Cloud Changing Paradigm with Meta Cloud

Kurangi Sudhakar¹, M. Venkateswarao²

¹M.Tech Student, in the Dept of IT, GITAM University, Visakhapatnam, AP., India
²Professor in the Dept of IT, GITAM University, Visakhapatnam, AP., India

Abstract— It is a techno-business disruptive model for, which is using large scale distributed data centers. These are either private or public or hybrid offering customers a scalable virtualized infrastructure or an abstracted set of services which are qualified by the service level agreements (SLA’s), and changed only by the academic IT resources consumed. The presence of yet more and more cloud involvement from a crowd of service provider’s call for a Meta cloud to straighten the edges of the pointed cloud background. This Meta cloud could solve hawker lock-in problems that current public and hybrid cloud communities face. Naturally public cloud service providers may not guaranteed for provide essential quality of service (EQOS) and also unconditional changing in pricing at any time. Businesses locked into a cloud have no promises that it will continue to provide the essential quality of services (EQOS).

As a final point, most public cloud provider’s terms of services saying that providers are unconditionally changing prices at any time. Hence a business locked into a cloud having no control over on mid-long term IT costs at the fundamental of these difficulties, we can classify a need for businesses to eternally monitor the cloud they have using and be able to hastily “swap to the horses” that means migrate to a different cloud if they faced any problems with their cloud or if their estimates speculated future issues. And also we concern security purpose leads to migrate one cloud to another cloud to mitigate the hawker lock-in problems.

Keywords— Cloud computing, resource patterns, migration and deployment recipes, hawker lock-in, Meta cloud API, Meta cloud proxy.

I. INTRODUCTION

Due to the principally to customer’s ability to use services on demand with a pay-as-you go pricing model, which has proved that cloud computing has achieved frequent implementation in recent years. Migrating to the cloud compliments leads to low level costs and high flexibility although it’s understandable advantages. However, many companies are quiver to “move to the cloud” mainly because of worries related to the service availability, data lock-in, expected cost savings, better efficiency and time to market(EQOS), increased security and greater storage capacity and also increased control over on SLA’s with the legal suspicious.

Most of the cloud service provider’s terms of services let that unconditionally changing in prices at any time. Hence businesses locked into a cloud have no control over on mid-long term IT costs.

1. Data lock-in is complicated for one thing, even though public cloud availability is naturally high and also outages still occur.
2. Businesses locked into such a cloud are fundamentally at a standstill until the cloud is return back to online moreover, public cloud providers may not guaranteed for particular service level agreements (SLA’s). That means businesses locked into a cloud have no promises that it will continue process to provide the Essential quality of services (EQOS).

At the fundamental of these difficulties, we can classify a necessity for businesses to eternally monitor the cloud. They are using and are able to hastily: “swap to the horses”—that means migrate to a different cloud if they facing problems with that or if their estimates speculate future issues. However migration is currently far from insignificant many clouds. Cloud service providers are swamping to the market with an uncertain or confused body of services, including the computer services such as the HP hellion public cloud, Amazon elastic compute cloud (EC2) and VMware Vcloud, or key-value stores, such as the Amazon simple storage services(S3). Some of these services are conceptually comparable to each other, where as other are infinitely different. But they are all ultimate, technically incomputable and they are following no standards but their own. Some of the un-paralleled clouds computing service providers are providing streamlines to the existing IT infrastructure. For example net magic-leading cloud computing service provider in India. While a cloud migration can present so many challenges are raised with 1.privacy & security (for the concern of security there is an encryption and decryption algorithm placed) 2. Reliability and availability 3.transaction and execution risks 4.limited scope for customization 5.cultural resistance 6.regulatory ambiguity 7.issues of taxation and reduce capital expenditures and operating costs while also benefiting from the dynamic scaling, high availability and effective resource allocation.
Meta cloud defines abstract way for mitigating to hawker lock-in and also provides many advantages like Pay-as-you go model for IT resource consumed, virtual and on-demand, agility- flexibility and elasticity, multi-tenancy, ease of implementation, pooled resources.

II. CURRENT WEATHER IN THE CLOUD (META CLOUD)

First of all we need to systematized programming API’s which enable developers to make cloud-neutral applications that are not hardwired to any single provider or any single cloud service. Cloud provider extraction libraries such as dasein (http://www.dasein.org), Dzone(http://www.java.dzone.com), celarcloud(http://www.celarcloud.eu), libcloud(http://libcloud.apache.org), github(http://github.com) provides unified API’s for accessing different hawker cloud products. By using these libraries, developers are relieved of technological Hawkers lock-in because they can switch cloud providers for their applications with comparatively overhead. as per the second ingredient the meta cloud uses resource templates to define wide features for that application requires from the cloud. For instant, an application must be able to notify that it requires a given number of computing resources, internet accessing, and database storage. Some current tools and initiatives for example Amazon cloud formation or the upcoming TOSCA specification (www.oasis-open.org/committees/Tosca) are working towards similar goals and can be adapted to provide these essential features for the Meta cloud. In addition to resource templates, the automated formation and provisioning of cloud applications also depends on sophisticated features actually deploy and install applications automatically. Foreseeable and controlled application deployment is a central issue for cost-effective and efficient deployment is in the cloud, even more so far the Meta cloud. Several applications provisioning solutions exist, enabling developers and dependences to allow for repeatable and managed resources provisioning. Notable examples are include Opscode Chef (www.opscode.com/chef), Zyrion (www.Zyrion.com), Puppet (http://puppetlabs.com), and juju (http://juju.ubuntu.com), at run time, an important aspect of the meta cloud is application monitoring. Application monitoring enable the Meta cloud to decide whether it is necessary to pro-vision of new instances of the application or migrate parts of it.

Various hawker’s provide tools for cloud monitoring, ranging from system-level monitoring such as CPU and bandwidth to application level monitoring. Amazon cloud watch (http://aws.amazon.com/cloud watch/) to SLA monitoring with moony-tis(http://portal.moonytis.com/index.php/), however, the meta cloud requires more sophisticated monitoring techniques and in particular approaches for making automated pro-visioning decisions at runtime based on current application user’s context and location.

III. META CLOUD PROXY

Meta cloud proxy’s serve as intermediaries between application and the cloud service providers. The Meta cloud provides the proxy objects, which are deployed with the application and run on the fitted cloud resources. These proxies are exhibits the Meta cloud API to remodel application requests into cloud-provider-specific requests and forward this to the respective cloud services. Proxies provide a way to execute deployment and migration recipes triggered by the Meta cloud’s fitting strategy. More over proxy objects send essential quality of services measurements to the resources monitoring component running among the Meta cloud.

The Meta cloud obtains the data by intercepting the application’s calls to the underlying cloud services and calculating process time, or by capital punishment short benchmark programs. Application can also define and monitor custom EQOS metrics that the proxy objects sends to the resource monitoring component to enable ‘advanced application-specific management schemes.

To avoid heavy load and computational blockages communication between proxies and the Meta cloud is maintain at a minimum proxies don’t run inside the Meta cloud. Regular service calls from the application to the proxy aren’t routed through the Meta cloud, other.
3.1 What is inside the Meta cloud?

To some extent, we can realize the Meta cloud based on combination of existing tools and concepts. Part of which we just observed fig 1. Dispose the Meta cloud body i.e. components. Insides the Meta cloud we can classify these components, based on whether they are important mainly for cloud software engineers during the development time or whether they are perform tasks during the runtime. We demonstrate their relationship using betting portal example.

3.2 Meta cloud API:

Meta cloud API provides a unified programming interface between differences among provider API operations. By using this API prevents their applications from being hardwired to a specific cloud service offering. The Meta cloud API can also build on available cloud provider abstraction API’s, as previously mentioned. While this 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 who’s explicit API’s don’t differ too much conceptually.

3.3 Resource patterns:

Developers describe the cloud services necessary to run an application using resource patterns. They can specify service types with additional properties, and a graph model expresses the inter-relation and functional dependences between various services. A developer creates the Meta cloud resource patterns using a simple domain-specific language (DSL). Letting them briefly specify required resources. Resource definitions are based on a hierarchal composition model; then developer can create configurable and reusable pattern components, which enable them and their teams to share and reuse common resource patterns in different projects. Developer’s model their application components and their basic runtime requirements, such as (provider-independent, normalized) CPU, memory, and I/O capacities as well as dependencies and weighted communication relations between these components.

The fitting strategy uses the weighted component relation to determine the application optimal deployment configuration. Moreover, resource pattern are allowed to developers to define constraints based on costs, component proximity and geographical distribution.

IV. MIGRATION AND DEPLOYMENT RECIPES

Deployment recipes are important for automation in the Meta cloud infrastructure. Such recipes cloud applications using Meta cloud development mechanisms.

The Meta cloud runtime extracts from provider particulars using proxy objects and automates application life cycle management.

Meta cloud migration allows for controlled deployment of the application including installed packages, starting required services, managing packages and application parameters and establishing links between related components. Migration recipes go one step further and describe how to migrate an application during runtime for example, migrate storage functionality from one service provider to another service provider. Recipes only describe initial deployment and migration; the fitting strategy and the Meta cloud proxy execute the actual process using the aforementioned automation tools.

4.1 A typical cloud computing use case:

Let’s assume a web-based sports portal for an event, which requires an enormously efficient and reliable infrastructure. Such as Olympic Games, cloud computing paradigm provides the necessary flexibility and elasticity for such scenario. It helps to service providers handle short term usage spikes without needing corresponding dedicated resources available often. However the problem is that, once an application has been developed based on particular service provider’s cloud and also using its specific API that application could bounded to that provider only. Deploying this application to another cloud would usually require overall rewriting and redesigning. Such hawker lock-in leads to fully dependence n cloud service operator. In the sports betting portal example in addition to the ability to scale applications by dynamically allocating resources and releasing resources. We must consider additional aspects, such as costs of resources and regional communication bandwidth and latency.

Let us consider the web based sports betting portal application is based on a lead balancer that sends HTTP requests to computing nodes hosting a web application. That lets the user to submit the bets.
Their requests are handled by the request handler and request handler places bet records in to message queue and subsequently stores them in relational database. For the further consideration of a service provider proven this scenario using only Amazon web service (AWS), EC2 to host application, simple queue services(SQS) as it’s cloud message queue and relational database system instead of being bounded to particular cloud operator. The betting portal should migrate in an optimal cloud environment. To leverage more diversity on cloud landscape support flexibility, and also avoid hawker’s lock-in problem.

4.2 Meta cloud must achieve two main goals

1. Find the optimal combination of cloud services for a certain application with regard to essential quality of services for users and price for hosting.
2. Develop a cloud-based application once and then run it anywhere including support for runtime migration. Slowly the Meta cloud idea has receiving some attention and several approaches try to trickle at least some parts of the problem.

4.3 proposed algorithm

4.3.1 Encryption

Byte State [4, Nb]
State = in
AddRoundKey(State, KeySchedule[0,Nb-1])
For Round= 1 step 1 to Nr-1
{} 
SubBytes(State)
ShiftRows(State)
Mixcolumns(State)
AddRoundKey(State, KeySchedule[Round*Nb,(Round+1)*Nb-1])
}
SubBytes(State)
ShiftRows(State)
AddRoundKey(state, keySchedule[Nr*Nb,(Nr+1)*Nb-1])
out= state

4.3.2 Decryption

Byte State[4,nb]
State = in
AddRoundKey(State keySchedule[Nr*Nb,(Nr+1)*Nb-1])
For Round = Nr-1 step-1 down to 1
{}
InvShiftRows(State)
InvSubBytes(State)
AddRoundKey(State, KeySchedule[Round*Nb,(Round+1)*Nb-1])
InvMixColumns(State)
}
InvShiftRows(State)
InvSubBytes(State)
AddRoundKey(State, KeySchedule[0,Nb-1])
Out=state.

4.4 Resource monitoring:

The resource monitoring component collects the data from the Meta cloud proxies about the resources, they are using on an application’s request. The resource monitoring component filters and processes these data and forwards to the knowledge base. Knowledge base stores them for further processing. This helps to generate comprehensive quality of service information about cloud service providers and the specific services they provide, including the response time, flexibility, availability and more service-specific quality statements.

4.5 Knowledge base:

Knowledge base stores all the information which is necessary to migration and data about cloud provider service, their price and QoS. Knowledge base stores the estimated migration costs and also stores customer provided resource templates and migration or deployment recipes. The knowledge base suggests which cloud providers are eligible for a certain customer. These usually comprise all providers the customer has an account with and providers that other possibilities for creating sub accounts on the fly. Several information sources contribute to the knowledge base: A Meta cloud proxy continuously sends data about application natures and cloud service QoS. Users can add cloud service providers, pricing and availability, capabilities manually or use crawling technique that can get this information automatically.
V. A META CLOUD USE CASE

Let’s come back to the sports portal use case. A Meta cloud compliant variant of this application accesses cloud services using the meta cloud API and doesn’t meet to the cloud-provider-specific service API’s. In our particular case, the application doesn’t depend on Amazon EC2,SQS, or RDS service API’s, but rather on the meta cloud’s compute, message queue, and relational database service API’s.

For initial deployment, the developer submits the application’s resources template to the meta cloud. And it specifies not only the three types of cloud services required to run the sports portal application, but also their necessary properties and how they depend on each other. For compute resources, the developer can specify CPU, RAM, and disk space according to terminology defined by Meta cloud resource template, which acts as a reference during deployment, runtime, and migration.

The resource template specification should also contain inter-dependencies such as the direct connection between the web service compute instances and the message queue services. Rich information that resource templates help fitting strategy component makes profound decisions about cloud service grading.

We can explain the working principle for initial deployment with a web crawling analogy, in which resource templates are queries and cloud service provider QoS and pricing information represent indexed documents. Algorithmic aspects of the actual grading are beyond this articles range.

If some template resources in the resource graph are only loosely coupled, then the meta cloud will be more interest to select resources from different cloud providers for a single application. Coming to our use case, however, we assume that fitting strategy grades the corresponding Amazon cloud services first, and the customer follows this recommendation. After the resources are specified, the meta cloud deploys the application, together with an instance of the meta cloud proxy, according to customer-recommended recipes.

During the runtime, the meta cloud proxy acts as mediator between the application components and the Amazon cloud resources and sends monitoring data to the resources monitor component which is running with in the meta cloud. Monitoring data helps process the application’s resource template and the provider’s overall QoS values, both stored in the knowledgebase. The fitting strategy component regularly checks this updated information, which might trigger to migration.

The Meta cloud could migrate front-end nodes to other providers to make them closer to the application’s users, for example.

Another reason for a migration might be updated pricing data. After price cut by rack space, for example, services might migrate to its cloud offerings. To make these decisions, the fitting strategies component must consider potential migration costs regarding time and money. The cloud migration is performed based on customer-recommended migration recipes. By working on meta cloud we face the following technical challenges. Resource monitoring must collect and process the data describing different cloud provider’s services such that fitting strategy can compare and grade their QoS properties in a normalized provider-independent manner.

Meta cloud infrastructure responsibilities

- Finding the balance between migration facilities provided by the Meta cloud and the application.
- Meta cloud responsible for resource monitoring which are taken from the resource monitoring template, we are using.
- Meta cloud infrastructure also responsible for fitting strategy that means providing essential QOS.

We argue that the Meta cloud should control the migration process but offers many interception points for applications to influence the process at all the time. The fitting strategies mainly based on input from runtime monitoring and resource template and effects them by executing migration and deployment recipes. Further research into clustered approaches from the information backup, and automatic computing fields.

VI. CONCLUSIONS

The Meta cloud can help mitigate hawker lock-in problems and promises apparent use of cloud computing services. Most of the basic technologies need to realize the Meta cloud already exists yet require combination. If we are facing any problems with the cloud service provider then we can migrate to another cloud service provider. By migrating one CSP to another CSP leads to pay-as-you go model. Meta cloud can provides multi-tenancy, ease of implementation, resource monitoring which helps us for flexibility. 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.
REFERENCES


Authors Profile

Mr. Kurangi Sudhakar is a P.G Student in the Department of IT in, GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT, Visakhapatnam and A.P.

Mail: k.sudhakar558@gmail.com

Mr. Mandapati Venkateswararao is a professor in the Department of IT in, GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT, Visakhapatnam and A.P.

mandapati_venkat@yahoo.co.in