4th Imeko TC19 Symposium on Environmental Instrumentation and Measurements
Protecting Environment, Climate Changes and Pollution Control
June 3-4, 2013, Lecce, Italy

AN EVENT-BASED SERVICE FOR SENSING DATA PROVISIONING IN THE CLOUD

M. Fazio, A. Puliafito, M. Villari
Universita' di Messina, Contrada di Dio, S. Agata, 98166, Messina, Italy, mfazio@unime.it

Abstract — The Cloud is arising as strategic technology in many application fields, since the theoretically unlimited availability of computing and storage resources in the Cloud. Recently, new Cloud services are offering interesting solutions also for processing and storing huge amount of data generated from thousands heterogeneous sensors deployed all around. In this paper, we present an innovative Cloud architecture helpful to efficiently manage a large set of data, in order to bringing together data provided by heterogeneous sensing infrastructures with the requirements of many data consumers. It allows to drive the data exchange, minimizing human-interactions and hiding the system complexity by exploiting a new data provisioning service based on the publish-subscribe model.

Keywords — Sensing, Cloud, PaaS, Data Provisioning, Publish-Subscribe Model, SWE, Big Data.

I. INTRODUCTION

A new generation of embedded devices provide an opportunity to create new business and social models by exploiting a strong interaction with the environment. They capture information from the physical world, processing and forward their observations towards remote management systems. This information travels along heterogeneous components, such as routers, databases, information systems and the Internet. In turn, this leads in the generation and movement of enormous amounts of data which have to be stored, processed and presented in a seamless, efficient and easily interpretable form. Cloud computing represents a very flexible technology, that can efficiently support services for sensing data provisioning, due to its ability in offering theoretically unlimited computing and storage capabilities, and efficient communication services for transferring terabyte flows between data centers [8]. Thus, the research community is showing great interest in developing Cloud platforms for supporting sensing infrastructures, useful in many application scenarios, such as smart cities [6][7], healthcare [10], homeland security [9] and so on.

This paper presents a new Cloud architecture for the retrieval of data gathered from several heterogeneous sensing infrastructures deployed in wide geographical areas. It is able to provide data according to an event-based approaches, based on the *Publish-Subscribe* model. The design of the proposed architecture is compliant with the Sensor Web Enablement (SWE) standard defined by the Open Geospatial Consortium. In the OGC-SWE framework [14], the SAS (Sensor Alert Service) specifications describe the interfaces for publishing and subscribing alerts coming from sensors. We have implemented our data provisioning service extending the interfaces of the SAS with functionalities for heterogeneous infrastructures integration, data abstraction and Big Data storage. The main contribution of this work is the strong decoupling of the activities of different entities that make use of the data provisioning service (e.g., users, applications, data providers, sensing devices,...). In fact, the Cloud hides the complexity of the whole system to all the involved stockholders.

The paper is organized as follows. Section II. discusses the most recent works on Cloud computing and sensing technology challenges. The proposed data provisioning service is described in Section III.. In Section IV., we present our Cloud architecture, discussing its design and the features of the provided service. In Section V., we discuss the current implementation of the service. Section VI. concludes our dissertation.

II. RELATED WORKS

In this paper, we make use of Cloud resources and services for providing a scalable remote management of sensing data. In this direction, in [5], the authors describe a context-oriented data acquisition and integration platform for Internet of Things over a Cloud computing environment. The work well discusses the representation of information based on an XLM-oriented approach, but it is not clear enough how they leverage the Cloud resources.

As stated in [11], ubiquitous sensor networks may enable sensing data collection at many points and places in real-time. Usually, collected data are saved on data storage systems, such as PCs, database servers or Web servers. The authors proposed the utilization of a free

ISBN:9788896515204

Cloud data service, such as Twitter, exploited to share observed data. They introduced the concept of ambient sensor Cloud system by using both Cloud computing and the Arduino-based Open Field Server platform. People may benefit of these data accessing them by Twitter as Followers.

Another example of Cloud-based Social Platforms for sensing is reported in [15]. The initiative is aimed to investigate the *mood* of users. Here the authors extrapolate mood information from social feeds (e.g., Twitter) and associate to them sensing data (e.g., weather). The mathematical model applied to a *mood space* tries to characterize users, their day-lives and tweets. Their assessment is conducted in off-line computation mode, and doesn't deal with massive computation constrains.

The authors in [16] have designed a Sensor-Cloud infrastructure enabling end-users to dynamically create virtual sensor groups. The introduction of *virtual items* (e.g., sensors, sensor groups, servers, etc.) is interesting indeed, but how they impact on the management of Cloud resources is not very clear. They use conventional SQL-based database to collect data, but this choice may represents a bottleneck in the architecture in terms of performance.

The Cloud data management service reported in [2] aims at massive sensing data management in the Cloud and is based on the Hadoop framework. The authors assert that traditional relational database management systems are a bottleneck in scenarios dealing with ultra-large-scale data sets. Thus, their Cloud data center adopts the Hadoop FileSystem (HDFS) for organizing clusters aimed at the Cloud.

A new architecture in which Server Webs and Server Sensors interact each others using a redirectable stream-oriented channels is described in [12]. The authors propose a quite complex framework, but using SSH tunnels for transferring sensing data files among many FTP servers is not a very flexible and scalable approach.

III. DATA PROVISIONING SERVICE

The scientific contribution of this paper is the design and implementation of a Cloud service for the provisioning of sensing data. To this aim, a new Cloud agent has been developed to implement and extend the SWE-SAS interfaces, that we call *SASAgent*.

The main task of the SASAgent is to provide a platform to meet the requirements of Data Consumers (DCs), which need environmental information to develop advanced services, such as statistical analysis, cross relation of data, data mining, and so on, with data supplied through different and heterogeneous sensing infrastructures, organized in Sensor Networks (SNs) or manged by Sensing Data Providers (SDPs). To offer a very efficient service, the SASAgent has to implement a flexible communication system to interconnect all the involved entities. It has to guarantees a seamless access to data offered by many SNs/SDPs for DCs. At the same time, each SN/SDP must be able to satisfy at the same time requests of several DCs. Therefore, the SASAgent has to strongly decouple the activities of both DCs and SNs/SDPs. To reach these goals, we have designed our Cloud service for data provisioning as an event notification system based on the publish-subscribe model.

A sensing infrastructure (e.g. SDP1 shown in Figure 1) advertises its offers to the DCs whenever they are available through the Cloud. The *Advertise* message includes a unique identifier for the offer, the feature of interest, the operation area and other optional information (such as unit of measurement, frequency in data sending, expiration of the advertise,...). Then, it starts sending available data(see Figure 2).

The sensing offers are stored by the SASAgent into the Cloud and, as shown in Figure 1, submitted to DCs as *Subscription Offerings* whenever a DC asks for the capabilities of one or more SDPs (*GetCapabilities* message).

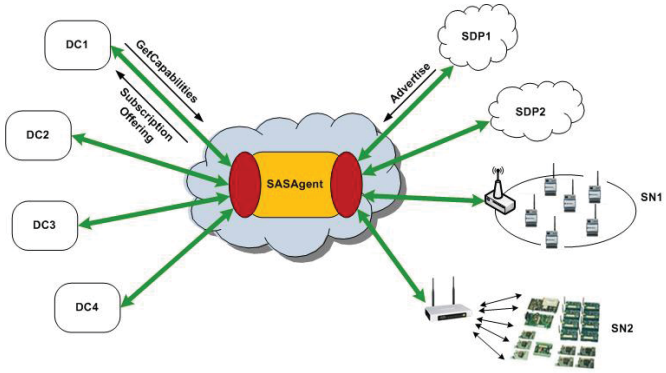

Fig. 1: Publish functionalities implemented by the SASAgent

If a DC is interested in one or more subscriptions, subscribes them (see Figure 2) by specifying their unique identifiers, the operational area and optionally filtering operations on data (such as thresholds, data aggregation, frequency rate of updates,...). From that time onwards, it will receive data related to the subscription agreement in a XMPP Multi-User Chat (MUC) room. Our system benefits of the advantages of the XMPP communication. In fact, XMPP (Extensible Messaging and Presence Protocol) [1] has been developed to drive communications in heterogeneous instant messaging systems (e.g., GTalk, WhatsApp, etc.), where it is possible to convey any type of data. XMPP is able to offer the following features: decentralized service, high degree of scalability, high number of hosts involved, flexibility in the system, interoperability and native security features based on TLS/SSL and XML-based encryption/signing. Also, through the MUC, several DCs may access sensing data related to the same subscription at the same time.

Summarizing, the main advantages of the SASAgent are: 1) Decoupling: publishers (that are SNs and SDPs)

ISBN:9788896515204 141 4th IMEKO TC19 SYMPOSIUM

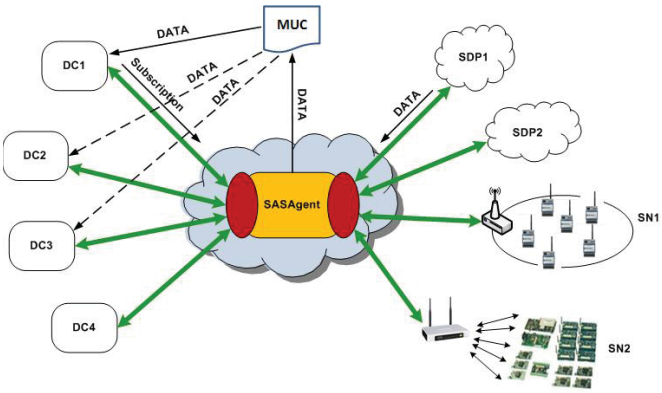

Fig. 2: Data access

are uncoupled from the subscriber since they do not need to know about their presence or requirements. At the same time, subscribers (that are DCs) can ignore the hardware and software infrastructure of the sensing system, taking advantages only from the logical communication channels. 2) Scalability: the publish-subscribe model improves the scalability of the traditional client-server approach, thanks to the parallel management of involved entities, chaching messages, shared rooms for data access. Moreover, the Cloud based implementation of the SAS Agent offers additional scalability features thanks to XMPP communications, distributed solutions and hierarchical logic infrastructures.

IV. CLOUD ARCHITECTURE

According to the NIST formalization [13], Cloud services are offered at three different levels: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). This paper is focused on the PaaS level, which exploits the virtualization of physical resources at the IaaS level to expose services for the SaaS layer. In this work, we have adopted a middleware for a scalable integration of heterogeneous and distributed resources at IaaS level developed at the University of Messina and named CLEVER [3] to build our sensing data provisioning service. In this section we provide a brief description of CLEVER and, next we explain how the SASAgent works within CLEVER.

A. *CLEVER at a glance*

CLEVER (CLoud-Enabled Virtual EnviRonment) is a Virtual Infrastructure Manager (VIM) able to manage Virtual Machines (VMs) in IaaS environments. It is based on a distributed clustered architecture, where each cluster belonging to a datacenter, is organized in two hierarchical layers, as depicted in Figure 3. All the CLEVER nodes supply an host level management module, called Host Manager (HM) and just one node includes a cluster level management module, called Cluster Manager (CM). A CM acts as interface between Cloud clients (end-users/applications exploiting the Cloud) and software agents running on HMs. The CM receives commands from clients, gives instructions to the HMs, elaborates information and finally sends results to clients. It also performs tasks for the management of Cloud resources and the monitoring of the working state of the cluster. Summarizing, the CM contains the intelligence for managing the cluster and implementing high-level services. On the contrary, HMs have low-level functionalities and work as peripheral components of the physical infrastructure of the Cloud. Even if at least one CM is active in the cluster, in order to ensure higher fault tolerance, some redundant CMs remain in a listening state to detect if the active CM turns off.

Software components running in the CM and HMs are called *agents*. Specifically, we refer to an agent on the CM as Cluster Agent (CA) and an agent on HMs as Host Agent (HA).

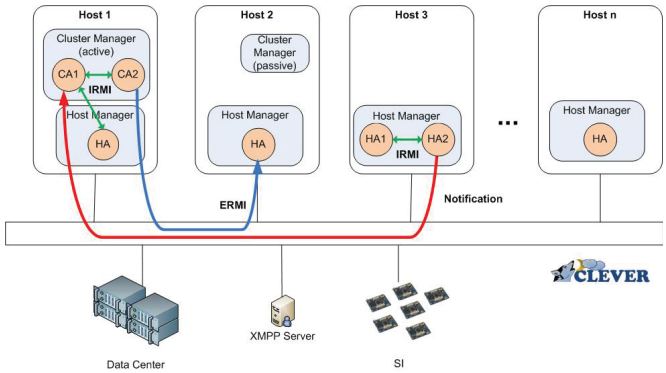

Fig. 3: Distributed organization of CLEVER components

CLEVER supports three types of communication among agents: Internal Remote Method Invoker (IRMI), External Remote Method Invoker (ERMI) and Notification. An IRMI communication refers to the message exchanging protocol among agents within the same manager (both in CMs and HMs). Since agents are separated processes running on the same host, IRMI communications are based on Inter Process Communications (IPCs). ERMI communications allow a CA to exchange messages with HAs. In fact, each CA has knowledge of all the agents in the HMs that depend on the CM itself and it can, also, request for a reply from HAs. ERMI communications are based on the XMPP protocol. Unlike CAs, HAs do not have knowledge of the CAs at the upper-layer of the hierarchy. Even if this design choice seems to limit the interoperability among the agents, it allows to reduce the complexity of inter-module communications and to increase scalability, fault-tolerance and availability of the system. To allows communications from HAs to CAs, CLEVER makes use of Notifications, which are sent from an HA to its CM and does not request for any reply and does

ISBN:9788896515204

not specify the CAs interested in the notification. The sorting of notifications among CAs is demanded to the CM. As the ERMI, also Notifications are based on the XMPP protocols, in order to benefit of flexibility and versatility in communications.

CLEVER has been designed with an eye toward horizontal federation. In a federation, each organization is typically independent and may not be altered by a unilateral decision of a "central government". Cloud federation offers two substantial benefits to Cloud providers. First, it allows providers to earn revenue from resources that would otherwise be idle or underutilized. Second, providers expand their geographic footprints and accommodate sudden spikes in demand without having to build new points-of-presence [4].

B. *SASAgent within CLEVER*

The CLEVER architecture has been extended for offering the sensing data provisioning service through the implementation of a new CLEVER agent, called SASAgent, as shown in Figure 4.

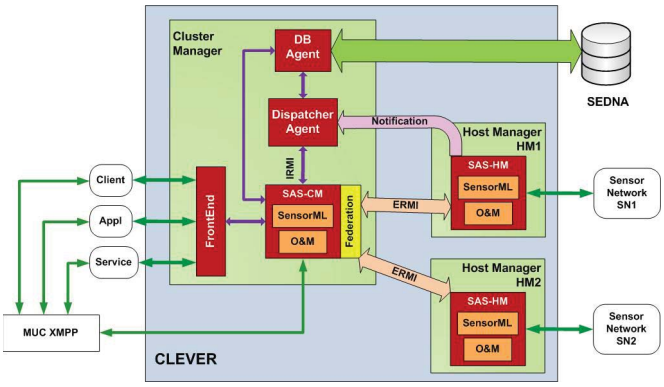

Fig. 4: Implementation of the SAS service in CLEVER

At the higher level, the SASAgent in the CM (SAS-CA) provides sensed data to DCs according to the publish-subscribe model discussed in Section III.. In the CLEVER architecture, also CAs and HAs interact following the publish-subscribe model, where the Dispatcher Agent is the broker that receives advertises and data from an HM and passes them as notifications to the interested agents in the CM. This approach allows many agents to receive the same notifications (for example, if they implement different services by using the same data set). At the startup, a SAS-CA registers its presence in the Dispacher registry. The Dispatcher registry includes all the agents able to receive notifications. Then, the SAS-CA subscribes at the Dispatcher Agent one or more advertises coming from HMs and lies in wait state.

The low-level functionalities of the SASAgent are implemented in HMs and support the interaction with heterogeneous sensing infrastructures. Thus, a SASAgent in a HM (SAS-HA) is activated for each SDP/SN. The SAS-HA has to supply the service with the description of sensors and observations, but it is not responsible for data formatting[1]. When a new sensing system starts working (eg. SN1 in Figure 4), it provides a description of its features (number of nodes, types of sensing, feature of interest,...) through an XML document drawn according to the SensorML format. As soon as the SAS-HA receives the advertise, it sends a Notification to the CM through the Dispatcher Agent, in order to prepare a Subscription. Then, the SAS-HA will forward all the data it receives from the SN to the CM through notifications. The SAS-CA pulls the XML document from the Dispatcher Agent, assigns a unique identifier to it and stores it in the *Subscription table*.

When a DC connects the Cloud, asks the SAS-CA for available subscriptions through the FrontEnd. If the DC subscribes one or more observations, in turn, the SAS-CM subscribes at the Dispatcher Agent all the notifications on the observations related to the subscription. For example, if the Subscription include two sensors able to measure temperature and pressure, the SAS-CA subscribes notifications from each sensor of the SN and for both temperature and pressure data.

Whenever the SAS-CM receives a notification, it processes the information held in the message to detect if a meaningful event is occurred and, eventually, forwards data in the MUC. The typology of detectable events depends on the particular type of sensed data or subscription. For example, an event can be notified if the sensing measurement is over a specific threshold or events can be sensing observations forwarded periodically, with a frequency specified at the time of the subscription.

To realize a very flexible system, many sensor networks deployed in the environment and managed from different administration can be aggregated in the Cloud by a Federation. Each sensor network is virtualized in the Cloud through a specific SAS-HA, but several SAS-HAs can be placed in one or more HMs. To merge data from all the SAS-HAs, the SAS-CA uses the Federation module, which is able to distinguish different data sources, manages them through polling policies and offers a seamless service to the higher layer. At the same time, many SAS-CAs can be present in the same CM, offering different services according to specific sensed data (e.g. car traffic analysis, temperature measurements, weather forecast,...).

The proposed architecture has been designed to offer also the additional service of on-demand data access, which allows users to ask for sensing data produced in a specific geographical area and in a limited time interval. Users make queries through the FrontEnd and the SAS-CA forwards them towards the DB Agent, that is responsible for interacting with a DBMS (DataBase Management System). In our architecture, the XML-based open-source SEDNA database has been used. It simplifies the data storage along with the queries to perform the retrieval of XML-based data using XPath and

[1] In the SWE-OGC framework, specific standards deal with this task, such as SOS (Sensor Observation Service)

ISBN:9788896515204 143 4th IMEKO TC19 SYMPOSIUM

XQuery.

V. SERVICE PROTOTYPE

API Method	Description
PhenomenonAdvertise	it offers the possibility to announce the type of information you publish.
CancelAdvertisement	it allows you to delete an advertisement run in previously.
GetObservation	to ask for sensing data
RenewAdvertisement	to renew an advertisement previously run.
ExpirationAdvertise	if the an advertisement has not renewed recently, the SASAgent may ask for a RenewAdvertise

Tab. 1: SAS-HM API

API Method	Description
GetCapabilities	it allows a DC to request a description of the capabilities of the specific server SAS.
Subscribe	it allows DCs to subscribe to the alert data.
CancelSubscription	it allows DCs to delete an active subscription.
RenewSubscription	to renew a subscription that is expiring.

Tab. 2: SAS-CM API

The SASAgent can be exploited by using its XMPP Java APIs. In particular, it specifies two sets of API. The first set allows SNs and SDPs to announce and publish actual measured data and observations. It is implemented in the SAS-HM and includes the methods listed in Table 1. The second set of API allows the interaction with DCs and are included in the SAS-CM. The methods already implemented in our prototype are listed in Table 2.

To test the behavior of the proposed architecture, we have developed a web site, which interacts with our Cloud by using the above API specifications. It includes XMPPHP, that is a XMPP client library for PHP/javascript, and offers to the end-users an easy access to the main functionalities of the system.

In the testbed, we have emulated two sensors (see Figure 5) able to perform several observations in a geographical area chosen at random. The subscription offering page is shown in Figure 6. Subscriptions are characterized by a unique identifier (e.g. Soi-642718737), the organization that provide data and a brief description of the observation. If the end-user subscribes one or more offerings, he/she can access data through the *Monitoring* section. Figure 7 shows how data are handed out in case of the subscription of temperature and pressure observations. Each row identifies an observation, specifying the type of observation, the measured value and the timestamp. On the right, some statistics are provided to the end-users.

Fig. 5: Emulated sensors in the testbed

Fig. 6: Subscription offering

VI. CONCLUSIONS

In this paper we have presented an innovative Cloud architecture able to deal with a huge amount of sensing data. We have discussed the advantages in using Cloud technologies in the management of sensed data, where sensing infrastructures can be managed by different companies and administrations. Data consumers get such data subscribing the service regardless any knowledge on the real source of data and hardware/software infrastructures deployed to provide them. We have also presented the CLEVER middleware, which fills the gap related to how data has to be treated in the Cloud, reducing communication, processing and storage costs. A new module of CLEVER, the SASAgent has been introduced. In particular the set of functionalities of the SASAgent has been described in detail.

ISBN:9788896515204

Fig. 7: Sensing data

VII. ACKNOWLEDGEMENTS

The work reported in this paper has been supported by the the project POR-FESR Sicily 2007-2013 SIMONE (Integrated System for monitoring the production of Electricity). We wish to thank Francesco Manera, who provided valuable assistance in the development of the system prototype.

VIII. REFERENCES

[1] The Extensible Messaging and Presence Protocol (XMPP) protocol: http://tools.ietf.org/html/rfc3920.

[2] Y. Bao, L. Ren, L. Zhang, X. Zhang, and Y. Luo. Massive sensor data management framework in cloud manufacturing based on hadoop. In *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, pages 397 –401, july 2012.

[3] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. Integration of CLEVER Clouds with Third Party Software Systems Through a REST Web Service Interface. In *17th IEEE Symposium on Computers and Communication (ISCC'12)*, 1-4 July 2012.

[4] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. Energy Sustainability in Cooperating Clouds. In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science (CLOSER 2013)*, Aachen, Germany, 8-10 May 2013.

[5] Y.-S. Chen and Y.-R. Chen. Context-oriented data acquisition and integration platform for internet of things. In *Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on*, pages 103 –108, nov. 2012.

[6] S. Dey, A. Chakraborty, S. Naskar, and P. Misra. Smart city surveillance: Leveraging benefits of cloud data stores. In *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, pages 868 –876, oct. 2012.

[7] M. Fazio, M. Paone, A. Puliafito, and M. Villari. Heterogeneous sensors become homogeneous things in smart cities. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 775 –780, July 2012.

[8] M. Fazio, M. Paone, A. Puliafito, and M. Villari. Huge amount of heterogeneous sensed data needs the cloud. In *International Multi-Conference on Systems, Signals and Devices, SSD 2012 - Summary Proceedings*, 2012.

[9] M. Fazio, M. Paone, A. Puliafito, and M. Villari. HSCloud: Cloud Architecture For Supporting Homeland Security. *International Journal on Smart Sensing and Intelligent Systems*, 5(1):246–276, March 2012.

[10] V. Gupta. Cloud computing in healthcare. Cisco Knowledge Network, September 16 2011. http://www.cisco.com/web/IN/about/network/cloud_computing.html.

[11] M. e. A. Hirafuji. Creating high-performance/low-cost ambient sensor cloud system using openfs (open field server) for high-throughput phenotyping. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pages 2090 –2092, sept. 2011.

[12] J. Melchor and M. Fukuda. A design of flexible data channels for sensor-cloud integration. In *Systems Engineering (ICSEng), 2011 21st International Conference on*, pages 251 –256, aug. 2011.

[13] NIST. Cloud Computing Reference Architecture http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505 December 2011.

[14] C. Reed, M. Botts, J. Davidson, and G. Percivall. OGC Sensor Web Enablement: Overview and High Level Architecture. *IEEE Autotestcon*, pages 372–380, 2007.

[15] S. Yerva, H. Jeung, and K. Aberer. Cloud based social and sensor data fusion. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 2494 –2501, july 2012.

[16] M. Yuriyama, T. Kushida, and M. Itakura. A new model of accelerating service innovation with sensor-cloud infrastructure. In *SRII Global Conference (SRII), 2011 Annual*, pages 308 –314, 29 2011-april 2 2011.