Real-time QoS monitoring for Cloud-based Big Data Analytics Applications in Mobile Environments

Khalid Alhamazani[1], Rajiv Ranjan[2], Prem Prakash Jayaraman[2], Karan Mitra[3], Meisong Wang[2], Zhiqiang (George) Huang[2], Lizhe Wang[4], Fethi Rabhi[1]
[1]School of Computer Science and Engineering, University of New South Wales
{ktal130,Fethir}@cse.unsw.edu.au
[2]CSIRO Computational Informatics
{rajiv.ranjan, prem.jayaraman, deanmeisong, zhiqiang.huang}@csiro.au
[3]Luleå University of Technology, Skellefteå Campus, 93187 Skellefteå, Sweden
karan.mitra@ltu.se
[4]Chinese Academy of Sciences, Beijing, China
{lizhe.wang}@gmail.com

***Abstract*—The service delivery model of cloud computing acts as a key enabler for big data analytics applications enhancing productivity, efficiency and reducing costs. The ever increasing flood of data generated from smart phones and sensors such as RFID readers, traffic cams etc require innovative provisioning and QoS monitoring approaches to continuously support big data analytics. To provide essential information for effective and efficient bid data analytics application QoS monitoring, in this paper we propose and develop CLAMS—Cross-Layer Multi-Cloud Application Monitoring-as-a-Service Framework. The proposed framework: (a) performs multi-cloud monitoring; and (b) addresses the issue of cross-layer monitoring of applications. We implement and demonstrate CLAMS functions on real-world multi-cloud platforms such as Amazon and Azure.**

Keywords- multi-clouds; cross-layer monitoring; QoS; cloud computing

I. INTRODUCTION

Cloud computing is an emerging ICT service paradigm, that offers a flexible access to huge pool of resources with practically no capital investment and with modest operating costs proportional to the actual use (pay-as-you use model) [2]. Emerging trends in big data analytics supported by advances in cloud computing have shifted the focus from "What data should we store" to "What can we do with the data" [1] leading to Analytics-as-a-service model. Big data analytics offer valuable insight into data that can offer a competitive advantage to organizations.

To support an Analytics-as-a-service model in the cloud specifically in environments where floods of data generated from smart phone and sensors are increasing by the day and unpredictable, the goal is to develop techniques that can ensure a guaranteed level of application performance. The big data analytics in the cloud e.g. Mahout supported by Hadoop require real-time QoS monitoring across the cloud layers to ensure application's availability and performance. Consider an example of a crowd-sensing application that is supported by a distributed big data analytics application (Hadoop + Mahout) in the cloud. The volume and variety of data depends on the number of users contributing data which in most cases is not known before-hand. Hence, it is essential to continuously monitor individual system performance at different layers such as Hadoop job tracker (PaaS offering), HDFS layer (PaaS offering) and VM performance/failure (IaaS offering). The above QoS parameters cumulatively affect the QoS of end-user of that big data analytics application (SaaS offering).

Further, typical of distributed application environments, the components could be deployed across multiple cloud environments. Hence, there is a need for the QoS monitoring service to work seamlessly in multi-cloud environments such as Amazon and Azure. E.g. monitoring frameworks such as CloudWatch and Fabric Controller provided by Amazon AWS and Microsoft Azure cannot monitor application/resources hosted on other cloud provider platforms.

Reliable and efficient management of application performance hosted on the cloud layers can be ensured by continuously monitoring the compute, storage, networking resources, application performance and their respective quality of service (QoS) across the layers. Primarily, monitoring is vital for: i) managing the QoS of resources offered by the cloud; ii) providing continuous information on the status of resources to cloud providers and application administrators; and (iii) detecting and debugging software and hardware problems affecting applications' QoS.

The previously stated arguments exemplify the need for cloud monitoring techniques with specific emphasis on the need for monitoring application components across layers and across multiple cloud provider environments. To develop optimized automatic big data analytics provisioning strategies, there is a need to gain more accurate monitoring data on application performance more specifically monitor the performance of application components individually. We aim to break free of current black-box based cloud monitoring approaches [4, 5, 6] that give very little attention to individual components of applications provisioned across cloud layers. Hence, this renders the need for a uniform, extensive and effective multi-cloud, cross-layer monitoring framework that aid in controlling the application QoS based on real-time monitoring of the status of application components and underlying cloud platform (hardware and software).

In this paper, we propose and demonstrate CLAMS—A Cross-Layer Multi-Cloud Application Monitoring-as-a-Service Framework [7]. The novel features of CLAMS includes: (i) ability to monitor and profile QoS of applications, whose parts or components are distributed across multiple public or private clouds and (ii) ability to provide visibility into QoS of individual components of

978-1-4799-5705-7/14 $31.00 © 2014 IEEE
DOI 10.1109/MDM.2014.74

IEEE computer society

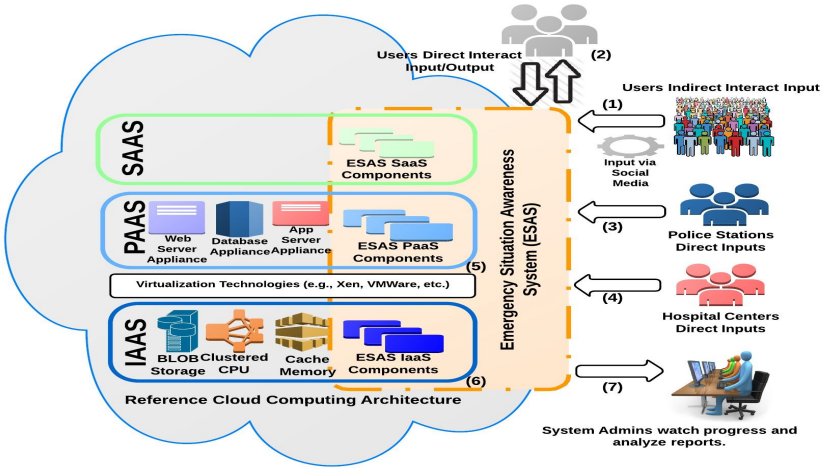

Figure 1. Example of a Emergency Situation Awareness System using Cloud Data Analytics

application stack (e.g., web server, database server). CLAMS follows a agent-based approach that is cloud provider environments agnostic. We present a proof-of-concept implementation of CLAMS framework monitoring application QoS across layers deployed in multi-cloud environments such as Amazon AWS and Microsoft Azure.

II. SCENARIO AND RESEARCH CHALLENGE

To illustrate the need and function of the proposed CLAMS framework, consider a scenario of an Emergency Situation Awareness system (ESAS) as depicted in Fig. 1. Systems such as ESAS are required to efficiently manage and respond to situations like public demonstrations, interior domestic clashes, revolutions, major festivals, and major public/national events. The system is a typical example of a big data analytics system including functions such as continuous data mining on data obtained from crowds and social media to detect potential danger, machine learning algorithms to predict future occurrences of events i.e. modeling of event outcomes based on current data etc. In such situations, sideline of danger and emergency cases cannot be expected. Hence, an immediate and continuous monitoring is required by authorities e.g. Police, and National Guards.

To support a system such as ESAS that requires a round-the-clock operation, robust techniques are required to ensure system performance and availability. Based on historical inferences, ESAS systems use certain policies and procedures to define Service Level Agreements (SLAs). Such policies and procedures are formulated to handle/avoid ESAS bottlenecks in terms of known QoS parameters e.g. e.g. network traffic hazards, VM failures etc. We go further and consider the following situations which foster the need for multi-cloud cross layer monitoring for optimized provisioning of big data analytics applications.

In some events, public users contributing data may increase phenomenally providing valuable inputs related to that event (Step 1). Such burst of input data is hard to estimated or predict. Moreover, dynamic changes in situation might require additional machine learning algorithms to be deployed on-demand to process and filter incoming data. The ESAS system will be required to cope with such dynamic demands to changing data patterns maintaining high level of system stability and availability.

When events as described previously occur, the demand on the system increases significantly. Further, in such situations, the interactions between the system and other users e.g. police, hospital etc., also increase as more events and dangerous situations are identified by the big data analytics algorithm running in the cloud. To cope with such dynamic situations, monitoring the entire VM as a black-box may not be sufficient to guarantee systems SLA. Knowledge on individual component performance can greatly help auto-provisioning systems make better-informed decisions providing insights into the following:

A. *Which layer failure occurs (which application's stack component)*

In case of failures, the challenge is to identify which component of the application that caused the failure. Knowing individual component performance accurately can greatly help in auto-scaling the corresponding layer at the right timing to avoid failure. To illustrate, when input data load increase, the load on the corresponding system components increase (e.g. queuing, Mahout, HDFS) that may lead to system failure. If the monitoring approach being adopted cannot do cross layer monitoring, scaling and provisioning may not detect the issue until it impacts the entire VM. On the other hand, if the monitoring is performed across cloud layers, identifying and rectifying a specific component (e.g. DB component at PaaS layer) in case of failure is possible providing means for more intelligent system scaling.

B. *Certainty of which cloud platform failure occurs (which application's stack component)*

In situations, where the ESAS components are hosted across different regions by different cloud providers CLAMS can monitor all application stack components distributed individually independent of the cloud platforms. Therefore,

system scaling can take the appropriate action eventually resulting in maintaining the agreed SLAs.

Fig. 2 presents the key strengths of the CLAMS framework and examples of monitoring application components across layers and cloud provider environments.

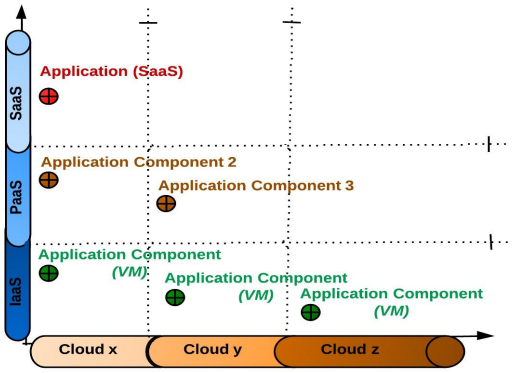

Figure 2: CLAMS – Cloud Monitoring Framework across Layers in Multi-cloud Environments

III. CLAMS: CROSS-LAYER MULTI-CLOUD APPLICATION MONITORING AS A SERVICE FRAMEWORK

CLAMS include mechanisms for efficient cloud monitoring at different *aaS layers. CLAMS provides standard interfaces and communication protocols that enable application/system administrator to gain awareness of the whole application stack across different cloud layers in heterogeneous, hybrid environments (monitor VMs hosted on different cloud platforms). In this way, CLAMS also satisfies the challenges related to interoperability between heterogeneous cloud resources. Fig. 3 presents a detailed architecture of the proposed CLAMS framework.

A. CLAMS Monitoring Manager

The CLAMS Monitoring Manager is a software component that collects QoS information from CLAMS Monitoring Agents running on several virtual machines (VMs) across multi-cloud environments. In particular, the monitoring manager collects the QoS values from the agents running at the SaaS, PaaS and IaaS layers. As soon as the monitoring system is initialized on the cloud(s), the VMs running the CLAMS manager(s) and the agents boot up. Using discovery mechanisms like broadcasting, selective broadcasting or decentralized discovery mechanisms [3], the agents and manager discover each other. The CLAMS monitoring manager employs a QoS data collection schema to store QoS statistics into the local database and an agent schema to maintain the list of discovered agents. The CLAMS monitoring manager also incorporates an API that is used by other monitoring manager or external service to share the QoS statistics.

B. CLAMS Monitoring Agent

Another key component of the CLAMS framework is the monitoring agent. The monitoring agent resides on VM running on the cloud and collects and sends QoS parameter values as requested by the manager. After the monitoring system initialization, the agent waits for the incoming requests from the manager or starts to push QoS data to the manager. Upon arrival of the request, the agent retrieves the stated QoS values belonging to a given process and/or a system resource and sends them back as a response to the Manager.

IV. SYSTEM IMPLEMENTATION

The proof-of-concept implementation of the proposed CLAMS framework has been developed using Java and is completely cross-platform interoperable i.e., it works on both

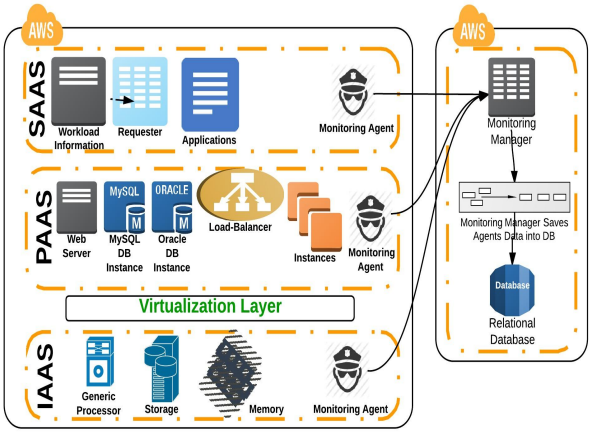

Figure 3: CLAMS Framework Architecture

Windows and/or Linux operating systems. Fig. 4a, 4b, 4c presents proof-of-concept implementation screenshots. Table 1 presents the list of application components and the corresponding layers that were monitored for demonstration purpose.

Table 1: Monitoring across different layers

Process/Resource	Description	Owner
Tomcat7w.exe	Apache Tomcat 7	User
MySqld.exe	MySQL Workbench 6.0	User
Javaw.exe	Monitoring Manager	User
Lsass.exe	Local Security Authority Process	System
Winlogon.exe	Windows Logon App.	System
Services.exe	Services and Controller App.	System
VM CPU Usage	CPU usage of the entire VM	System
VM Memory Usage	Memory usage of the entire VM	System

Monitoring Agent Implementation: The process of retrieving QoS targets is done by utilizing functionalities provided by SNMP, SIGAR and other custom built APIs. For instance, SNMP is used to retrieve the QoS values related to networking, number of packets in and out, route information and, number of network interfaces. SIGAR (http://www.hyperic.com/products/sigar) is used to obtain access to low-level system information such as CPU usage, actual used memory, actual free memory, total memory and process specific information (e.g. CPU and memory

consumed by a process). To enable SNMP monitoring, we define new SNMP Objects Identifiers (OIDs) in a sequence. For example function to get the CPU usage of a specific process (tomcat) is assigned an OID .1.3.6.1.9.1.1.0.0. Similarly, function to get process memory is assigned an OID .1.3.6.1.9.1.1.0.1.

Monitoring Manager Implementation: The monitoring manager uses a MySQL database to store the QoS statistics collected from the agents. For the proof-of-concept implementation, we used a pull approach where the Manager is responsible to poll for QoS data from agents distributed across multiple cloud provider VMs. The manager uses a simple broadcasting mechanism for agent discovery. The polling interval is a pre-defined constant and can be changed using the manager configuration files.

Agent Manager Communication: For the proof-of-concept implementation, the communication between the agent and the manager has been implemented using two techniques namely RESTful Web services and SNMP. Having a RESTful approach enables easy lightweight communication between CLAMS agents and manager/super manager. Using a standardized SNMP interface makes CLAMS compatible with existing SNMP-based applications, tools and system and reduces the effort involved in collecting QoS statistics.

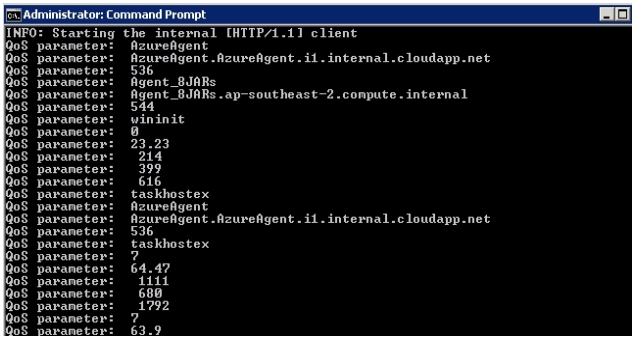

Figure 4a: CLAMS proof-of-concept Implementation

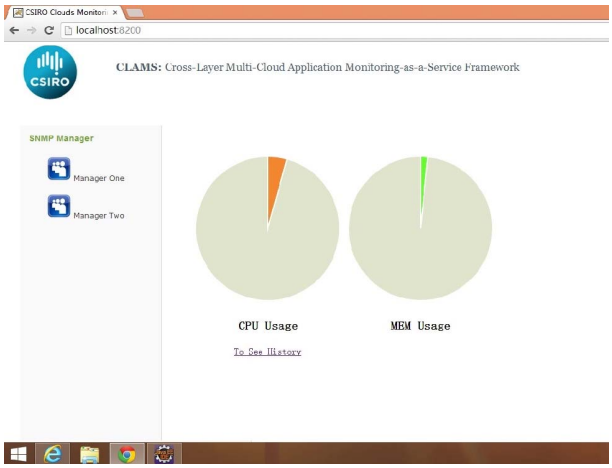

Figure 4b: Screenshots of CPU and MEN Usage

Figure 4c: Screenshots of the Curve of CPU Usage

Monitoring Manager Interface: The monitoring manager implements a RESTful API allowing other applications to request QoS data of individual application components running in a multi-cloud environment. The web interface consumes the API to present the data about applications and agents visually. The web interface is developed using HTML5 and java scripts. Fig. 4b, and 4c presents the web interface screenshots.

V. CONCLUSION

In this paper, we proposed CLAMS, a multi-cloud cross layer cloud monitoring framework. CLAMS provides its users the ability to monitor applications by providing insights into individual component performance as against a black-box approach. We implemented and demonstrated CLAMS in a multi cloud environment (Amazon and Azure) monitoring processes across layers. Our implementation outcomes show the proposed approach is feasible and does not impose significant overheads on the application.

REFERENCES

[1] Intel, “Big Data in the Cloud: Converging Technologies”, Febuyrary 2013. Available from: http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/big-data-cloud-technologies-brief.pdf

[2] R. Ranjan, K. Mitra, D. Georgakopoulos, “MediaWise Cloud Content Orchestrator”, *Journal of Internet Services and Applications, Springer*, vol. 4, Jan 2013.

[3] R. Ranjan; L. Chan; A. Harwood.; S. Karunasekera; R. Buyya., "Decentralised Resource Discovery Service for Large Scale Federated Grids," e-Science and Grid Computing, IEEE International Conference on , vol., no., pp.379,387, 10-13 Dec. 2007.

[4] M. K. Nair and V. Gopalakrishna, ",ÄòCloudCop‚Äô: Putting network-admin on cloud nine towards Cloud Computing for Network Monitoring," in *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, 2009, pp. 1-6.

[5] S. A. De Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *Communications Magazine, IEEE,* vol. 49, pp. 130-137, 2011.

[6] M. Kutare, G. Eisenhauer, C. Wang, K. Schwan, V. Talwar, and M. Wolf, "Monalytics: online monitoring and analytics for managing large scale data centers," in *Proceedings of the 7th international conference on Autonomic computing*, 2010, pp. 141-150.

[7] K. Alhamazani, R. Ranjan, K. Mitra, P. P. Jayaraman, Z. Huang, L. Wang, and F. Rabhi, “CLAMS: Cross-Layer Multi-Cloud Application Monitoring-as-a-Service Framework” *2014 IEEE SCC.* **[ACCEPTED]**