Computer Standards & Interfaces 47 (2016) 42–51

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Conversion cost and specification on interfaces of key-value stores

Jie Song [a,*], Kun Guo [a], Jieping Wang [b], Haibo Li [b], Yubin Bao [c], Ge Yu [c]

[a] Software College, Northeastern University, Shenyang 110169, China
[b] China Electronics Standardization Institute, Beijing 100007, China
[c] School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China

A R T I C L E I N F O

Article history:
Received 20 August 2015
Received in revised form 14 February 2016
Accepted 14 February 2016
Available online 2 March 2016

Keywords:
Key-value stores
Interoperability
Conversion costs
Interfaces specification

A B S T R A C T

Due to the quick growth of data created and analyzed by industry and business requirements become more complex, many companies come to employ more than one key-value store together to serve different tasks. Considering key-value stores currently define their own interfaces which have different attributes and semantics, interoperability among these key-value stores is weak. To get the best interoperability, we may choose the store whose interfaces are similar to the others, or we may define an interface specification such as SQL specification in relational databases. We propose an interface description model (IDM for short) to abstract interfaces of different key-value stores, and an algorithm to quantify their differences, named as conversion cost. With the help of these algorithms, we can measure and compare the interoperability of given two stores. After studying the interoperability of many stores, we propose an interface prototype, which has the minimum conversion cost to the interfaces of other stores, as a reference to the interface specification of key-value store. Experiments show the features of interfaces, and prove that the proposed prototype has the best interoperability to other typical stores.

1. Introduction

With the rapid development of information technologies, the scale of data created and analyzed by industry grows larger and larger. Traditional relational databases, which organize data into relational models, meet a performance bottleneck when querying and managing big data in such situations [1]. To break the limitation of relational databases, NoSQL database such as key-value store emerged in recent years. Data relationship in key-value store is simplified and transactional property for ACID is generally given up [2]. Varieties of key-value stores are designed in different architectures to meet different requirements, but all of them follow key-value data model which will be explained in Section 3. Due to their great performance and flexibility, key-value stores are widely used in industry in support of the big data management [3].

Different databases, especially databases in different categories, are in different architectures. If multiple databases are employed within a system, each database has its own advantages when handling some specified types of data, and is efficient in specified scenarios. Using and maintaining more than one database within a system help to reduce the heavy burdens caused by various use cases. Each use case corresponds to the most appropriate database, therefore the efficiency and scalability of system are guaranteed. And for techniques such as data warehouse and data integration, due to the variety of big data, heterogeneous databases are often employed for storing data. Besides, the system, which can employ multiple databases, has better portability. It avoids vendor lock-in by changing existing database easily, compared with the system that only locked on one database. The last but not least, applications should access multiple databases via a uniform interface. Therefore, the diversity and complexity of the underlying databases are transparent to the developers.

Because of the previous reasons, the usage of multiple key-value stores becomes more common and used frequently in industry. For an operational system, performance is the key issue, and diverse data are required to meet various requirements. Multiple databases support the diverse data well with high performance. For analytical systems, it is frequent to perform data analysis on the massive and various data. We tend to store data in heterogeneous databases because of the different equipment or different approaches it takes during data collection. The middleware of query driven data integration, which access multiple databases with a uniform interface, is frequently adopted in data analysis system.

Actually, most Internet companies like Facebook have already deployed multiple databases into their development environments. Facebook adopts several key-value stores to meet its various business requirements: HBase for messages service and monitoring, Haystack for photo store and Memcached for in-memory data store. Twitter employs Cassandra for atomic counting and HBase to power its search engine. E-business sites like Amazon adopts key-value store such as MongoDB and Riak to record users' click stream, and Redis to achieve effective static pages serving or caching of product related data. The Chinese online shopping platform, named TaoBao, also adopts the in-

* Corresponding author
E-mail address: songjie@mail.neu.edu.cn (J. Song).

http://dx.doi.org/10.1016/j.csi.2016.02.007
0920-5489/© 2016 Elsevier B.V. All rights reserved.

memory database, transaction supported key-value database, massive contents oriented key-value store and achieved oriented storage to support its business. Interoperability, representing the ability to share data and work together among key-value stores, is necessary [4]. From another perspective, considering the compatibility, portability, extensibility and reusability of system, it is underlying databases that dominate these properties. It is beneficial if a system could change their underlying databases easily, or support heterogeneous databases. For instance, the uniform interface of data access layer and the lower cost of mapping database interface which will contribute to the portability.

Existing key-value stores, such as HBase, SimpleDB, CouchDB, Riak and MongoDB, have similar data model but distinctive architectures. They provide the similar interfaces, but the same concept may correspond to different terms and structures. Most key-value stores now provide interfaces to access data directly through HTTP protocol that is RESTful API (Application Program Interface) [5]. However, when invoking interfaces through HTTP protocol, users must create a HTTP request and then parse the data appended in the HTTP response, and the corresponding key-value store defines the particular data format of HTTP messages. Due to the difference of interfaces in format, it is inevitable to develop specialized programs, which are costly and less of scalability, for data exchange and data integration needs among key-value stores. Interoperations among database based applications and programming on data access frameworks are hard to achieve as well.

Considering interfaces of key-value stores are more or less distinctive, users have to understand many interface formats. Data access turns to be inefficient and tedious. If there is a specification on interfaces of key-value store, as same as SQL of relational databases, the problem of interoperability and vendor lock-in could be solved. When replacing existing key-value stores, such kind of specification helps to choose the appropriate one among alternative stores, which has less conversion cost to achieve interoperations with key-value stores remained in the system. And it can also provide guidance for the design of related database interfaces and standards through finding relatively less costly interface format. Unfortunately, there is no such specification so far and few researches study the conversion costs of interfaces of key-value stores. In this paper, based on the “Information technology, cloud data storage and management, part 5, specification on interfaces of Key-value store” [6], which is the project issued by “Information Technology Standardization Administration of China”, we define and evaluate the conversion cost on interfaces of key-value stores, and provide a reference to the specification on them.

In this paper, database refers to key-value store since which is not a proper database; interface refers to an operational one in database, such as insert interface and delete interface; (interface) distance is a cost of adapting one interface into another. Database interfaces refer to all operational interfaces of database; database distance is an aggregated cost of adapting interfaces of one database into those of another database correspondingly. Based on the previous description, there are several pending questions which may help to the interface specification of key-value stores. (1) Given interfaces of two databases, what is the conversion cost between them, how to quantify the cost. (2) Whether there is an existing key-value store which has the minimum conversion cost to other databases; (3) Theoretically, whether an interface, whose conversion cost to the other interfaces is minimum, could be defined. (4) How to design and estimate the interface specification. As much as our knowledge, several research works, which are explained in Section 2, are focus on the similar topics.

In this paper we present an interface description model to describe interfaces of key-value stores, and the model highlights the crucial structure that makes the interface distinctive among others. Each interface is abstracted into a tree structure through this model and we propose a quantification algorithm to evaluate the difference, named as “distance”, between interfaces. Essentially, given two interfaces of the same operation in two key-value stores, the distance is the cost it takes when adapting one interface into another. Referring the edit graph algorithm, the distance can be calculated, and then, the interfaces which have the minimum distances to the others indicate a theoretic specification. Experimental results prove that the proposed specification has less distance than other interfaces from existing key-value stores. Our contributions in this paper are listed as follows. (1) Propose an interface description model to abstract interface characteristics. (2) Propose a general way to quantify and evaluate the difference between two RESTful interfaces defined by key-value stores and represent the difference as distance. (3) Based on the conversion costs of interfaces of existing key-value stores, we have discussed the use of the proposed approach and propose an interface specification (a referenced one).

The rest of this paper is organized as follows. Following the introduction, Section 2 introduces the related work. Section 3 introduces the definitions of interface description model and Section 4 explains the algorithms of distance quantification. Section 5 discusses several potential scenarios of the conversion cost and Section 6 introduces several proposed database interfaces as a prototype of national specification on interfaces of key-value store. And in Section 7, we evaluate the distance of HBase, SimpleDB, CouchDB, Riak, MongoDB and our Prototype. The experiment result shows that Prototype has these minimum conversation costs than others comprehensively. Finally, conclusions and future works are summarized in Section 8.

2. Related work

To guarantee the interoperability of traditional relational databases, some researchers suggest integrating multiple autonomous database systems as a federated database [7,8,9]. The federation database mainly maintains mappings to data structures or schemas of any two different databases. Although the solution has solved data integration problem among different databases, however, it tends to build a management system over existing databases. It is hard to maintain the system and fails to fix the problem from the perspective of high scalability and interoperability. In fact, interoperability could be simply improved if all databases follow an interface specification, such as SQL language in relational databases. SQL-like query is also an effective implementation to access data stored in different key-value stores. In the data integration, SQL queries express data conditions to filter data in various key-value stores and relational databases. Those conditions also help to organize data into a predefined schema that is easier to access and analyze [10,11,12,13].

Atzeni et al. proposed a common programming interface [14], which plays a meta-layer role when users access to key-value stores, to reduce the heterogeneity of interfaces during the development. This common interface is designed in an object-oriented manner and provides three simplified operations, *put*, *get*, and *delete*, for users to interact with key-value stores. By calling corresponding methods and passing specified parameters, the details and constraints of underlying key-value stores are transparent to users and developers.

Marinos et al. presented a series of patterns that make relational databases accessible through Web services possible [15]. Haselmann et al. proposed a group of universal APIs to support unified APIs used for diversity of Data-as-a-Service systems [16]. The paper introduced a feasible way to design universal APIs and discussed the features of universal APIs'. These APIs, such as querying data and defining schema, are RESTful, resource-oriented, and intuitional URIs. However, they lack the comprehensive definitions of HTTP request and response, so that these universal APIs are not completed and impractical. These APIs inspire our work because they are not only compatible with traditional relational databases and XML databases, but also NoSQL databases. However, the weakness is that the generalization makes these APIs redundant and hard to describe even represented in REST style. No evaluation standard or experiment is proposed in the paper, it is hard to estimate the effectiveness and efficiency.

The similar interface definitions are WADL [17]. It formalizes the description of HTTP-based applications, which provide programmatic