MCloud API For Managing Data in Multi-Cloud Environment

Maram Alhafdi, Ahmed Barnawi
malhafzy0001@stu.kau.edu.sa, ambarnawi@kau.edu.sa
1 Department of Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Abstract - Cloud computing environment is one of the widely growing software deployment platforms in the world of technology. There is a continuous increase in using the cloud-based storage as service from developers and users. Cloud storage services provide numerous advantages such as high scalability, availability and pay-as-you-go cost model. In addition, cloud providers are offering different options for storage services. For example, Amazon provides S3 service as a scalable, durable and available distributed object store. Azure provides SQL databases as traditional SQL databases. In this domain, one of the biggest challenges is the interoperability between different storage systems from the various cloud providers, due to the lack of the unified methods for accessing, interfacing and managing the stored data. In this paper, we present our approach to address this challenge by building a system that enables the users and developers to manage their data hosted in different storage systems from the various cloud providers for a single point of interface. Our prototype is under evaluation.

Keywords: Cloud computing, data management, interoperability, vendor lock-in, multi-cloud, and storage systems.

1 Introduction

Cloud computing has become one of the hottest core technical topics in the modern software development era. The National Institute of Standards and Technology defined the cloud computing as "a model for enabling convenient, resource pooling, ubiquitous, on-demand access which can be easily delivered with different types of service provider interaction" [1]. Cloud storage is one of the most essential services of cloud computing, which offers storage-as-a-service that allows data owners to store their data in the cloud[2]. In practice, four different data storage and access mechanisms are common across the cloud vendor [3]:

- Blob storage also known as object storage is useful for storing unstructured data. Examples are Amazon S3 and IBM Cloud Object Storage.
- Table storage is non-relational databases called "NoSQL." An example of this is Amazon DynamoDB
- Queue storage is a dedicated queue storage mechanism, often used for fully managed message queuing service. For example, Amazon SQS
- Relational table storage is based on traditional SQL databases. For example, Amazon RDS, and Azure SQL databases

Different cloud providers such as Amazon, and Microsoft Azure, supported the previous storage systems and offered them to their customers. The user should use the front end or API to get access to or store the data. Unfortunately, those cloud providers incompatible with each other and created these services with different APIs. Therefore, the incompatibility in standards and formats while getting access to the cloud has become a big issue and causes vendor lock-in problem [4]. Consequently, one of the major challenges in cloud computing environment is cloud interoperability. Interoperability is defined as the ability of heterogeneous systems to work and interact together. For clouds, interoperability means the ability for multiple cloud providers to work together with minimal or null user effort. However, most of the cloud providers built their services without consideration of interoperability [5][6].

Some developers and users are more closely to use a particular storage type of a specific provider due to the features the provider is offering or the pricing of this service. For that, the user hosted his data on multi-cloud. When the user wants to access and manage his data, he will deal with different cloud interfaces and then he will face a problem in managing this hosted data due to vendor lock-in. Moreover, they face a lack of interoperability among these different cloud providers, and the unified methods for accessing, interfacing, and managing the stored data in these storage systems.
To address these challenges, we proposed building a unified interface (MCloud) to manage the data hosted in different storage systems in multi-cloud. MCloud standardizes APIs for Blob storage, Table storage, Queue storage, and Relational table storage, to allow developers and users to be independent from cloud vendors and to provide a transparent way for the users to access and manage their data. For Blob storage, MCloud supports Amazon Simple Storage Service (Amazon S3) and IBM Bluemix Object Storage. In addition, it supports DynamoDB for Table storage, and Amazon Simple Queue Service (SQS) for the Queue storage type. Finally, it supports Relational Database Service (RDS) from Amazon provider, and Azure SQL Database for Relational table storage.

The remainder of the paper is organized as follows. In the following section, we discuss the related work. In section 3, we explain the system design which we follow to implement MCloud. We present MCloud API, and the services it provides in section 4. In Section 5, we present the future work and conclusion.

2 Related Work

In the past, several researches about the unified cloud interfaces were published whether as independent ones or as a part of a broader approach. A majority of them focused on the infrastructure provisioning model [4], or the unified interfaces for application deployment and management among cloud platforms. Furthermore, existing approaches for PaaS focused on supporting a unified deployment of applications. In addition, the recent researches have focused more on the management capabilities for applications in the cloud such as developing, deploying and migrating multiple data stores. The following paragraphs provide an overview of the related work and demonstrate how our work differs and contributes to the existing approaches. Table 1 is a summary of related works and its limitations.

Several standardization organizations propose standards to overcome the vendor-in lock, and to deal with the data stored in a cloud environment. Storage Network Industry Association (SNIA) proposed Cloud Data Management Interface (CDMI), which is an ISO/IEC standard that enables cloud solution vendors to meet the growing need of interoperability for data stored in the cloud [7]. Open Cloud Computing Interface classified as a standardized approach for Open Cloud Computing Interface. It provides a set of specifications for cloud tasks such as deployment, dynamic scaling, and monitoring across different cloud providers. However, most standard proposals suffer from the lack of acceptance and participation by cloud providers [8].

Another approach to integrating multiple CSPs is to use proxy servers, which act as an intermediary among multiple CSPs providing transparent access, and gathering data from multiple CSPs, however, this approach is a single point of failure [9]. For example, CDMI-compatible proxy supports only two types of storage mentioned in section 1, blobs and queues storages [3]. Some applications like Cloudfuze, which is a centralized interface, are used to manage the files stored in different cloud service providers [10]. In addition, CYRUS [9] is a client-defined architecture that integrates multiple CSPs into one unified cloud and allows clients to share files and specify their desired performance levels. But these applications just support blob storage type.

SimpleCloud is an API that allows using the storage services independent of cloud platforms [8]. It is similar to our approach, but it allows the developers to write the portable code that can interoperate with multiple cloud vendors, however it only supports PHP language for web applications. MCloud provides a common interface to access and manage the hosted data in the multiple cloud providers, regardless of the programming language the developers use to build their applications. CDPort [6] proposed a common data model and a standardized API for NoSQL databases. Moreover, CDPort supported the transformation and exchange of data that is stored on NoSQL databases. It provided only NoSQL storage type.

Some researches focused on assisting the developers to manage their applications. Such as [11] which facilitates the developer’s task and enables the development of applications using multiple data stores. The developer can use this approach to develop, deploy and migrate multiple data stores applications in cloud environments. A unified interface for application deployment and management among cloud platforms is presented in [4]. Openshift is a platform for developers to build, test, deploy and run cloud applications; it supports no-lock-in at PaaS level as approach in [4]. By using Openshift, the developer can focus only on designing and coding, whereas all the infrastructure and middleware management are handled by Openshift [8]. MCloud supports no-lock-in at SaaS level.

All these studies in the previous paragraph provide a way to assist the developers to manage their applications. Our research focused on supporting the developers to manage the data hosted in a cloud environment; it provides storage as a service to the developers. Therefore, a developer can access the storage from different cloud providers, provision it, view its details and manage it.
3 System Design

The aim of our interface is to unify the core management functions of the data hosted in multi-cloud. We focus solely on the creation of a management interface, which covers all storage types that mentioned in section 1. MCloud is using a layer for wrapping three SDKs of different clouds into a common API to get access to the data hosted in different storage systems as shown in Figure 1.

To achieve our goals, we built the interface with Python language using the SDKs for some cloud providers. Firstly, using Boto3 SDK provided by Amazon to enable MCloud to support four services: Amazon Simple Storage Service (s3), DynamoDB (NoSQL database), Relational Database Service (RDS), and Simple Queue Service (SQS). Secondly, using Azure SDK provided by Microsoft Azure to enable MCloud to support Azure SQL Database. Thirdly, using the Swiftclient SDK to enable the users to access the Object Storage service that is provided by IBM Bluemix.

MCloud is easy to install and setup, so it is useful to the end-user who cares about using blob storage from different cloud providers and needs a unified interface to manage his/her data. Furthermore, MCloud is helpful for the developer who cares about provision and uses a different storage system from various cloud providers.

4 MCloud API

MCloud is a common API for delivering different storage services over a multi-cloud. This platform supports all types of storage in cloud computing environment (blob storage, queue storage, table storage, and relational table storage). MCloud supports the interoperability among cloud providers in a transparent way, especially for the user who works with different storage systems from various cloud providers.

MCloud supports the four-different data storage and their services as the following:

1. Blob storage: MCloud supports two of the most common blob storage: Amazon Simple Storage Service (Amazon S3) and IBM Bluemix Object Storage. MCloud supported the operations of creating a bucket, emptying it, deleting it, and viewing its contents. It also facilitates the control over the bucket to upload files in it, download files from it, and delete files from it. Finally, the API also provides options to create and delete folders.

2. Table storage: MCloud facilitates the control over Amazon DynamoDB. API allows to create a table, delete it, and insert items in it. In addition, it supports the operations over items in the table; the user can update the item, delete it, and query it.

3. Queue storage: MCloud enables the user to get control over Amazon Simple Queue Service (SQS). Moreover, it supports delete queues, view queue details, purge queues, configure queues, and view the messages. In addition, it supports the operations of creating permissions, and deleting them. Finally,
the API also provides options for sending a message, and deleting it.

4. Relational table storage: MCloud supports two of the most common Relational table storages: Amazon Relational Database Service, and Azure SQL Database. MCloud facilitates control over these services. For Amazon(RDS), the user can create an instance, view its details, modify it, and delete it. For Azure SQL, API allows to create (SQL database, resource group, and server) and delete them. Finally, the API also provides options to view the databases on a server, view items of a resource group, and set server firewall.

5 Conclusion

As we mentioned, there are several solutions aiming to give transparent access to multiple storage systems. However, one solution cannot achieve all user requirements. There is still a problem in managing the data hosted in different storage services of various cloud providers. In addition, there is a lack of the unified methods for accessing, and managing this stored data. Therefore, we provide an approach to unify the interfaces for the storage services provided by various cloud providers. Our prototype is under evaluation. In future work, we aim to add more services which provide billing information about the storage the user used, and an approach for selecting the best storage type depending on the user requirements and the storage prices.

6 References
