitoring configuration. Resources in a cloud display similar behavior. PCMONS is designed to transparently install the monitoring sensors while the element is joining the cloud.

Reference [9] identifies some differences between cloud monitoring and grid monitoring, especially in terms of interfaces and service provisioning. Clouds have user-centric interfaces (i.e., they do not have to change their working habits and environments, e.g., programming languages), whereas in grids they need to learn grid commands or APIs; therefore, the level of expertise to use a cloud is significantly lower when compared to grids. Another characteristic difference is that clouds are managed by single entities [10], whereas grids may not have any central management entity.

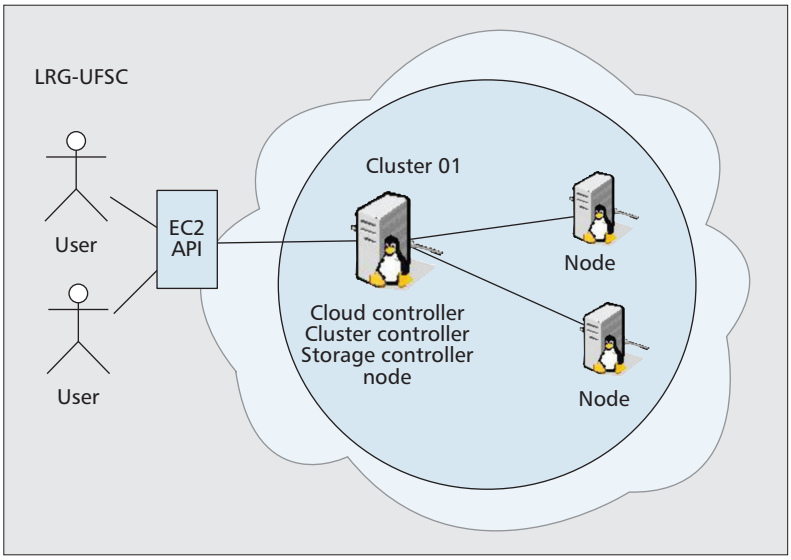

Figure 3. *Testbed environment.*

Cloud Monitoring

To date, there are no publications that address the particulars of private clouds. Reference [11] defines general requirements for cloud monitoring (considering all deployment models) and proposes a cloud monitoring framework. One requirement is multitenancy; however, in private clouds this feature does not have much relevance, because all users belong to the same organization. Regarding the framework, it is based on agents, which may be problematic when devices have different operating systems (e.g., embedded systems) or limited resources. Another possible problem is that this framework does not consider the cloud solution deployed, it only monitors nodes in the network. PCMONS supports two approaches, agents and central monitoring, and is highly adaptable; therefore, little effort is needed to support solutions already in use by the organization, making the migration to a private cloud straightforward.

Key Lessons Learned

Related to Test-Bed Preparation

Despite the fact that clouds abstract complexity and technology details, when preparing a testbed or even a production cloud, some technological issues must be handled directly. In private clouds, a common goal is to take advantage of the existing facilities. Thus, having several different Linux distributions adds an extra configuration burden, for example. Another situation relates to hardware compatibility. Software platforms for cloud computing, such as Eucalyptus and OpenNebula, support a number of different hypervisors, each with its own characteristics. An

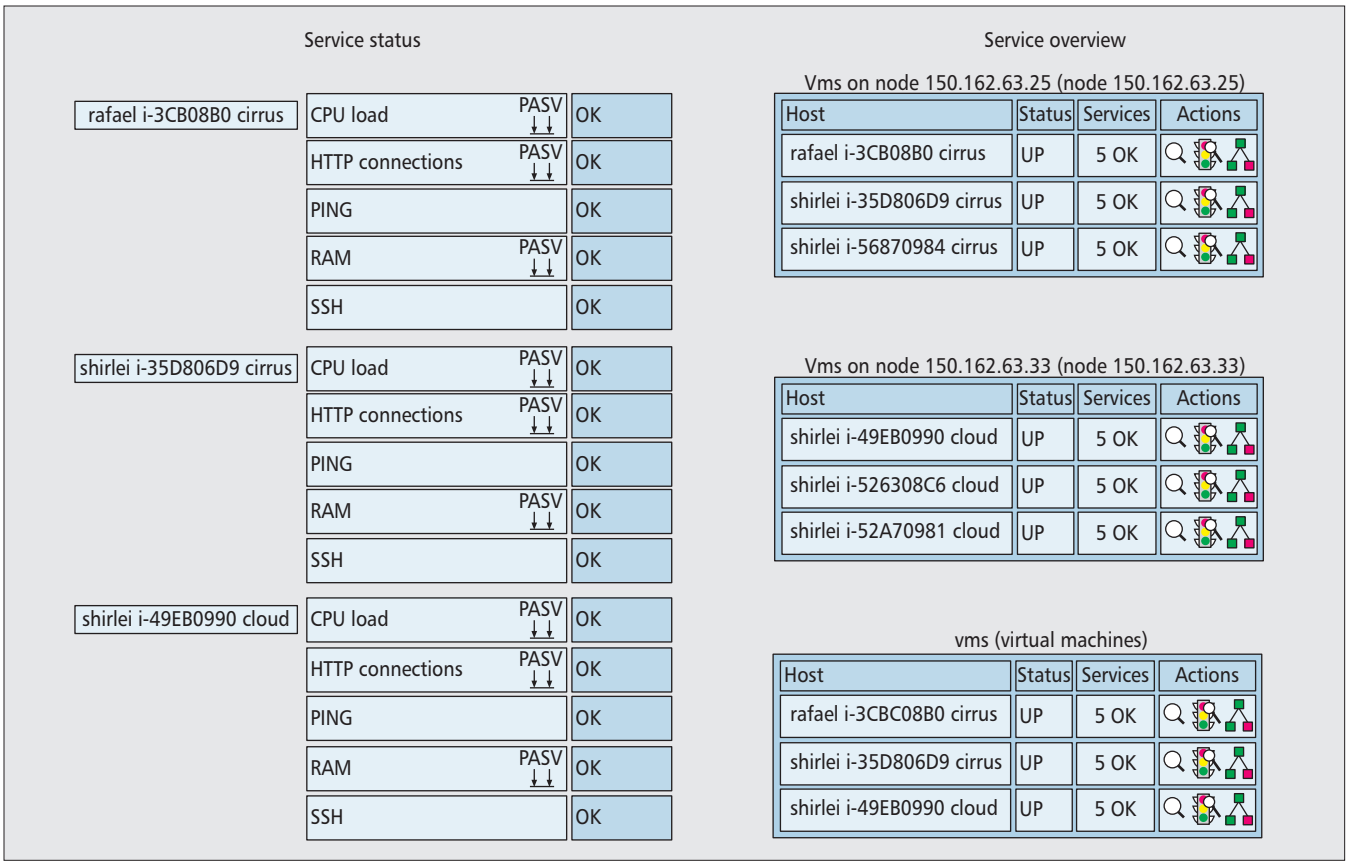

Figure 4. *Representative Nagios interface of the monitored cloud services.*

example is the KVM hypervisor: it has great performance but requires hardware virtualization support that not all processors provide.

Design and Implementation

When implementing a software solution, it is necessary to choose a programming language. This choice depends on many factors, such as how familiar the programming team is with the language, and which language is best suited for the problem. Another decision to make is which platforms to support. Some of these decisions may tightly couple the solution to specific software tools. We opted for solutions well established in the market to facilitate the use of PCMONS in the running structures with little effort and prioritized an adaptable and extensible solution.

As the programming language, first we decided to use Python, but during the development process we were not always able to use that language. Since PCMONS' integration layer handles heterogeneous software, others were necessary, among them Perl (Eucalyptus integration) and Bash script (interaction with Linux); other languages may be necessary in the future.

At the beginning of the project, dynamism was not defined as a requirement due to the reduced environment of a private cloud (compared to public clouds); however, PCMONS testing showed that fast response for new VMs is necessary and should be considered in a private cloud project.

Another lesson learned concerned metrics. Originally, we planned to define some basic common metrics for private clouds, but later found that metrics are often specific to each case.

Root cause detection has proven of utmost importance for private clouds and should be given special attention in cloud environments.

Standardization and Available Implementations

Before choosing a specific tool for private clouds, it is important to verify to what extent cloud standards are implemented by the tool. If no standards are implemented, your data might be locked in to a specific solution, as well as your VMs' configurations. Additionally, extending private clouds into hybrid clouds may be difficult, as will changes in infrastructure and cloud solutions. Some tools, such as OpenNebula, have begun implementing standardization efforts, including the OCCI API. By the time we drafted this article, Eucalyptus had released its 2.0 version, which does not include any new cloud-specific standard, but continues to support an Amazon EC2-compatible interface.

Conclusion and Future Work

This article summarizes some cloud computing concepts and our personal experience with this new paradigm. Also, it shows that cloud computing is not just hyperbole, but a viable way of optimizing existing computing resources in a

datacenter. The current portfolio of open tools lacks open source, interoperable management and monitoring tools. To address this critical gap, we designed a monitoring architecture, and validated the architecture by developing PCMONS. A case study was constructed that reproduced the everyday problems of private cloud users and contributed to their solutions.

During the process, we noted that cloud computing monitoring can benefit from tools and concepts already established for distributed computing management. Furthermore, orchestrating monitoring solutions on installed infrastructures is viable, including the use of well-known monitoring tools like Nagios. On the other hand, even in private clouds, the heterogeneity of computing resources might require some extra effort when implementing the monitoring solution.

We argue that due to its flexible and extensible architecture, PCMONS may be adapted for use by cloud telephony providers to gather and centralize monitoring information, which should improve quality of services.

To monitor specific metrics, especially in an interface-independent manner, a set of preconfigured monitoring plug-ins must be developed. For future work, we intend to improve PCMONS to monitor other metrics and support other open source tools like OpenNebula and Zabbix.

Finally, based on the results achieved with this research, we believe that cloud computing has a bright future. The present work provides first steps toward a monitoring architecture for private clouds.

References

[1] H. Nguyen Van, F. Dang Tran, and J.-M. Menaud, "Autonomic Virtual Resource Management for Service Hosting Platforms," *ICSE Wksp. Software Eng. Challenges of Cloud Computing*, 2009, pp. 1–8.

[2] T. Velte, A. Velte, and R. Elsenpeter, *Cloud Computing Basics*, 1st ed., McGraw-Hill Osborne Media, Sept. 2009.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145 (draft), Jan. 2011, pp. 1–7.

[4] B. Sotomayor *et al.*, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing*, vol. 13, no. 5, Sept. 2009, pp. 14–22.

[5] T. Metsch, "Open Cloud Computing Interface — Use Cases and Requirements for a Cloud API," Open Grid Forum, Sept. 2009.

[6] C. Weinhardt *et al.*, "Business Models in the Service World," *IT Professional,* vol. 11, no. 2, Mar.–Apr. 2009, pp. 28–33.

[7] W. Chung and R. Chang, "A New Mechanism for Resource Monitoring in Grid Computing," *Future Gen. Comp. Sys. 25*, 1 Jan. 2009, pp. 1–7.

[8] M. Yiduo *et al.*, "Rapid and Automated Deployment of Monitoring Services in Grid Environments," *APSCC 2007*, 2007, pp. 328–35.

[9] L. Wang *et al.*, "Scientific Cloud Computing: Early Definition and Experience," *2008 10th IEEE IEEE Int'l. Conf. High Perf. Computing and Commun.*, Sept. 2008, pp. 825–30.

[10] M. Brock and A. Goscinski, "Grids vs. Clouds," *IEEE 2010 5th Int'l. Conf. Future Info. Tech.*, May 2010, pp. 1–6.

[11] P. Hasselmeyer and N. d'Heureuse, "Towards Holistic Multi-Tenant Monitoring for Virtual Data Centers," *2010 IEEE/IFIP NOMS Wksps.*, 19–23 Apr. 2010, pp. 350–56.

Biographies

Shirlei Aparecida de Chaves (shirlei@inf.ufsc.br) holds an M.Sc degree in computer science from the Federal University of Santa Catarina, Brazil. During her M.Sc. she worked at the Networks and Management Laboratory as a cloud computing researcher. She currently works as a systems analyst at E3C Technology, Brazil. Her industrial experience includes network management, Linux development and customization, project feasibility reports on system changes and integration, and close work with clients and developers to ensure technical compatibility. Her research interests include network and service management, and distributed and cloud computing.

Rafael Brundo Uriarte (rafael.uriarte@inf.ufsc.br) is currently working on his M.Sc. in computer science at the Federal University of Santa Catarina. From 2005 to 2007 he worked at the Federation of Industries of Santa Catarina State (FIESC) as a network manager, and in the last three years he has carried out a research in the cloud computing area at the Networks and Management Laboratory. His research interests include autonomic computing, network management, cloud computing, and security.

Carlos Becker Westphall (westphal@inf.ufsc.br) is a full professor in the Department of Informatics and Statistics at the Federal University of Santa Catarina, where he is the leader of the Networks and Management Laboratory. His research interests include network and service management, security, and cloud computing. He received his D.Sc. in computer science at Paul Sabatier University, France. He was the founder of LANOMS. In 2011 he was named an IARIA Fellow. He has served as Technical Program and/or Organizing Committee member (since 1994) of IFIP/IEEE IM, IEEE/IFIP NOMS, IEEE/IFIP DSOM, IEEE LANOMS, and IEEE APNOMS. He has been on the Board of Editors (since 1995) and Senior Technical Editor (since 2003) of the *Journal of Network and Systems Management* of Springer and an Editorial Board member (since 2004) of the *Computer Networks Journal* of Elsevier. He has also been an Associate Editor (since 2006) of the *Journal of Communication and Information Systems* of IEEE ComSoc/SBrT. Since 1993 he has been a member of IFIP TC6 Working Group 6.6 (Management of Networks and Distributed Systems), and since 2003 a member of the core team of the TeleManagementForum Universities Program (TMF UP). Since 2008 he has been Latin America International Academy, Research, and Industry Association (IARIA) Liaison Board Chair. He was a member (2004–2005 and 2006–2007) of the IEEE ComSoc Membership Programs Development Board. From May 2000 to May 2005 he acted as Secretary of the IEEE Committee on Network Operation and Management (CNOM). From May 2005 to May 2009 he acted as Vice-Chair of IEEE CNOM. He has been a member of IEEE CNOM since 1994.

To monitor specific metrics, especially in an interface-independent manner, a set of pre-configured monitoring plug-ins must be developed. For future work, we intend to improve PCMONS to monitor other metrics and to support other open-source tools like OpenNebula and Zabbix.