ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 3; Issue 6; November-December-2015; Page No. 30-33

An enhanced approach to support multi data stores applications in cloud environment

Gouri S. Joshi[1], Gayatri Bhandari[2]

[1]JSPM's Bhivarabai Sawant Institute of Technology and Research, Wagholi Pune 412 207

joshigouri198@gmail.com

[2]JSPM's Bhivarabai Sawant Institute of Technology and Research, Wagholi Pune 412 207

gayatri.bhandari1980@gmail.com

Abstract

In early days generation of large amount of data and the rise of cloud computing have introduced new aspects for data management. In this paper we reviewed integrated set of models, algorithms and tools pointing at relieving developers' goal for developing, deploying and migrating multi data stores applications in cloud atmosphere. An approach focuses mainly on First, a unifying data model followed by applications developers to communicate with heterogeneous relational and NoSQL data stores. Depending on that, queries are expressed using OPEN-PaaS-Data Base API (ODBAPI), unique REST API granting programmers to write their code independently of the target data stores. Second, virtual data stores act as a mediator and communicate with integrated data stores covered by ODBAPI.

Key Words: REST-based API, NoSQL, data stores, rational data stores

Introduction

Cloud computing has recently risen as a new computing paradigm which empower on-demand and it has scalable provision of resources. It also has platforms and software as services. Cloud computing is divided into three levels [1]: 1. Infrastructure as a Service (IaaS) provides access to the abstracted view on the hardware, 2, the Platform-as-a-Service (PaaS) supplies programming and execution environments to the developers, and 3, the Software as a Service (SaaS) enabling software applications to be used by cloud's end users.

Cloud computing adds execution environments for some emerging applications like big data management due to its elasticity property. In this paper the variety property [2] of big data is mainly focused and more precisely on multiple data store based applications in the cloud.

To satisfy variety of storage requirements, cloud applications requires accessing and interacting with several relational and NoSQL data stores having heterogeneous APIs of the data stores which induces problems while constructing, deploying and migrating multiple data store applications. Main four problems are:

Pb_1:
Elephantine workload on the developer: These days data stores have heterogeneous and variety of APIs. Developers of multiple data store based applications need to be known all these APIs while coding their applications

Pb_2:
No declarative way to execute complex queries: Heterogeneity of the data models has any declarative way to define and execute complex queries over several data stores. This is mainly due to the absence of a global schema of heterogeneous data stores. In addition, NoSQL data stores are scheme less. It means developers should have to manage with the implementation of such complex queries.

Pb_3:
Code adaptation: Application developers need to re-adapt the application source code to interact with new data stores when applications are migrating from one cloud environment to another. Developers should have potential to learn and use new APIs.

Pb_4:
Tedious and non-standard processes of discovery and deployment: Once an application is developed or migrated, developers need to spread it into cloud provider. Discovering the most appropriate cloud

Corresponding author: Gouri S. Joshi

environment which can provide required data stores and deploying the application on it are tedious and meticulous provider-specific process.

In this paper an integrated set of models, algorithms and tools objecting at reducing developers' tasks to develop, deploy and migrate multiple data stores based applications in cloud environment are discussed.

First, a unifying data model which is used by applications developers to communicate with different data stores is defined. This model deals with queries of heterogeneity within data models and the nonappearance of the schemes in NoSQL data stores. Based on this model, all types of queries using OPEN-PaaS-DataBase API (ODBAPI) may express and executed by developers. This API is a assembled and a unified REST-based API [3] for executing queries over relational and NoSQL data store. The main features of ODBAPI are twofold: (i) seperating cloud applications from data stores to promote the migration process, and (ii) smoothing the developers' task by reducing the burden of managing dissimilar APIs.

Second, virtual data stores (VDS) to calculate and optimize the execution of queries are proposed - especially complex queries over different data stores. In order to define and the execution of queries over heterogeneous data models, the unifying data model has been used to attain with correspondence rules. The solution is based on algebraic trees composed of data sources and algebraic operators and algebraic trees annotation.

Third, a clear approach for discovering proper cloud environments and redistributing applications on them in the time letting developers simply focuses on specifying their storage and computing requirements is presented.

RELATED WORK:

OVERVIEW OF THE SYSTEM

Figure 1 show the main constitutes of the system and how these constituents intervene during the development, discovery and deployment and execution steps. Solution show in particular how below four elements enable overcoming the problems (Pb1 - Pb4) listed in introduction. The solution relies on the following four elements:

A) Unifying data model
B) REST API/services
C) Virtual data stores
D) Dedicated components for discovery and deployment

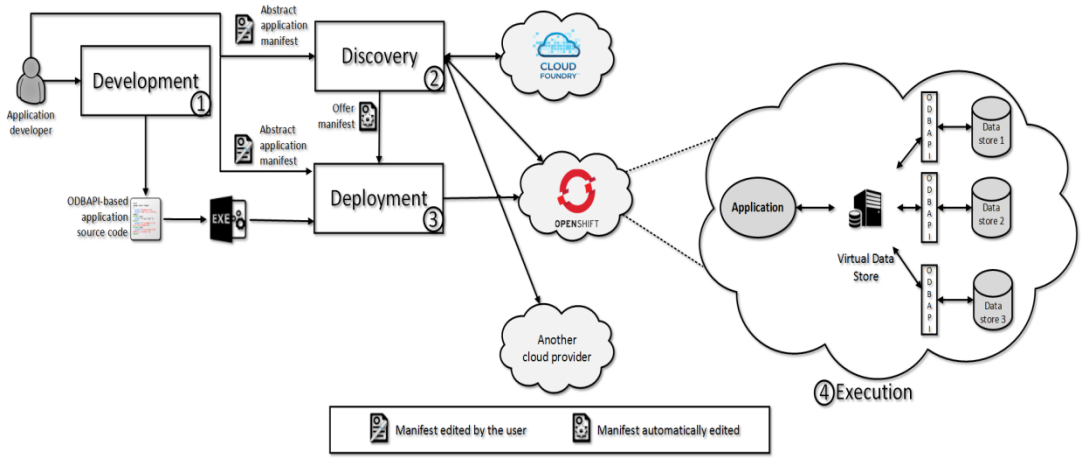

Figure 1: Overview of the solution

A) Unifying data model:

A data model which abstracts from the fundamental (explicit/implicit) integrated data store models, and supplies a unified view so that developers can explain their queries on heterogeneous data stores is defined. The developers' stand of a global data model signified according to unifying model and which accomodate local data store models during the development phase. Unifying data model separates query definitions from the data stores unique languages. (Contributing to resolving Pb1 and Pb2).

For manipulating complex objects, more complex algebra can be used, notably N1NF algebra [4].

B) REST API/services:

Based on unifying data model, a resource model upon which a REST API is developed, called ODBAPI, which enables to communicate with involved data stores in a specific and uniform way. Each data store will be then covered behind a REST service applying ODBAPI. Implemented API separates the intercommunication with data stores through their specific drivers. With the help of unifying data model to define the queries and ODBAPI to communicate with the data stores, developers need not to deal with different languages and APIs and need not to adapt their code when migrating their applications (resolving thereafter Pb1 and Pb3).

C) Virtual data stores

Wrapper REST services allow executing simple queries over the contributed data stores. However, they cannot execute complex queries (like as join, union, etc.). In this approach, virtual data store (VDS) a unique component holding responsibility for executing queries acknowledged by a multiple data store application is considered. A VDS:

(1) Possess the global data model associating the different data stores. which is specified according to unifying data model and a bunch of correspondence rules

(2) Is accessible as a REST service complying to the ODBAPI

(3) Maintains the end-points of the wrapper REST services (in other word the integrated data stores).

VDS executes CRUD and complex queries submitted by a multi data store application by interacting with appropriate data stores via their REST services. Developers are able to express their join queries over multiple data stores in a declarative way and take in charge the burden of their executions through VDS (resolving thereafter Pb2). Query optimization is done using a cost function [5] defined by a linear combination of the response time of the CPU, the time of the input/output, and the time of the communication or the data shipping
(Cost model = $\alpha*t_{CPU}+\beta*t_{I/O}+\gamma*t_{COM}$).

D) Dedicated components for discovery and deployment:

In this approach, the discovery and deployment modules are responsible of finding appropriate cloud environments and deploying multiple data store applications on them respectively. As shown in Fig. 1, developers first express their requirements about the used data stores as well as the computation environment via an abstract application manifest.

(Fig. 2 shows the architecture of abstract application manifest mode). Based on that manifest, the discovery component finds and selects the appropriate cloud environment and produces an offer manifest. This manifest will be in turn used by the deployment component to deploy the application on that selected environment. The discovery and deployment module relieves the application developers from the burden of dealing with different APIs and discovery/deployment procedures (resolving thereafter Pb4).

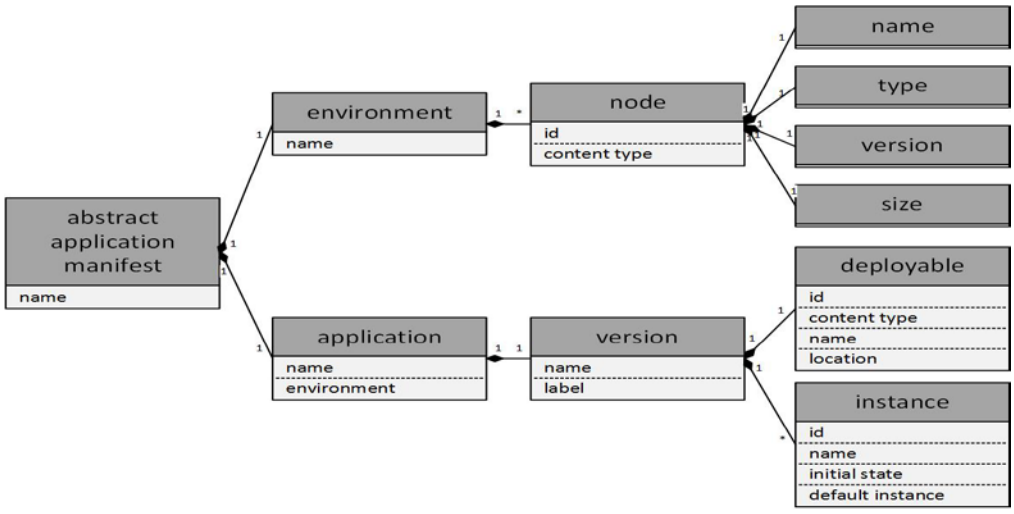

Figure 2: Abstract application manifest model

2. IMPORTANCE

It is easy to handle multi data store application by using this technique. It can handle multiple database types such as relational, NoSQL and key-value by using same APIs, which make easier to code for application developer. There is no need to write different APIs for different data types. It can handle database migration task more easily without affecting the application development code, only we need to make changes in open pass database APIs.

3. CONCLUSION:

In this paper, we have analyzed a generic approach is explained to facilitate the developer task and enable the development of applications using multiple data stores while remaining agnostic to these latter. There is no need to write different APIs to interact with heterogeneous data. Three solutions are explained in this paper: i) Open-PaaS-Database API for CRUD operations. ii) Virtual data stores for complex queries execution. iii) Manifest for data stores discovery and automatic application deployment.

4. REFERENCES:

1. C. Baun, M. Kunze, J. Nimis, and S. Tai, Cloud Computing - Web-Based Dynamic IT Services. Springer, 2011..
2. A. McAfee and E. Brynjolfsson, “Big data: The management revolution. (cover story).” Harvard Business Review, vol. 90, no. 10, pp. 60–68, 2012.
3. R. Sellami, S. Bhiri, and B. Defude, “ODBAPI: a unified REST API for relational and NoSQL data stores,” in The IEEE 3rd International Congress on Big Data (BigData’14), Anchorage, Alaska, USA, June 27 - July 2, 2014, 2014
4. S. Abiteboul and N. Bidoit, “Non first normal form relations: An algebra allowing data restructuring,” J. Comput. Syst. Sci., vol. 33, no. 3, pp. 361–393, 1986.
5. D. Kossmann, “The state of the art in distributed query processing,” ACM Comput. Surv., vol. 32, no. 4, pp. 422–469, Dec. 2000.