INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 5, Issue 4 OCT 2016

Designing a Platform for Multi Data Stores Operations on Cloud Computing Environment

[1]J. KARTHIK, [2]SRIKAR GOUD KONDA, [3]B. DIVYA CHOWDARY

[1]Assistant Professor, Department of CSE, Kommuri Pratap Reddy Institute of Technology, Ghanpur Village, Ghatkesar, Ranga Reddy District, Telangana State, India

[2] M. Tech Student, Department of CSE, Kommuri Pratap Reddy Institute of Technology, Ghanpur Village, Ghatkesar, Ranga Reddy District, Telangana State, India

[3] Assistant Professor, Department of CSE, Kommuri Pratap Reddy Institute of Technology, Ghanpur Village, Ghatkesar, Ranga Reddy District, Telangana State, India

ABSTRACT – One of the most important goals of the PaaS is to support huge data stores by way of ensuring elasticity, scalability and portability. Many of the applications must manage various kinds of data that a single store cannot effectively support. Therefore, clouds need to set up more than one data stores, permitting applications to select the ones corresponding to their data necessities. Many applications and programs need to have interaction with several heterogeneous data stores relying on the form of data they must control: conventional data types, files, graph information from social networks, general key-value information, and many others. Interacting with heterogeneous data models thru distinctive APIs, and multiple data store applications imposes difficult obligations to their developers. Certainly, programmers need to be acquainted with specific APIs. Further, the execution of complicated queries over heterogeneous data models can't, presently, be accomplished in a declarative manner as it's miles was with mono-data store utility, and consequently requires more implementation efforts. Furthermore, developers want to grasp and cope with the complicated procedures of cloud discovery, and application deployment and execution. On this paper we recommend an incorporated set of models, algorithms and equipment aiming at assuaging developers undertaking for developing, deploying and migrating multiple data stores based applications in cloud environments. Our method focuses especially on three points. First, we will provide a unifying data model utilized by applications developers to engage with heterogeneous relational and NoSQL information stores. Second, we recommend virtual data stores, which act as a mediator and have interaction with integrated data stores wrapped by using ODBAPI. This run-time component helps the execution of single and complex queries over heterogeneous information stores. Subsequently, we will present a declarative approach that allows lightening the load of the tedious and non-general responsibilities of (1) discovering applicable cloud environment and (2) deploying applications on them at the same time as letting developers to genuinely focus on specifying their storage and computing necessities.

I. INTRODUCTION

Because of its elasticity property, cloud computing affords interesting execution environments for numerous emerging applications which includes large

data management. In line with the national Institute of standards and Technology1 (NIST), big data is data which exceed the potential or capability of modern or conventional strategies and systems. It is especially based totally on the 3-Vs model in which the three Vs refer to volume, velocity (speed) and variety properties. Volume means the processing of massive quantities of information. Velocity indicates the increasing rate at which data flows. In the end, variety refers to the diversity of information resources. Veracity is extensively proposed and represents the best quality of data. In our work, we primarily focus especially on the variety property and more exactly on multiple information store primarily based applications in the cloud.

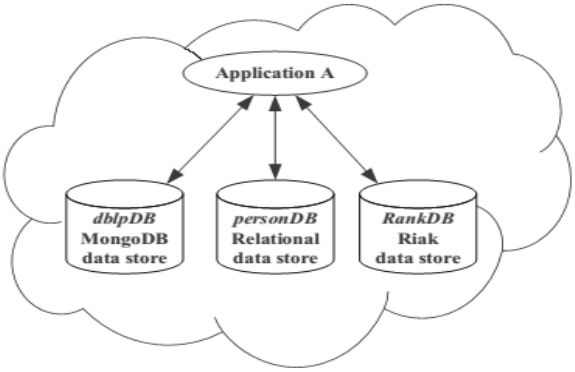


Figure1. Application interacting with three Data Stores in Cloud Computing

In order to satisfy various storage necessities, cloud applications typically ought to access and interact with various relative and NoSQL information stores having heterogeneous Api's. The heterogeneousness of the data stores induces many issues once developing, deploying and migrating multiple data store applications. Below, we have a tendency to list important four issues that we are coping with during this paper.

The Problems while the Multi Data Store based Application is Developed are as follows.

P1: Heavy workload on developer: These days data stores have various and heterogeneous APIs. Developers of multi data store based applications have to be familiar with all of these APIs when coding their programs.

P2: no declarative method for execution of complicated queries: Because of heterogeneity of data models, currently they have any declarative way to define and execute the complex queries over several heterogeneous data stores. This happens mainly due to the absence of a global schema of the heterogeneous data stores.

P3: Code adaptation Problem: When the application is migrating from one cloud environment to another environment, application developers should have to re-adapt the source code to interact with new data stores. Developers also have potentially to learn and master the new APIs.

P4: The tedious and non-standard process discovery and deployment: Once if an application is developed or being migrated, developers should have to deploy it into a cloud environment. Discovering probably the most suitable cloud environment supplying the required data shops and deploying the application on it is tedious and meticulous supplier-detailed approach.

In this paper work, we are proposing an integrated set of data models, algorithms and the tools aiming at reducing developers' tasks for the development, deployment and migrating the multiple data stores based applications in the cloud environment.

First, we outline a unifying data model utilized by applications developers to engage with distinct data stores. Primarily based on this model, developers can also express and execute any form of queries by using OPEN-PaaS-DataBase API (ODBAPI). This API is a streamlined and a unified REST-primarily based API

for executing queries over relational and NoSQL information stores. The highlights of ODBAPI are twofold: (i) decoupling cloud application programs from data stores with a view to facilitate the migration technique, and (ii) easing the developers' undertaking by way of lightening the load of handling distinctive APIs.

Second, we can propose virtual data stores to evaluate and optimize the execution process of queries - especially complex ones- over different types of data stores. Third, we are going to present a declarative approach for discovering of the appropriate cloud environments and deploying the applications on them while permitting developers to simply focus on specifying their storage and the computing requirements.

II. RELATED WORK

In the earlier work, we fascinated about present solutions of the modern-day aiding more than one data outlets headquartered applications in the cloud environment. More exactly, (i) we described different scenarios regarding the best way applications use knowledge retailers, (ii) we defined the data requisites of purposes in cloud environment, and (iii) we analyzed and labelled present works on cloud information administration, specializing in multiple knowledge shops necessities. Thus, we mentioned six necessities of using more than one data stores in a cloud environment. Centred on these requirements, we suggest our finish-to-finish solution to aid a couple of data outlets applications in cloud environments. To the pleasant of our knowledge, there are not any a further international resolution addressing all the problems we focus on. On this part, we talk about the associated works to each and every important contribution in this paper.

We start via imparting works aiming at proposing precise API to enable CRUD queries execution across relational and NoSQL information retailers. There are some proposals just like the spring knowledge Framework , SOS, and ONDM that enable software to query knowledge from exclusive NoSQL knowledge shops. The spring information Framework provides some ordinary abstractions to handle unique forms of NoSQL DBMSs and relational DBMSs. Nonetheless, the addition of a brand new data retailer just isn't so easy and the answer is strongly linked to the Java programming model. SOS presents CRUD operations at the level of person information store. These operations are furnished by way of GET, PUT, and DELETE ways. They argue that SOS may also be accelerated to combine relational knowledge store; meanwhile, there is not any proof of the effectively and the extensibility of their procedure. Whereas, ONDM provides ORM like API established on fashionable Java Persistence API (JPA) to software builders to have interaction with NoSQL data outlets.

Additionally to the CRUD operations execution across relational and NoSQL knowledge outlets, we discover some massive works proposing mediation situated options for join queries execution. In the context of relational integration, Lenzerini formalizes the concepts of relational query rewriting on a couple of sources. In addition, Manolescu proposed an XML situated mediation by using extending these proposals to handle semi-structured data. These works think that a built-in schema (or international schema) exists which isn't possible in our case given that we are dealing with schema-less data shops. For this reason, they may be able to now not help NoSQL data outlets. Our suggestion for expressing and executing becomes a member of queries is closed to these works and we can reuse their optimization methods. Therapy et al represent the global schema by a relational information

mannequin to query NoSQL and relational DBMSs. They outline also a mapping language to map attributes of the information sources to the worldwide schema and a bridge question language to rewrite queries. In the second step, treatment et al lengthens their answer by using an Ontology based data access (OBDA). Additionally, they substitute BQL with SPARQL. Although their notion is promising, there are some missing functionalities.

III. FRAMEWORK

A. System Overview

The approach of a system mainly relies on the four elements. Those are as follows:

i. A Unifying Data Model
ii. REST API/Services
iii. Virtual Data Stores
iv. Dedicated Components for Discovery and Deployment

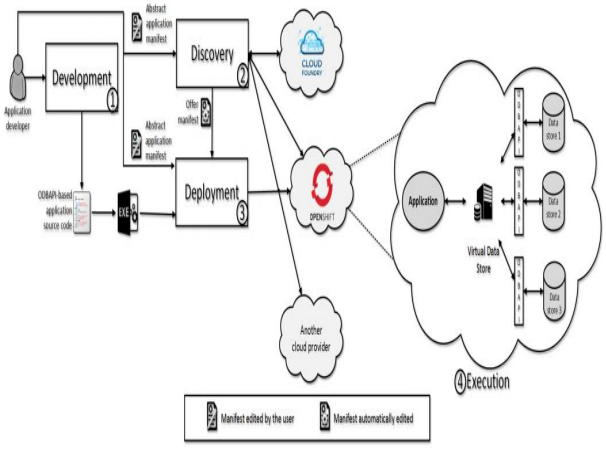


Figure2. Overview of System

Unifying Data Model:

We outline a data model which abstracts from the underlying (specific/implicit) incorporated data store models, and offer a commonplace and unified view in order that developers can define their queries over heterogeneous information stores. In the course of the development step, the developers do away with a worldwide data model expressed in accordance to our unifying model and which integrates nearby data store models.

REST API/Services:

with the help of on our unifying data model, we define a useful resource model upon which we construct a REST API, known as ODBAPI, permitting to interact with involved information stores in a completely unique and uniform manner. Every data store may be then wrapped in the back of a REST service provider enforcing ODBAPI. Our API decouples the interactions with statistics stores from their specific drivers.

Virtual Data Stores:

In our approach, we take into account virtual data store a particular component accountable for executing queries submitted by means of a more than one data store application. A VDS (i) holds the worldwide data model integrating the distinct information stores and that is certain in accordance to our unifying records model and a set of correspondence guidelines, (ii) is obtainable as a rest service complying to the ODBAPI, and (iii) continues the end-points of the wrapper rest services.

Components for Discovery and Deployment of Application:

As shown in Figure 2, developers can express their requirements about used data stores and also computation environment by way of an abstract manifest of an application. With the help of that manifest, a discovery component finds and picks the appropriate cloud environment and then produces an offer manifest. This manifest will be in turn used by deployment component to deploy an application on

particular selected environment. The modules discovery and deployment can relieve the application developers from the heavy burden of dealing with various API's and these procedures.

B. Open PAAS Database API

On this section, we deliver a creation of ODBAPI that is a rest API which permits the execution of CRUD operations on various sorts of data stores helping our unifying model of. This API is designed to provide an abstraction layer and additionally a continuing interaction with information stores which might be deployed in cloud surroundings. Developers can execute their queries in a uniform way irrespective of sort of the records stores (relational or NoSQL). An overview of the API is given in Figure 3.

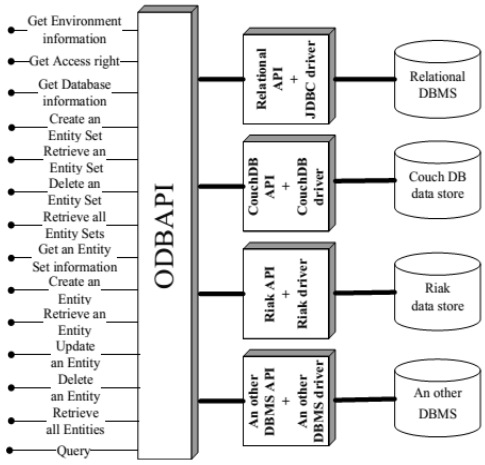


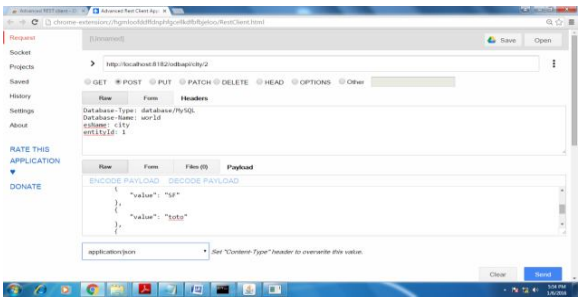
Figure3. Overview of ODBAPI

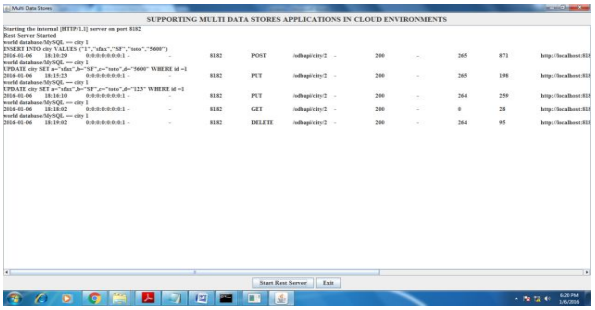
The figure is split in four foremost components that we introduce in a following, begins from right side to the left facet. To start with, we've got few of the deployed data stores (as an instance relational DBMS, couch DB, and so forth.) that a developer can also want to engage with. Second, we can find a proprietary API and the motive force of every data store supported by way of the ODBAPI. As an instance, we make use of API implementation JDBC API and MySQL driving force to engage with the relational DBMS. The third a part of Fig. 3 refers to ODBAPI implementation. However, this component is shared among all integrated records stores. Further, it includes particular implementation of each and every data store. To integrate the new information storage and outline the interactions with it, simply one has to add the precise implementation of specific information store. Finally, Figure. 4 suggests diverse operations that ODBAPI gives.

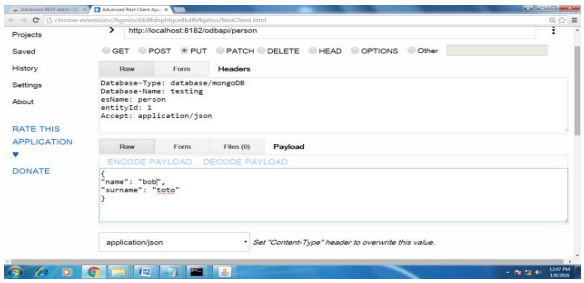
IV. EXPERIMENTAL RESULTS

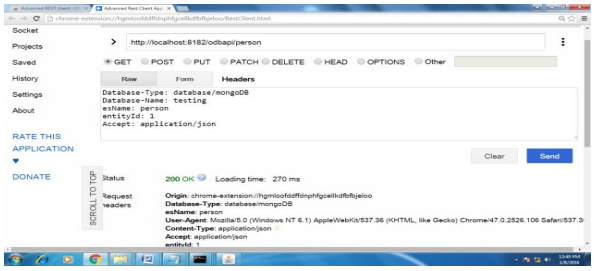
In order to reveal the feasibility and the application utility of our API, we offer a client that we called ODBAPI Client. The latter lets in a developer to use ODBAPI operation thru JAVA techniques. As a result, it is simple for him to program his application. We developed additionally another ODBAPI-primarily based customer meant to address the management of relational and NoSQL data stores in a cloud company. We show a screenshot of the consumer interface of this client. In fact, it gives an outline of two heterogeneous information stores MYSQL and MongoDB.

The experimental results are given below.









V. CONCLUSION

We are using the cloud computing platform to perform many data intensive operations and in cloud environment we are accessing many kinds of services. We are also accessing storage as a service. We store various types of data in various data stores. Each application requires storing and manipulating certain types of data at multiple heterogeneous data stores. But unfortunately there is no such system or API for performing these types of data operations. From this paper we propose an approach to make the undertaking of a developer and to permit application development via using a couple of heterogeneous data stores. We present a unified data model which is incorporated with the REST based ODBAPI that allows interacting with involved data sources in completely unique and uniform manner. Our approach can cope with the virtual data sources to discover and deploy an application in cloud environment with good performance of complex evaluation and execution.

REFERENCES

[1] C. Baun, M. Kunze, J. Nimis, and S. Tai, Cloud Computing - WebBased 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] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, “Consistency rationing in the cloud: Pay only when it matters,” PVLDB, vol. 2, no. 1, pp. 253–264, 2009.

[4] 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.

[5] 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.

[6] D. Kossmann, “The state of the art in distributed query processing,” ACM Comput. Surv., vol. 32, no. 4, pp. 422–469, Dec. 2000.

[7] M. Sellami, S. Yangui, M. Mohamed, and S. Tata, “Paasindependent provisioning and management of applications in the cloud,” in 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013, 2013, pp. 693–700.

[8] R. Sellami and B. Defude, “Using multiple data stores in the cloud: Challenges and solutions,” in Data Management in Cloud, Grid and P2P Systems - 6th International Conference, Globe 2013, Prague, Czech Republic, August 28-29, 2013. Proceedings, 2013, pp. 87–98.

[9] M. Pollack, O. Gierke, T. Risberg, J. Brisbin, and M. Hunger, Eds., Spring Data. O’Reilly Media, October 2012.

[10] P. Atzeni, F. Bugiotti, and L. Rossi, "Uniform access to nonrelational database systems: The sos platform," in Advanced Information Systems Engineering - 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings, 2012, pp. 160– 174.

[11] L. Cabibbo, "Ondm: an object-nosql datastore mapper," Faculty of Engineering, Roma Tre University. Retrieved June 15th, 2013.

[12] R. K. Lomotey and R. Deters, "Rsenter: Tool for topics and terms extraction from unstructured data debris," in IEEE International Congress on Big Data, BigData Congress 2013, June 27 2013-July 2, 2013, 2013, pp. 395–402.

[13] "Data mining from document-append nosql," International Journal of Services Computing (IJSC), vol. 2, no. 2, pp. 17–29, 2014.

[14] M. Lenzerini, "Data integration: A theoretical perspective," in Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ser. PODS '02, 2002, pp. 233–246.

[15] I. Manolescu, D. Florescu, and D. Kossmann, "Answering xml queries on heterogeneous data sources," in Proceedings of the 27th International Conference on Very Large Data Bases, ser. VLDB '01, 2001, pp. 241–250.

[16] O. Cure and et al., "Data integration over NoSQL stores using´ access path based mappings," in Database and Expert Systems Applications - 22nd International Conference, DEXA 2011. Proceedings, Part I, 2011, pp. 481–495.

Author Details

Author 1:

J. Karthik, Assistant Professor, Department of CSE, Kommuri Pratap Reddy Institute of Technology, Affiliated to JNTUH and Approved by AICTE, Ghanpur Village, Ghatkesar, Ranga Reddy Distyrict, Telangana State, India.

Mail id: jilla.karthik@gmail.com

Author 2:

Srikar Goud Konda, M.Tech Student, Department of CSE, Kommuri Pratap Reddy Institute of Technology, Affiliated to JNTUH and Approved by AICTE, Ghanpur Village, Ghatkesar, Ranga Reddy Distyrict, Telangana State, India.

Mail id: kondasrikargoud@gmail.com

Author 3:

B. Divya Chowdary, Assistant Professor, Department of CSE, Kommuri Pratap Reddy Institute of Technology, Affiliated to JNTUH and Approved by AICTE, Ghanpur Village, Ghatkesar, Ranga Reddy Distyrict, Telangana State, India.

Mail id: csehod.kprit@gmail.com