IJDCST @Sep-Oct, Issue- V-2, I-8, SW-25
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Incorporation of Meta Cloud in CSPs to Solve Vendor Lock-In Problems

Shaik Ismail Jabeebulla, S. Rama Krishna,
M-tech Student Scholar, Department of Computer Science Engineering, VRS & YRN College of Engineering & Technology, Chirala;Prakasam (Dt); Andhra Pradesh, India.
Assistant Professor &H.O.D,Department of Computer Science Engineering,VRS & YRN College of Engineering &Technology, Chirala; Prakasam (Dt), Andhra Pradesh, India.

Abstract — The cloud computing paradigm has achieved widespread adoption in recent years. Its success is due largely to customers' ability to use services on demand with a pay-as-you go pricing model, which has proved convenient in many respects. Low costs and high flexibility make migrating to the cloud compelling. Despite its obvious advantages, however, many companies hesitate to "move to the cloud," mainly because of concerns related to service availability, data lock-in, and legal uncertainties.1 Lock in is particularly problematic. For one thing, even though public cloud availability is generally high, outages still occur.2 Businesses locked into such a cloud are essentially at a standstill until the cloud is back online. Moreover, public cloud providers generally don't guarantee particular service level agreements (SLAs)3 — that is, businesses locked into a cloud have no guarantees that it will continue to provide the required quality of service (QoS). Finally, most public cloud providers' terms of service let that provider unilaterally change pricing at any time. Hence, a business locked into a cloud has no mid- or long term control over its own IT costs. At the core of all these problems, we can identify a need for businesses to permanently monitor the cloud they're using and be able to rapidly "change horses" — that is, migrate to a different cloud if they discover problems or if their estimates predict future issues.

We aim to restore the accidentally lost passwords and file keys of the data user. The existing system has the options to upload file by the data owner, verification of data by a Trusted Third Party and Cloud Service Provider and file retrieval by the data user by appropriate and secure authentication mechanisms but does not have an option to restore the accidentally lost passwords or restore and resend the file keys for file retrieval. In this enhancement, we incorporate the ways the restore the lost passwords and regenerate the file keys in case of emergency situations by adding an additional level of authentication. This helps to retrieve files in case of emergencies.

Index Terms— Data storage, privacy-preserving, public audit ability, cryptographic protocols, cloud computing.

I. INTRODUCTION

The cloud computing paradigm has achieved widespread adoption in recent years. Its success is due largely to customers' ability to use services on demand with a pay-as-you go pricing model, which has proved convenient in many respects. Low costs and high flexibility make migrating to the cloud compelling. Despite its obvious advantages, however, many com-panies hesitate to "move to the cloud," mainly because of concerns related to service availabil-ity, data lock-in, and legal uncertainties.[1] Lock-in is particularly problematic. For one thing, even though public cloud availability is gener-ally high, outages still occur.[2] Businesses locked into such a cloud are essentially at a standstill until the cloud is back online. Moreover, \public cloud providers generally don't guarantee\ par-ticular service level agreements (SLAs)[3] — that is, businesses locked into a cloud have no guaran-tees that it will continue to provide the required quality of service (QoS). Finally, most public cloud providers' terms of service let that provider unilaterally change pricing at any time. Hence, a business locked into a cloud has no mid- or long-term control over its own IT costs.

At the core of all these problems, we can identify a need for businesses to permanently monitor the cloud they're using and be able to rapidly "change horses" — that is, migrate to a different cloud if they

discover problems or if their estimates predict future issues. However, migration is currently far from trivial. Myriad cloud providers are flooding the market with a confusing body of services, including compute services such as the Amazon Elastic Compute Cloud (EC2) and VMware vCloud, or key-value stores, such as the Amazon Simple Storage Service (S3). Some of these services are conceptually comparable to each other, whereas others are vastly different, but they're all, ultimately, technically incompatible and follow no standards but their own. To further complicate the situation, many companies not (only) build on public clouds for their cloud computing needs, but combine public offerings with their own private clouds, leading to so-called hybrid cloud setups.[4]

Here, we introduce the concept of a meta cloud that incorporates design time and runtime components. This meta cloud would abstract away from existing offerings' technical incompat-ibilities, thus mitigating vendor lock-in. It helps users find the right set of cloud services for a particular use case and supports an application's initial deployment and runtime migration.

II. RELATED WORK

CLOUD COMPUTING USE CASE

Let's consider a Web-based sports portal for an event such as the Olympic Games, which allows users to place bets. An event this large requires an enormously efficient and reliable infrastructure, and the cloud computing paradigm provides the necessary flexibility and elasticit y for such a scenario. It lets service providers handle short-term usage spikes without needing res-pective dedicated resources available continuously. The problem, however, is that once an application has been developed based on one particular provider's cloud services and using its specific API, that application is bound to that provider; deploying it on another cloud would usually require completely redesigning and rewriting it. Such vendor lock-in leads to strong dependence on the cloud service operator. In the sports portal example, in addition to the ability to scale applications up and down by dynamically allocating and releasing resources, we must consider additional aspects, such as resource costs and regional communication bandwidth and latency.

Let's assume the sports betting portal application is based on a load balancer that forwards HTTP requests to numerous computing nodes hosting a Web application that lets users submit a bet. Request handlers place bet records in a message queue and subsequently store them in a relational database. Let's further assume a service provider realizes this scenario using only Amazon Web Services (AWS), EC2 to host applications, Simple Queue Service (SQS) as its cloud message queue, and the Relational Database Service (RDS) as a database system. Instead of being bound to one cloud operator, however, the betting application should be hosted in an optimal cloud environment.

CURRENT WEATHER IN THE (META) CLOUD

First, standardized programming APIs must enable developers to create cloud-neutral applications that aren't hardwired to any single provider or cloud service. Cloud provider abstraction libraries such as libcloud (http:// libcloud.apache.org), fog (http://fog. io), and jclouds (www.jclouds.org) provide unified APIs for accessing different vendors' cloud products. Using these libraries, developers are relieved of technological vendor lock-in because they can switch cloud providers for their applications with relatively low overhead.

As a second ingredient, the meta cloud uses resource -templates to define concrete features that the application requires from the cloud. For instance, an applica-tion must be able to specify that it requires a given number of com-puting resources, Internet access, and database storage. Some current tools and initiatives — for example, Amazon's-CloudFormation (http:// aws.amazon.com/cloudformation/) or the upcoming TOSCA specification (www.oasis-open.org/committees/ tosca) — are working toward similar goals and can be adapted to provide these required features for the meta cloud.

In addition to resource templates, the automated formation and pro-visioning of cloud applications

also depends on sophisticated features to actually deploy and install applica-tions automatically. Predictable and controlled application deployment is a central issue for cost-effective and efficient deployments in the cloud, and even more so for the meta cloud. Several application provisioning solutions exist, enabling developers and administrators to declaratively specify deployment artifacts and dependencies to allow for repeatable and managed resource provisioning. Notable examples include Opscode Chef (w w w.opscode.com/chef/), -Puppet (http://puppetlabs.com), and juju (http://juju.ubuntu.com).

At runtime, an important aspect of the meta cloud is application monitoring, which enables the meta cloud to decide whether it's nec-essary to provision new instances of the application or migrate parts of it. Various vendors provide tools for cloud monitoring, ranging from system-level monitoring (such as CPU and bandwith) to application-level monitoring (Amazon's CloudWatch; http://aws.amazon.com/cloudwatch/) to SLA monitoring (as with moni-tis; http://portal.monitis.com/index. php/cloud-monitoring). However, the meta cloud requires more sophisticated monitoring techniques and, in particular, approaches for making automated provisioning decisions at runtime based on cur-rent application users' context and location.

INSIDE THE META CLOUD

To some extent, we can realize the meta cloud based on a combination of existing tools and concepts, part of which we just examined. Figure 1 depicts the meta cloud's main components. We can categorize these components based on whether they're important mainly for cloud software engineers during development time or whether they perform tasks during runtime. We illustrate their interplay using the sports betting portal example.

META CLOUD API

The meta cloud API provides a unified programming interface to abstract from the differences among provider API implementations. For customers, using t his API prevents t heir application from being hard-wired to a specific cloud service offering. The meta cloud API can build on available cloud provider abstraction APIs, as previously mentioned. Although these deal mostly with key-value stores and compute services, in principle, all services can be covered that are abstract enough for more than one provider to offer and whose specific APIs don't differ too much, conceptually.[6]

In the existing system, there are options to upload file by the data owner, verification of data by a Trusted Third Party and Cloud Service Provider and file retrieval by the data user by appropriate and secure authentication mechanisms but does not have an option to restore the accidentally lost passwords or restore and resend the file keys for file retrieval. The enhancement for this project aims to restore the accidentally lost passwords and file keys of the data user. We incorporate the ways the restore the lost passwords and regenerate the file keys in case of emergency situations by adding an additional level of authentication like asking the user to answer a secret question at the time of registration. By appropriate answering of the secret key, we can either block the user or allow the user to download the files in case of emergencies.

RESOURCE TEMPLATES

Developers describe the cloud ser-vices necessary to run an application- using resource templates. They can specify service types with additional- properties, and a graph model expresses the interrelation and functional dependencies between services-. Developers create the meta cloud resource templates using a simple domain-specific language (DSL), letting them concisely specify required resources. Resource defi-nitions are based on a hierarchical composition model; thus developers can create configurable and reusable template components, which enable them and their teams to share and reuse common resource templates-

in different projects. Using the DSL, developers model their application components and their basic

-runtime requirements, such as (provider-independently normalized) CPU, memor y, and I/O capacities, as well as dependencies and weighted communication relations between these components. The provision-ing strategy uses the weighted com-ponent relations to determine the application's optimal deployment configuration. Moreover, resource templates allow developers to define constraints based on costs, compo-nent proximity, and geographical distribution.

MIGRATION AND DEPLOYMENT RECIPES

Deployment recipes are an important ingredient for automation in the meta cloud infrastructure. Such recipes allow for controlled deployment of the application, including installing packages, starting required services, managing package and applica-tion parameters, and establishing links between related components. Automation- tools such as Opscode Chef provide an extensive set of functionalities that are directly inte-grated into the meta cloud environ-ment. Migration recipes go one step further and describe how to migrate an application during runtime — for example, migrate storage function-ality from one service provider to another. Recipes only describe ini-tial deployment and migration; the provisioning strategy and the meta cloud proxy execute the actual pro-cess using the aforementioned auto-mation tools.

META CLOUD PROXY

The meta cloud provides proxy objects, which are deployed with the application and run on the provi-sioned cloud resources. They serve as mediators between the application and the cloud provider. These prox-ies expose the meta cloud API to the application, transform application requests into cloud-provider-specific requests, and forward them to the respective cloud services. Proxies provide a way to execute deployment and migration recipes triggered by the meta cloud's provisioning strat-egy. Moreover, proxy objects send QoS statistics to the resource moni-toring component running within the meta cloud. The meta cloud obtains the data by intercepting the applica-tion's calls to the underlying cloud services and measuring their pro-cessing time, or by executing short benchmark programs.

III. EXISTING SYSTEM

Cloud providers are flooding the market with a confusing body of services, including computer services such as the Amazon Elastic Compute Cloud (EC2) and VMware v Cloud, or key-value stores, such as the Amazon Simple Storage Service (S3). Some of these services are conceptually comparable to each other, whereas others are vastly different, but they're all, ultimately, technically incompatible and follow no standards but their own. To further complicate the situation, many companies not (only) build on public clouds for their cloud computing needs, but combine public offerings with their own private clouds, leading to so-called hybrid clouds.

IV. PROPOSED SYSTEM

Here, we introduce the concept of a meta cloud that incorporates design time and runtime components. This meta cloud would abstract away from existing offerings' technical incompatibilities, thus mitigating vendor lock-in. It helps users find the right set of cloud services for a particular use case and supports an application's initial deployment and runtime migration.

We aim to restore the accidentally lost passwords and file keys of the data user. We incorporate the ways the restore the lost passwords and regenerate the file keys in case of emergency situations by adding an additional level of authentication like asking the user to answer a secret question at the time of registration. By appropriate answering of the secret key, we can either block the user or allow the user to download the files in case of emergencies.

V. CONCLUSION

The meta cloud can help mitigate vendor lock-in and promises transparent use of cloud computing services. Most of the basic technologies necessary to realize the meta cloud already exist, yet lack integration. Thus, integrating these state-of-the-art tools promises a huge leap toward the meta cloud. To avoid meta cloud lock-in, the community must drive the ideas and create a truly open meta cloud with

added value for all customers and broad support for different providers and implementation technologies.

By implementing the above discussed, additional level of authentication mechanism, we are able to enable the user to download the files shared by the owner in case of emergencies in a secure way.

REFERENCES

[1] M. Armbrust et al., "A View of Cloud Computing," *Comm. ACM*, vol. 53, no. 4, 2010, pp. 50–58.

[2] B.P. Rimal, E. Choi, and I. Lumb, "A Tax-onomy and Survey of Cloud Computing Systems," *Proc. Int'l Conf. Networked Com-puting and Advanced Information Man-agement*, IEEE CS Press, 2009, pp. 44–51.

[3] J. Skene, D.D. Lamanna, and W. Emmerich, "Precise Service Level Agreements,"- *Proc. 26th Int'l Conf. Software- Eng.* (ICSE 04), IEEE CS Press, 2004, pp. 179–188.

[4] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-of-the-Art and Research Challenges," *J. Internet Services and Applications*, vol. 1, no. 1, 2010, pp. 7–18.

[5] M.D. Dikaiakos, A. Katsifodimos, and G. Pallis, "Minersoft: Software Retrieval in Grid and Cloud Computing Infrastruc-tures," *ACM Trans. Internet Technology*, vol. 12, no. 1, 2012, pp. 2:1–2:34.

[6] Winds of Change from Vendor Lock-In to the Meta Cloud, IEEE INTERNET COMPUTING VOL:17 NO:1 YEAR 2013

AUTHOR'S PROFILE

ISMAIL JABEEBULLA. SHAIK received B.Tech degree from Malineni Lakshmaiaha Engineering & Technology, Kanumalla, Singarayakonda, Praksam, Andhra Pradesh. And currently pursuing M.Tech in Computer Science Engineering at VRS & YRN College of Engineering &Technology, Chirala, Prakasam (Dt), Andhra Pradesh. His areas of interest include Cloud computing.

Mr. S. Rama Krishna is presently working as Associate professor and Head of CSE Department in VRS & YRN College of Engineering & Technology, Chirala, AP, India. He completed his B.Tech degree in Computer Science and Engineering from JNTUH, Hyderabad. And then completed his M.Tech. in Computer Science and Engineering as his specialization from JNTUH, Hyderabad. Now he is pursuing his Ph.D Degree in Computer Science and Engineering from JNTUH, Hyderabad. His research areas include Cloud Computing and Security. He has a teaching experience of 10 years. He published papers in 3 International Journals and 1 National Conference.