Principles of Pervasive Cloud Monitoring

Gokce Gorbil, David Garcia Perez, Eduardo Huedo Cuesta

Abstract Accurate and fine-grained monitoring of dynamic and heterogeneous cloud resources is essential to the overall operation of the cloud. In this paper, we review the principles of pervasive cloud monitoring, and discuss the requirements of a pervasive monitoring solution needed to support proactive and autonomous management of cloud resources. This paper reviews existing monitoring solutions used by the industry and assesses their suitability to support pervasive monitoring. We find that the *collectd* daemon is a good candidate to form the basis of a light-weight monitoring agent that supports high resolution probing, but it needs to be supplemented by higher-level interaction capabilities for pervasive monitoring.

Key words: Cloud monitoring; pervasive monitoring; cloud monitoring tools

1 Introduction

Cloud providers need to manage increasingly large and complex cloud infrastructures and assure the availability, reliability, elasticity and performance of offered cloud services [1]. As the scale and complexity of cloud facilities increase, providers need to adopt an autonomous and proactive management approach in order to meet quality-of-service (QoS) guarantees at competitive prices [2]. While monitoring is essential to many activities in the cloud [3], it is especially crucial for resource and

Gokce Gorbil
Imperial College London, Dept. of Electrical and Electronic Engineering, Intelligent Systems and Networks Group, SW7 2AZ London, United Kingdom, e-mail: g.gorbil@imperial.ac.uk

David Garcia Perez
Atos Origin, Barcelona, Spain, e-mail: david.garciaperez@atos.net

Eduardo Huedo Cuesta
Complutense University of Madrid, Dept. of Computer Architecture and Automation, Distributed Systems Architecture Group, 28040 Madrid, Spain, e-mail: ehuedo@fdi.ucm.es

performance management to enable autonomous control, which requires accurate and fine-grained monitoring of virtual [4] and physical computing and storage resources [5,6], applications [7], and the network [8]. In this paper, we review the principles and requirements of pervasive cloud monitoring, and present how concepts from self-aware networks [9] can be leveraged to design a pervasive monitoring solution. We also provide a short review of existing cloud monitoring tools, and discuss integration possibilities with the OpenNebula cloud management platform.

2 Requirements of Pervasive Cloud Monitoring

Cloud monitoring needs to be elastic to accommodate for the dynamism of the cloud environment due to consumer churn and virtualization, and it needs to support runtime configuration changes. For autonomous and proactive cloud management, monitoring data is required to provide an accurate representation of the cloud state and needs to be delivered in a timely fashion in order to enable quick decisions in the face of changes. A pervasive monitoring solution needs to accommodate for volatility in virtual resources [4] and for the migration of virtual machines (VMs) [10]. The locations of the probes of the monitoring system that collect measurements from cloud resources are specified by the cloud layers given below [5], as the layer at which a probe is located limits the types of events and resources that the probe can observe and monitor [11,12]. In this paper, we consider all layers except the facility layer as relevant to the monitoring solution.

- Facility layer: Consists of the physical infrastructure of the data center(s) where the cloud is hosted. Monitoring at this layer considers data center operations, energy consumption, environmental impact, and physical security, such as surveillance and architectural resilience [11].
- Network layer: Consists of the communication links and paths within a cloud, between clouds, and between the cloud and the user.
- Hardware layer: Consists of the physical components of the computing, storage and networking equipment.
- Operating system (OS) layer: Consists of the host and guest operating systems, and the hypervisors, i.e. the virtual machine (VM) managers.
- Middleware layer: Normally only present in the platform-as-a-service (PaaS) and software-as-a-service (SaaS) models, it consists of the software layer between the user application and the OS.
- Application layer: Consists of the user applications running in the cloud.
- User layer: Consists of the end users of the cloud applications and the applications running outside the cloud, e.g. a web browser of the end user.

Due to the complexity of cloud platforms and the heterogeneity of supported applications, there are many *metrics* of interest that need to be monitored concurrently [3]. In addition to actual measurements, all metrics can be evaluated in terms of statistical indicators and temporal characteristics, which need to be supported by the monitoring solution in order to reduce the volume of monitoring traffic. Large scale clouds present challenges in terms of the *scalability* of the monitoring solution due to the high number and types of resources that need to be monitored. Scalability is also important considering that the monitoring solution needs to be able to handle many metrics concurrently, perhaps as many as a hundred. The pervasive monitoring solution therefore needs to efficiently collect, transfer and analyze data from many probes without impairing the normal operations of the cloud. The scalability issue has been mainly addressed by aggregation and filtering of the monitoring data [6,10,13–16]. Other approaches to improve scalability include autonomous monitoring solutions that self-configure, for example by changing the monitoring intervals and parameters [2,17]. In the next section, we discuss how concepts from self-aware networks can be adopted to perform *goal-oriented monitoring*, which addresses, among other things, the scalability issue.

3 Agent-based Adaptive Pervasive Monitoring

One of the concerns in cloud monitoring is congestion and failures in the communication networks since any network disruptions affect not only the hosted cloud services but also the services used by the cloud provider for the monitoring and management of the cloud. Self-aware adaptive overlay networks [9,18] provide an attractive solution to address these issues. We propose that an *inter-cloud overlay network* is constructed in order to monitor inter-cloud communications and to provide resilient and adaptive communications. In this section, we first present the Cognitive Packet Network (CPN) [19,20], which is a self-aware packet routing protocol that implements many of the principles of pervasive monitoring in a network context in order to measure its performance and the conditions of the network in a distributed manner [21]. As the CPN self-monitors and self-manages the communication paths, it can autonomously adapt to changing conditions in the network. Its self-* properties make the CPN a good solution for the monitoring of intra-cloud and inter-cloud communications, and for autonomous adaptation of inter-cloud overlay paths. In addition to monitoring the network links and routers, CPN can also be used as the basis of a pervasive monitoring solution employed to collect measurements from cloud resources. We will present an overview of an adaptive agent-based pervasive monitoring solution based on the CPN later in this section.

Based on measurements it collects from the network nodes and links, CPN adaptively finds the best routes according to QoS criteria, such as low packet delay, specified by the users of the network. CPN employs adaptive learning techniques with random neural networks (RNNs) [22–24] and reinforcement learning [25] in order to make routing decisions at each network node. It uses smart packets (SPs) for exploring the network, dumb packets (DPs), which are source-routed, to carry the payload, and acknowledgments (ACKs) to collect measurements performed by the SPs and DPs and to train the neural networks. SPs are generated by the source node on demand, i.e. when the user requests a new path with a given QoS goal or when it wants to discover parts of the network state. At each hop, SPs are routed based on the experiences of previous packets, employing a learning algorithm, mainly reinforcement learning on RNNs. The RNN is a neural network model inspired by biology, and it is characterized by positive (excitatory) and negative (inhibitory) signals in the form of spikes of unit amplitude which are exchanged between neurons and alter the potential of the neurons. A neuron is connected to one or more neurons, forming a neural network; each neural link has a weight, which can be positive or negative.

In the RNN, the state q_i of the ith neuron represents the probability that it is excited, and satisfies the following system of non-linear equations:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}$$

where

$$\lambda^+(i) = \sum_j q_j w^+_{ji} + \Lambda^+_i, \quad \lambda^-(i) = \sum_j q_j w^-_{ji} + \Lambda^-_i, \quad r(i) = \sum_j w^+_{ij} + w^-_{ij}$$

w^+_{ji} is the rate at which neuron j sends excitation spikes to neuron i when j is excited, w^-_{ji} is the rate at which neuron j sends inhibition spikes to neuron i when j is excited, and $r(i)$ is the total firing rate of neuron i. Λ^+_i and Λ^-_i are the constant rates of the external positive and negative signal arrivals at neuron i, respectively. These external signal arrivals follow stationary Poisson distributions. For an N neuron RNN, the parameters are the $N \times N$ weight matrices W^+ and W^- which need to be learned from the inputs.

Each node in the CPN stores an RNN for each QoS goal and source-destination pair. Each RNN has a neuron for each communication link at the node. The RNN is used to make adaptive decisions regarding the routing of SPs by routing the SPs probabilistically according to the weights of the neurons, which are updated using reinforcement learning. The QoS goal of the RNN is expressed as a function to be minimized, and the reward used in the reinforcement learning algorithm is simply the inverse of this function. Each received ACK triggers an update of the RNN based on the performance metrics observed by the SP or DP that resulted in the ACK.

We adopt the CPN in order to monitor cloud communications. In addition, we adopt concepts from the CPN to design an agent-based adaptive monitoring solution that performs goal-based monitoring, where metrics relevant to performance management are measured locally at each resource by *monitoring agents*, and collected on demand by *monitoring managers* to enable intelligent resource allocation decisions.

The use of configurable software agents has been proposed in the literature in order to construct a flexible monitoring architecture [26]. In agent-based monitoring solutions, agents located at the physical and virtual resources implement one or more probes that collect measurements of metrics from the resource either on demand or periodically. Agent-based solutions are attractive for pervasive cloud monitoring since they provide elasticity, as the monitoring agents can be designed so that they start-up and shut-down with the resource they are attached to, e.g. VMs, and migrate with them, enabling easy monitoring of VMs even under migration. Furthermore, agents can be contacted at run-time in order to enable and disable monitoring activities on a resource-by-resource basis, providing a highly configurable and adaptive solution.

Monitoring agents store their measurements in local repositories. Smart packets are sent by the monitoring managers in order to collect measurements from these repositories, similar to the way that SPs are used in standard CPN by the source nodes. The monitoring managers make the collected measurements available to the resource management system to enable QoS-based decisions, and they may also configure monitoring activities of the agents depending on their QoS goals. It is expected that multiple monitoring managers will exist for fault tolerance and load balancing purposes.

4 A Review of Cloud Monitoring Tools

We provide a review of some of the popular cloud monitoring tools in Table 1; a more comprehensive review can be found in [27]. Out of the reviewed monitoring tools, Zabbix, Ganglia and Nagios are the most capable and flexible. They offer a single tool for the monitoring of different system and application metrics, enable the user to set-up and receive alerts and notifications, provide aggregation mechanisms for the monitored metrics, and keep a historical record of the monitoring data. All three projects are widely used and have strong community support.

Despite the capabilities of Zabbix, Ganglia and Nagios, we believe that *collectd* is the best candidate to form the basis of a light-weight monitoring agent as part of the pervasive monitoring solution outlined in the previous section, since it supports a wide variety of metrics and high resolution probing for many concurrent probes. collectd can also probe at different layers, e.g. at the application, middleware and OS layers, and thus can support both high-level and low-level monitoring. It is also customizable via plug-ins, allowing the addition of more application-specific probes. In order to add further capabilities to the pervasive monitoring solution, such as visualization and analysis, the system needs to be interfaced with higher-level monitoring tools. Nagios appears to be the best choice for this purpose due to its extensibility and flexibility. Measurements from the collectd-based monitoring agents can be provided to Nagios via the collectd-nagios plug-in, and alerts for the user can be configured and transmitted by Nagios. Nagios would also present a front-end to the human user for visualization of metrics and configuration of the monitoring agents, which is possible by extending Nagios.

Tool	Description
Collectl	A lightweight monitoring tool that collects system-level information. Provides basic monitoring only, and limited to Linux systems.
SAR	Offers basic system-level monitoring. Graphs supported via SAG.
SIGAR	A portable API for system-level monitoring, providing bindings for Java, Python, Ruby, C#, etc. Not currently developed.
Monit	A utility for monitoring and managing processes, programs, files, directories, and devices on Unix and Linux systems. Designed to conduct automatic maintenance and repairs.
sFlow	Network traffic monitor. Widely supported by networking equipment, such as routers.
Munin	Client-server based monitoring tool, designed for easy visualization of monitoring data. Mainly for Linux systems, but Windows systems are supported via third party plug-ins. Can be extended to perform actions based on the gathered data.
Ganglia	A scalable distributed monitoring system developed for computing clusters and grids. Uses a multicast-based publish-subscribe protocol to decouple communication endpoints. Employs popular and proven technologies to represent (XML), transfer (XDR) and store monitoring data (RRDtool). Provides a graphical interface for visualization of monitoring data, in addition to an alert system. Extensible via custom clients.
Nagios	A full monitoring solution designed to monitor applications, services, operating systems, network protocols and system metrics with a single tool. Supports configuration of actions to take when certain conditions are satisfied. Provides visualization and alerts.
Zabbix	Monitoring solution mainly designed to monitor networks and network services. Uses SQL databases to store monitoring data, and provides a web front-end and an API to access it. System-level metrics on the hosts can be monitored via the provided agent.
collectd	A lightweight daemon for periodically collecting system metrics, supporting a wide variety of metrics at different layers, e.g. application, OS, networking. Supports high resolution probing. Extensible via plug-ins. Provides several mechanisms to store the monitoring data, e.g. via the RRDtool.

Table 1: Cloud monitoring tools

5 Monitoring with OpenNebula

The autonomous cloud management system employs a cloud management toolkit that provides a “dumb” interface in order to actuate its decisions. The cloud community currently favors two open-source industry standard cloud management toolkits: OpenStack and OpenNebula. Both projects provide tools and services to set-up, configure and manage a cloud platform. At present, OpenNebula provides a more mature software stack, and easier installation and configuration.

OpenNebula relies on monitoring the cloud infrastructure in order to assess the state of the resources. OpenNebula’s monitoring subsystem gathers information from the physical hosts and the VMs by executing a set of static probes in the monitored resources, the output of which are sent to the cloud manager using either push or pull mode. When using the *pull mode*, the manager periodically and actively queries each host. This mode is limited by the number of active connections that can be made concurrently since hosts are queried sequentially. As such it is only appropriate for small-scale clouds, consisting of 50 hosts or less, and when low update frequencies are acceptable (e.g. for 50 hosts, the monitoring period would typically be around 5 minutes). In the UDP-based *push mode*, each host periodically sends monitoring data to the manager. This is more scalable than the pull mode and therefore better suited for larger infrastructures and high update frequencies.

OpenNebula allows simple customization of the monitoring drivers and the creation of new drivers which exclusively use the pull model to report the data. OpenNebula also provides the *OneGate module* to enable VMs to push application-related monitoring information to the manager. However, OneGate is designed for only small-scale application-layer monitoring. The pervasive monitoring solution and OpenNebula can be integrated by either implementing a new OpenNebula monitoring driver in order to interface the monitoring agents directly with OpenNebula, or by integrating OpenNebula with a third-party tool such as Nagios or Ganglia that acts as an intermediary between the monitoring agents and OpenNebula.

6 Future Work

We provided an overview of an agent-based design for a pervasive monitoring solution based on concepts from self-aware networks, and reviewed some of the popular cloud monitoring tools. Among the tools reviewed, collectd is the best candidate to form the basis of a light-weight monitoring agent. In future work, we will extend our initial monitoring design, and evaluate the performance of resource allocation decisions based on goal-oriented monitoring.

Acknowledgements The work presented in this paper was partially supported by the EU FP7 research project PANACEA (Proactive Autonomous Management of Cloud Resources) under grant agreement no. 610764.

References

1. M. Armbrust, A. Fox, R. Griffth, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
2. T. Lorimer and R. Sterritt. “Autonomic management of cloud neighborhoods through pulse monitoring,” *in Proc. 5th IEEE Int. Conf. on Utility and Cloud Computing (UCC’12)*, pp. 295–302, Nov. 2012.
3. G. Aceto, A. Botta, W. de Donato, and A. Pescape. “Cloud monitoring: a survey,” *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013.
4. F.-F. Han et al. “Virtual resource monitoring in cloud computing,” *Journal of Shanghai University (English Ed.)*, vol. 15, no. 5, pp. 381–385, 2011.

5. J. Montes et al. "GMonE: a complete approach to cloud monitoring," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2026–2040, Oct. 2013.
6. J. Povedano-Molina et al. "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant clouds," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2041–2056, Oct. 2013.
7. K. Alhamazani et al. "Cloud monitoring for optimizing the QoS of hosted applications," *in Proc. 4th IEEE Int. Conf. on Cloud Computing Technology and Science (CloudCom'12)*, pp. 765-770, Dec. 2012.
8. L. Atzori, F. Granelli, and A. Pescape. "A network-oriented survey and open issues in cloud computing," *in Cloud Computing: Methodology, Systems, and Applications*, pp. 91-108, CRC Press, 2011.
9. E. Gelenbe. "Steps toward self-aware networks," *Communications of the ACM*, vol. 52, no. 7, pp. 66–75, Jul. 2009.
10. B. Konig, C. J. M. Alcaraz, and J. Kirschnick. "Elastic monitoring framework for cloud infrastructures," *IET Communications*, vol. 6, no. 10, pp. 1306–1315, Jul. 2012.
11. J. Spring. "Monitoring cloud computing by layer, part 1," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 66–68, Mar. 2011.
12. J. Spring. "Monitoring cloud computing by layer, part 2," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 52–55, May 2011.
13. Y. Meng, Z. Luan, Z. Cheng, and D. Qian. "Differentiating data collection for cloud environment monitoring," *in Proc. 2013 IFIP/IEEE Int. Symp. on Integrated Network Management (IM'13)*, pp. 868–871, May 2013.
14. J. S. Ward and A. Baker. "Monitoring large-scale cloud systems with layered gossip protocols," *arXiv Computing Research Repository*, vol. abs/1305.7403, May 2013.
15. H. T. Kung, C.-K. Lin, and D. Vlah. "CloudSense: Continuous fine-grain cloud monitoring with compressive sensing," *in Proc. 3rd USENIX W'shop on Hot Topics in Cloud Computing (HotCloud'11)*, Jun. 2011.
16. C. Canali and R. Lancellotti. "Improving scalability of cloud monitoring through PCA-based clustering of virtual machines," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 38–52, Jan. 2014.
17. G. Katsaros et al. "A self-adaptive hierarchical monitoring mechanism for clouds," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1029–1041, May 2012.
18. R. Lent, O. H. Abdelrahman, and G. Gorbil. "A low-latency and self-adapting application layer multicast," *Computer and Information Sciences*, series LNEE, vol. 62, pp. 169–172. Sep. 2010.
19. E. Gelenbe, R. Lent, and A. Nunez. "Self-aware networks and QoS," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1478–1489, Sep. 2004.
20. E. Gelenbe, Z. Xu, and E. Seref. "Cognitive packet networks," *in Proc. 11th Int. Conf. on Tools with Artificial Intelligence*, pp. 47–54, Nov. 1999.
21. G. Sakellari. "The cognitive packet network: a survey," *The Computer Journal*, vol. 53, no. 3, pp. 268–279, Jun. 2009.
22. E. Gelenbe. "Sensible decisions based on QoS," *Computational Management Science*, vol. 1, no. 1, pp. 1–14, Dec. 2003.
23. E. Gelenbe and S. Timotheou. "Random neural networks with synchronised interactions," *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, Sep. 2008.
24. E. Gelenbe and K. Hussain. "Learning in the multiple class random neural network," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1257–1267, Nov. 2002.
25. U. Halici. "Reinforcement learning with internal expectation for the random neural network," *European Journal of Operational Research*, vol. 126, no. 2, pp. 288–307, Oct. 2000.
26. R. Aversa, L. Tasquier, and S. Venticinque. "Management of cloud infrastructures through agents," *in Proc. 3rd Int. Conf. on Emerging Intelligent Data and Web Technologies (EIDWT'12)*, pp. 46–52, Sep. 2012.
27. K. Alhamazani et al. "An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art," *arXiv Computing Research Repository*, vol. abs/1312.6170, Dec. 2013.