International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 1, January 2015, PP 13-17
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Vendor Lock in Winds Relocation to Meta Cloud

Ravi Kumar.K[1], K. Bheemalingappa[2]MTech

[1]PG Scholar, CSE, BITS, AP, India
[2]Assistant Professor, CSE, BITS, AP, India

Abstract: *The emergence of more and more cloud offerings from a multitude of service providers calls for a meta cloud, which smoothens the jagged cloud landscape. We discuss our proposal for such a meta cloud, and explain how it solves the lock-in problems that current users of public and hybrid clouds face. The cloud computing pattern has gained widespread adoption in recent years. It was success due to large to customers' ability to utilize the responsibilities as per requirement with a pay-as-you go worth model, which has confirmed as acceptable in many aspects. Less expenditure and max flexibility make progress around to the cloud convincing. In spite of its obvious flexibilities, though, a lot companies pause to shift to the cloud mostly cause of concerns associated to service accessibility, data lock-in, and legal doubts.*

1. INTRODUCTION

To some point, we can understand the Meta cloud based on a grouping of existing tools and concept, part of which we just examine. Figure 1 depicts the Meta cloud's main components. We can arrange these components based on whether they're important generally for cloud software developers throughout expansion time or whether they execute tasks throughout runtime. We explain their interaction utilizing the games gambling portal for a simple example. The Meta cloud API gives a combined programming interface to summary from the difference among source implementations of API. For users, utilizing this Application Program Interface prevent their request from being typically-wired to a particular cloud service submission. The API of Meta cloud can develop on available source cloud provider abstraction APIs, as previously mentioned. Even these deals mostly with the key value stores and computer services, in standard, all services can be covered that are theoretical more than one service to offer and whose specific APIs don't differ too much, theoretically.

Resource template engineers explain the cloud services required to process an application utilizing resource templates. They can identify service categories with extra properties, and a model of graph explores the functional and interrelation dependency between services. Developers create the Meta cloud reserve templates utilizing a plain DSL (domain-specific language), hire them in a few words specify necessary resources. store definitions are based on a kind of masterpiece model; thus engineers can develop reusable and configurable template components, which use them and their groups to reuse and share general resource templates in various projects.

Utilizing the domain-specific language, engineers model components of their application and their necessary runtime needs, like as memory, I/O capacities, and CPU, as well as weighted and dependency communication between these components. The provision strategy uses the relations of subjective component to conclude the application's optimal deployment configuration. Moreover, resource template allows engineers to describe restrictions depending upon expenditures, geographical distribution and component proximity. We should accept the games wagering entry application is focused around a heap balancer that advances HTTP solicitations to various registering hubs facilitating a Web application that lets clients submit a wager. Demand handlers put down wager records in a message line and in this way store them in a social database.

We should further accept an administration supplier understands this situation utilizing just Amazon Web Services (AWS), Ec2 to have applications, Simple Queue Service (SQS) as its cloud message line, and the Relational Database Service (RDS) as a database framework. As

opposed to being certain to one cloud administrator, on the other hand the wagering application ought to be facilitated in an ideal nature's turf. To influence a more various cloud scene, help flexibility, and dodge merchant lock-in, the meta cloud must attain two primary objectives: find the ideal combo of cloud administrations for a certain application concerning Qos for clients and cost for facilitating; and develop a cloud-based application once, then run it anyplace, including backing for runtime movement. Of late, the meta cloud thought has gotten some consideration, and a few methodologies attempt to handle at any rate parts of the issue.

2. A Typical Cloud Computing Use Case

The BBC has recently reported1 on their technology strategy for operating the Web portal of the 2012 Olympic Games. An event of this dimension requires an enormously e_cient and reliable infrastructure and the cloud computing paradigm provides exibility and elasticity that is needed for such a scenario. It allows handling short-term usage spikes without the need to have respective dedicated re- sources available all the time. The problem is, however, that once an application has been developed based on cloud services of one particular provider using its specific API it is bound to the provider; deploying the application on another cloud would usually require completely redesigning and rewriting it.

This vendor lock-in leads to strong dependence on the cloud service operator. It the example of the sports portal, besides the ability to scale applications up and down by dynamically allocating/releasing resources, additional aspects such as resource costs and regional communication bandwidth and latency have to be considered. Let us assume the sports betting portal application is based on a load balancer that forwards HTTP requests to a number of computing nodes hosting a web application, which allows users to submit a bet. Bet records are put into a message queue and subsequently stored into a relational database. Using Amazon AWS cloud services, this scenario can be realized using EC2 to host applications, SQS as cloud message queue, and _nally RDS as database system. Instead of being bound to one cloud operator, the betting application should be hosted on the optimal cloud environment.

In order to leverage a more diverse cloud landscape, to support exibility, and to avoid vendor lock-in, the following main goals need to be achieved by the meta cloud.

(1) Find the optimal combination of cloud services for a certain application with regard to QoS for the users and price for hosting.

(2) Develop a cloud-based application once, then run it anywhere, including sup- port for runtime migration. Lately, the meta cloud idea has already received some attention and several approaches try to tackle at least parts of the problem. In the next section we will briey discuss currently available solutions.

3. Inside the Meta Cloud

To some extent, the meta cloud can be realized based on a combination of existing tools and concepts, part of which are presented in the previous section.

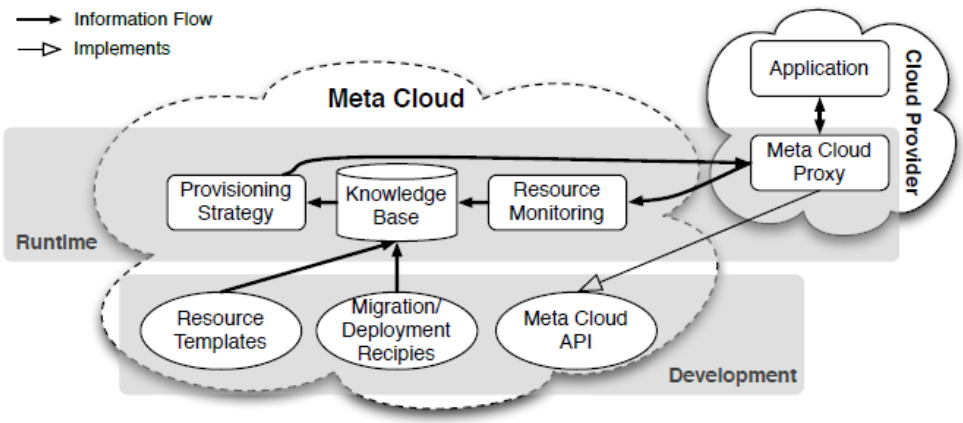

Figure 1. Conceptual Overview of the Meta Cloud.

The main components of the meta cloud, depicted in Figure 1, are described in the following and their interplay is illustrated using the previously introduced sports betting portal example. The components of the meta cloud can be distinguished whether they are mainly important for cloud software engineers during development time or whether they perform tasks during runtime.

3.1. Meta Cloud API

The meta cloud API gives a brought together programming interface to extract from the distinctions among supplier API usage. For clients, utilizing this API keeps their application from being hard-wired to a particular cloud administration advertising. The meta cloud API can expand on accessible cloud supplier reflection Apis, as formerly said. Despite the fact that these arrangement basically with key value stores and register administrations, on a fundamental level, all administrations can be secured that are unique enough for more than one supplier to offer and whose particular Apis don't vary excessively, thoughtfully.

3.2. Resource Templates

Engineers depict the cloud administrations important to run an application utilizing asset formats. They can point out administration sorts with extra properties, and a chart model communicates the interrelation and utilitarian conditions between administrations. Designers make the meta cloud asset formats utilizing a basic space particular dialect (DSL), giving them a chance to compactly indicate obliged assets. Asset definitions are focused around a progressive structure model; hence engineers can make configurable and reusable format segments, which empower them and their groups to impart and reuse normal asset layouts in diverse ventures.

Utilizing the DSL, designers demonstrate their application segments and their essential runtime prerequisites, for example, (supplier autonomously standardized) CPU, memory, and I/O limits, and conditions and weighted correspondence relations between these parts.

The provisioning procedure utilizes the weighted segment relations to focus the application's ideal arrangement design. Besides, asset layouts permit designers to characterize

obligations focused around expenses, segment roximity, and topographical dispersion.

3.3. Migration and Deployment Recipes

Sending formulas are an imperative element for computerization in the meta cloud base. Such formulas take into account controlled arrangement of the application, including introducing bundles, beginning obliged administrations, overseeing bundle and application parameters, and making connections between related segments. Computerization devices, for example, Opscode

Chef give a far reaching set of functionalities that are specifically coordinated into the meta nature's domain. Relocation formulas go above and beyond and portray how to relocate an application amid runtime — for instance, move stockpiling usefulness starting with one administration supplier then onto the next. Formulas just portray starting arrangement and relocation; the provisioning methodology and the meta cloud substitute execute the genuine

procedure utilizing the previously stated computerization instrument

3.4. Meta Cloud Proxy

The meta cloud gives substitute items, which are sent with the application and run on the provisioned cloud assets. They serve as middle people between the application and the cloud supplier. These substitutes uncover the meta cloud API to the application, change application demands into cloud-supplier particular solicitations, and forward them to the individual cloud administrations. Substitutes give an approach to execute sending and relocation formulas activated by the meta cloud's provisioning methodology. Additionally, substitute items send Qos facts to the asset checking segment running inside the meta cloud. The meta cloud gets the information by blocking the application's calls to the underlying cloud administrations and measuring their preparing time, or by executing short benchmark programs.

Applications can likewise characterize and screen custom Qos measurements that the substitute articles send to the asset checking segment to empower progressed, application-particular administration methodologies. To evade high load and computational bottlenecks, correspondence in the middle of substitutes and the meta cloud is kept at any rate. Substitutes don't run inside the

meta cloud, and customary administration calls from the application to the substitute aren't steered through the meta cloud, either.

3.5. Resource Monitoring

On an application's ask for, the asset checking part gets information gathered by meta cloud substitutes about the assets they're utilizing. The segment channels and procedures these information and afterward stores them on the learning base for further preparing. This aides create extensive Qos data about cloud administration suppliers and the specific administrations they give, including reaction time, accessibility, and more administration particular quality explanations.

3.6. Provisioning Strategy

The provisioning technique segment basically matches an application's cloud administration prerequisites to genuine cloud administration suppliers. It discovers and positions cloud administrations focused around information in the learning base. The introductory sending choice is focused around the asset layouts, detailing the asset necessities of an application, together with Qos and evaluating data about administration suppliers. The result is a rundown of conceivable cloud administration blends positioned as per expected Qos and expenses. At runtime, the part can reason about whether moving an asset to an alternate asset supplier is advantageous focused around new bits of knowledge into the application's conduct and upgraded cloud supplier Qos or evaluating information. Thinking about moving additionally includes computing relocation costs. Choices about the provisioning methodology bring about the part executing client characterized arrangement or movement scripts.

3.7. Knowledge Base

The learning base stores information about cloud supplier benefits, their estimating and Qos, and data important to gauge relocation costs. It likewise stores client gave asset formats and movement or arrangement formulas. The learning base shows which cloud suppliers are qualified for a certain client. These normally embody all suppliers the client has a record with and suppliers that offer potential outcomes for making (sub)accounts on the fly. A few data sources help the information base: meta cloud substitutes frequently send information about application conduct and cloud administration Qos. Clients can include cloud administration

suppliers' evaluating and capacities physically or use slithering methods that can get this data consequently.

4. A META CLOUD USE CASE

Let us come back to the previously introduced sports application use case. A meta cloud compliant variant of this application accesses cloud services using the meta cloud API, and does not directly talk to the cloud provider specic service APIs. For the particular case this means the application does not depend on Amazon's EC2, SQS, or RDS service APIs, but on meta cloud's compute, message queue, and relational database service APIs. For initial deployment the developer submits the application's resource template to the meta cloud. It species not only the three types of cloud services needed for run- ning the sports application, but also their necessary properties and how they depend on each other. For compute resources, for instance, CPU, RAM, and disk space can be specified, according to terminology de_ned by the meta cloud resource template DSL. Each resource can be named in the template, which allows for referencing dur- ing deployment, runtime, and migration. The resource template speci_cation should also contain interdependencies, like the direct connection between the web service compute instances and the message queue service.

The rich information provided by resource templates helps provisioning strategy component to make profound decisions about cloud service ranking. The working principle for initial deployment can be explained by web search analogy, in which resource templates are queries, cloud service provider QoS and pricing information represent indexed documents. Algorithmic aspects of the actual ranking are beyond the scope of this article. If some resources in the resource graph are only loosely coupled, then it is more likely that resources from di_erent cloud providers may be selected for a single application. In our use case, however, we assume that the

provisioning strategy ranks the respective Amazon cloud services _rst, and that the customer follows this recommendation.

After the resources are determined the application together with an instance of the meta cloud proxy is deployed, according to customer provided recipes. During runtime, the meta cloud proxy mediates between the application components and the Amazon cloud resources, and sends monitoring data to the resource monitoring component running within the meta cloud.

REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Gri_th, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. Commun. ACM, 53(4):50{58, April 2010.

[2] Marios D. Dikaiakos, Asterios Katsifodimos, and George Pallis. Minersoft: Software retrieval in grid and cloud computing infrastructures. ACM Transactions on Internet Technology (ACM TOIT), 12(1), July 2012.

[3] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A Taxonomy and Survey of Cloud Comput- ing Systems. In International Conference on Networked Computing and Advanced Information Management, pages 44 51, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[4] James Skene, D. Davide Lamanna, and Wolfgang Emmerich. Precise Service Level Agreements. In Proceedings of the 26th International Conference on Software Engineering (ICSE'04), pages 179{188, Washington, DC, USA, 2004. IEEE Computer Society.

[5] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. J. Internet Services and Applications, 1(1):7{18, 2010.