A Survey on Supporting Different Data Stores in Cloud Environment

Priyanka Waghmare[1] Manisha Nirgude[2]
[1,2]Walchand Institute of Technology Solapur

Abstract— Big data is data which increases the capacity or capability of current or conventional methods and systems. It is mainly based on the three Vs refer to volume, velocity and variety properties. Volume means the processing of large amounts of information. Velocity denotes the increasing rate at which data flows. Finally, variety refers to the diversity of data sources. As we are getting these different kinds of data from various sources, one of the challenges is to access such data. In this paper, we discussed some of the solutions to solve the problems of accessing different kinds of the data in cloud environment and the challenges for accessing such a data. We are proposing a unique interface to handle structured as well as unstructured data through restful API.

Key words: Heterogeneous Databases, Restful API, Structured data, Unstructured data

I. INTRODUCTION

Advances in Web technology on the Internet have resulted in immense processing and storage requirements. The data, emerging from different sources like social media, blogs, twitter etc which is in different format. Generally such kind of data may be in structured as well as unstructured format. To analyze such data, big data applications usually need to interact with different structured and unstructured data store. Each database has their specific APIs to access it. The data heterogeneity creates several problems while analyzing multiple data store applications. Specifically the application developer have many problems like heavy workload, no declarative way for complex query execution, code adaption etc.

In order to satisfy different requirements of user, cloud applications usually need to interact with different structured and unstructured data stores having heterogeneous APIs. This APIs heterogeneity produces two main problems. First it will connect cloud applications to specific data stores interfering therefore their migration. Second, it requires developers to be known with different APIs. In this work, the main approach is to use streamlined and a unified REST API enabling to execute different unary and binary operations on different structured and unstructured databases. It will be useful for the application developer who wants to access different heterogeneous databases for analysis purpose and also it will be helpful to reduce the overload of the developer

This API is used to provide the seamless interaction with data stores located in a cloud environment. Developers can execute different unary and binary operations.

A. *Restful API*

REST stands for Representational State Transfer. It is based on a stateless, client-server, cacheable communications protocol and in virtually all cases, the HTTP protocol is used. The main purpose of ODBAPI are as one for decoupling cloud applications from data stores in order to the shifting process , and second for easing the application developers task by reducing the burden of managing different APIs. ODBAPI separates cloud applications from data stores showing therefore their migration. Moreover it reduces developer's task by removing the burden of managing different APIs.

B. *NoSQL*

NoSQL is a technology for storing data which does not have specific structure that it is useful for storing unstructured data. They don't require SQL statements for accessing data stores. The NoSQL databases can usually scale across different servers. With the help of this we access different heterogeneous unstructured data stores.

II. LITERATURE REVIEW

In this section, we discussed different techniques for accessing structured as well as unstructured data. Some problems and their solutions on cloud environment.

H. L. Truong, M. Comerio, F. D. Paoli, G. R. Gangadharan, and S. Dustdar studied different problems of cloud computing. Some of them are discussed as follows.[1] This provides dynamically scalable and frequently virtualized resources which are offered as services. In this case, there is a services pool that supports platforms and mechanisms in order to surety the applications development and execution. Such services is called platform as a service (PaaS). One of the main goals of the PaaS is to handle large data stores by assuring elasticity, scalability and portability. Many applications have to manage various types of data that a single store can't efficiently support. Adjunct , clouds need to exert multiple data stores, allowing applications to select those consonant to their data requirements. They discuss the necessity of such environments and analyze current state of the art. They defined and classified the applications requirements in term of multiple-data stores in the PaaS through three possible cases. They have analyzed the literature of these requirements and presented a synthesis of this analysis. This combination shows that many works are still needed to support all desired requirements. We are currently working to address part of these requirements, focusing on the design and implementation of a stretch based approach, allowing applications to negotiate the best data store corresponding to their data requirements.

P. Atzeni, F. Bugiotti, and L. Rossi addresses needs that are very important in large-scale applications, commenting because of the contrasting with well-known relational achievements.[2] One of the main problems often mentioned is the heterogeneity of the languages and the interfaces they tender to developers and users. Various platforms and languages have been proposed, and applications developed for one system require main effort to be migrated to another one. Here they propose a unique programming interface to NoSQL systems (and also to relational ones) called SOS (Save Our Systems). Its goal is to support application development by abstracting the specific details of the various systems. It is based on a meta modeling approach, in the sense that the fixed interfaces of the individual systems are mapped to a common one. The

tool gives interoperability as well, since a single application can interact with several systems at the same time.

They introduced a programming model that enables homogeneous treatment of non-relational schemas. They provided a meta-layer that allows the creation and querying of NoSQL databases defined in MongoDB, HBase and Redis using a common set of simple atomic operation. They also described an example where the interface we provide enables the simultaneous use of several NoSQL databases in a way that it is available for the application and for the programmers. It is viewed that such elementary operations might reduce the power of the underlying databases. Actually, in this paper they do not deal with a formal analysis of information capacity of the involved models. However, it is explicit that when lower level primitives are involved, expressive power is not limited with reference to the whole language, but only to the single statement. This means that a query that can be showed as one statement in HBase, for example, will require two or more statements in the common query language.

O. Cur´e and et al studied techniques for the large amount of data generated by user interactions on the Web,[3] some companies are currently modifying in the domain of data management by designing their own systems. Many of researcher are referred to as NoSQL databases, standing for 'Not only SQL'. With their vast adoption will emerge new needs and data integration will surely be one of them. In this paper, they did adapt a framework meet for the integration of relational data to a large context where both NoSQL and relational databases can be integrated. One important circulation consists in the efficient answering of queries expressed over these data sources. The highly denormalized mode of NoSQL databases results in varying performance costs for several possible query translations. Thus a data integration aiming NoSQL databases needs to generate an optimized translation for a given query. Their contributions are to propose (i) an access path based mapping solution that takes benefit of the design choices of each data source, (ii) integrate preferences to handle conflicts between sources and (iii) a query language that bridges the gap between the SQL query expressed by the user and the query language of the data sources. We also present a prototype implementation, where the target schema is represented as a set of relations and which enables the integration of two of the most popular NoSQL database models, namely document and column family stores.

This paper is a first approach to integrate data coming from NoSQL stores and relational databases into a single virtualized database. Due to the increasing popularity of this novel trend of databases, we consider that such data integration systems will be quite useful in the near future. Our system adopts a relational approach for the target schema which enables end-users to express queries in the declarative SQL language. Several modification steps are then required to obtain results from the data stored at each of the sources. Hence a bridge query language has been presented as the cornerstone of these transformations. Another important part of their system is the mapping language which (i) handles uncertainty and contradicting information at the sources by defining preferences over mapping assertions and (ii) supports the setting of access path information in order to create an efficiently process able query plan. On basic results, the overhead of these transformation steps does not impact the performance of query answering.

Some researcher proposed different Big Data scenarios often involve vast collections of nested data objects, typically referred to as "documents." [4][7][8] The researchers analyzed some problem of document management at web scale level have inspiring a latest view towards the development of document-centric "NoSQL" data stores. Many query tasks naturally contain reasoning over data residing across NoSQL and relational "SQL" databases. Having data divided over different stores currently implies labor-intensive manual work for data consumers. In this paper, they propose a general framework to coupling bridge the gap between SQL and NoSQL. In their framework, documents are logically incorporated in the relational store, and querying is performed via a novel NoSQL query pattern extension to the SQL language. Such behavior allow the user to gives conditions on the document-centric data, where the rest of the SQL query refers to the corresponding NoSQL data via variable bindings. One of the solution for translating the user query to an equivalent SQL query, and present optimization strategies for query processing. They implemented a prototype of our framework using PostgreSQL and MongoDB and have performed an extensive theoretical analysis. Their study shows the practical probability of their framework, proving the possibility of seamless coordinated query processing over relational and document-centric data stores.

In this paper they presented the first generic extensible framework for coordinated querying across SQL and NoSQL stores which removes the need for ad-hoc manual intervention of the user in query (re)formulation and optimization. An extensive theoretical study manifested practical feasibility of the framework and the proposed implementation strategies. The groundwork laid here opens many interesting avenues for further research. They close by listing a few undertaking directions. They have just scratched the surface of implementation and optimization strategies. They give two opinions for future work here. They will try to study adaptations and extensions of indexing and caching mechanisms developed for RDF, a triple-based data model, and XML to more scalable implementations of F. Individual queries are often part of a longer-running collection. It would be definitely worthwhile to enquire strategies for multi-query optimization with respect to a dynamic query workload. There is recent work in the community towards standardization of document query languages and their semantics. An important interesting topic for further investigation is to support their results with these emerging efforts.

Some Researcher studied different data types that have been increasingly provided using the data-as-a-service (DaaS) model, [5][6] a form of cloud computing services and the core element of data marketplaces. This provides the on-the-fly data composition and usage for several data intensive applications in e-science and business domains. However, data offered by DaaS are constrained by several data concerns that, if not automatically being reasoned properly, will lead to a wrong way of using them. In this

paper, they support the view that data concerns should be explicitly modeled and specified in data contracts to support concern-aware data selection and utilization. They perform a detailed analysis of current techniques for data contracts in the cloud. Instead of relying on a specific representation of data contracts, we introduce an abstract model for data contracts that can be used to build different types of data contracts for specific types of data. Based on the abstract model, we propose different techniques for evaluating data contracts that can be integrated into data service selection and composition frameworks. They also illustrate their approach with some real-world scenarios and show how data contracts can be integrated into data agreement exchange services in the cloud.

Although various data marketplaces and DaaS emerge and provide multitude sets of data, data contracts associated with these data so far are mainly written in textual form for human beings. Furthermore, what constitutes data contracts has not been deeply investigated. In this paper, we analyze data contracts in DaaS and data marketplaces in detail. They have developed an initial abstract data contract model that can be used by different communities to specify conditions applied to data provided via DaaS. Their approach for supporting the definition of data contracts that takes into account diverse types of data terms is based on the community model. Based on our data contract model, we have presented some possible methods and defined guidelines to develop an application for data contract compatibility evaluation for data composition.

III. Conclusion

In this survey paper, we have discussed different problems and challenges to access different heterogeneous data stores on cloud environment and how to access different data stores on cloud environment. Also we have seen how to handle with structured as well as unstructured data. We are proposing A Common Interface to reduce the workload of an application developer to access different kinds of data which is stored on heterogeneous data model. It will be useful for optimizing different query execution on cloud Environment.

Acknowledgments

I would like to thank Department of Computer Science and Engineering, Walchand Institute of Technology Solapur.

References

[1] H. L. Truong, M. Comerio, F. D. Paoli, G. R. Gangadharan, and S. Dustdar, "Data contracts for cloud- based data marketplaces," IJCSE, vol. 7, no. 4, pp. 280–295, 2012.
[2] 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.
[3] O. Cur´e 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.
[4] J. Roijackers and G. H. L. Fletcher, "On bridging relational and document-centric data stores," in Big Data - 29th British National Conference on Databases, BNCOD'13, 2013, pp. 135–148.
[5] C. Baun, M. Kunze, J. Nimis, and S. Tai, Cloud Computing - WebBased Dynamic IT Services. Springer, 2011.
[6] 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.
[7] D. Kossmann, "The state of the art in distributed query processing," ACM Comput. Surv., vol. 2, no. 4, pp. 422–469, Dec. 2000.
[8] M. Sellami, S. Yangui, M. Mohamed, and S. Tata, "Paas independent 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.
[9] C. Baun, M. Kunze, J. Nimis, and S. Tai, Cloud Computing - WebBased Dynamic IT Services. Springer, 2011.

