A Key-Value based Application Platform for Enterprise Big Data

Bo Hu\(^1,2\), Yutao Ma\(^3,4,*,\), Liang-Jie Zhang\(^1,2\), Jiake Shi\(^1\), Jiayan Zhong\(^1\)
1. NERC for Supporting Software of Enterprise Internet Service, Kingdee, Shenzhen 518057, China
2. Kingdee Research, Kingdee International Software Group Co. Ltd., Shenzhen 518057, China
3. State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
4. WISET Automation Co., Ltd., Wuhan Iron and Steel Group Corporation, Wuhan 430080, China
*E-mail: ytma@whu.edu.cn

Abstract—Big data poses big challenges to modern enterprises. Traditional data-intensive business applications begin to fall behind the times, because of insufficient capabilities to process large data volumes, ever-changing streaming data and unstructured information. Furthermore, large, complex and all-in-one enterprise applications are no longer popular, whereas lightweight and fragmentary applications become welcome for the sake of the bursts of cloud computing and mobile internet. Below this kind of situation, this paper proposed a big data application platform for enterprises to simply develop and operate personalized data retrieval, data analysis, business intelligence and other data-intensive services. Unlike current related products, it is distinguished for the design of key-value based hybrid data storage and a service-oriented outsourcing architecture. The former is used to resolve the frequently-encountered issue of diversified and massive data storage, and the latter enables an open environment for third-party vendors, which promotes a self-increasing services ecosystem for big data application development. In order to validate the feasibility of our platform, the paper also developed a sentiment analysis application for Kingdee popular products based on the services provided by the platform. The result of our experiment proves that the platform shows enough potential to effectively facilitate data-intensive application development.

Keywords—key-value store; application platform; big data; business intelligence

I. INTRODUCTION

In 2011, 79% of 2,100 companies surveyed by Harvard Business Review reported that they used or planned to use social media. The increasing usage of Facebook, Twitter and other applications of Web 2.0 for business witnessed an Internet-driven transformation of enterprises. Besides being used in marketing goods and services, these applications are also being used in collecting and analyzing market feedbacks, and in connecting employees, coordinating suppliers and streamlining nearly every aspect of modern enterprises. According to the forecast of McKinsey & Co., in the next 5 years, marketers anticipate spending 19.5% of their budgets on social media, nearly three times the current level.

However, a large number of enterprises are faced with the dilemma of being unable to make use of those social applications to their best advantage, because few of them have the capacities to store, process, and reuse such huge amount of data. As the age for Big Data is coming, all data in modern enterprises have skyrocketed in the last several years, most of which are unstructured, diverse and complex. Unfortunately, as we know, big data processing is beyond the abilities of current data-intensive software techniques such as Business Intelligence (BI) and On-Line Analytical Processing (OLAP). On the other hand, large, complex and all-in-one desktop applications are no longer popular, while lightweight and fragmentary applications for mobile electronic devices become welcome for the sake of the bursts of cloud computing and mobile internet. Below this kind of situation, novel methods and relevant XaaS (Anything as a Service) products should be developed to meet the enterprises’ requirements in the era of Big Data.

Therefore, this paper presents the ideas and design about a big data application platform for modern enterprises to support easy-to-use personalized data retrieval, data analysis, business intelligence and other data-intensive services. The platform was designed based on Data Pool, a hybrid storage that can implement data persistence of massive data with both structured and unstructured features. Then, the Data Pool was encapsulated as a storage service, incorporated with other components such as Data Adapters, Algorithm Library and App Keeper, so as to provide data access, data storage, data processing and management functions for upper-level applications based on the Service Oriented Architecture (SOA). Compared with traditional data-intensive systems, the platform we proposed is big data-oriented, business independent, and pluggable and extensible.

The rest of the paper is structured as follows. Section II introduces the motivation to develop such a platform and the requirements of modern enterprises in the era of Big Data. Section III presents the details about Data Pool, which is a hybrid storage unit with unified interfaces for big data access. Section IV depicts the design of service oriented outsourcing architecture for the platform. In Section V, we present the result of our experiments. Section VI introduces the related
work, and we draw the conclusion and point out the future work in the Section VII.

II. MOTIVATIONS

A. Background

In the past decades, Data Warehouse has been widely used in enterprise data analytics. Business Intelligence (BI) provides a set of methodologies, processes, architectures, and technologies, which transform raw data into valuable information to enable more effective strategic, tactical, and operational insights and decision-making [1]. As we know, BI applications use data gathered from a data warehouse or a data mart. However, the situation begins to change when meeting the growing trend of big data.

In recent years, with the widespread use of wiki and the popularity of micro blog and other Web 2.0 applications for business, there has been an explosive increase in the amount of data in different types of enterprises, which even exceeds the rate of Moore’s Law. For example, Wal-Mart, one of the famous worldwide supermarkets, collects more than 2.5 petabytes (PB) of data every hour from its customer transactions, and it also has related 40 billion photos held by Facebook alone in order to facilitate the marketing. According to the International Data Corporation’s (IDC) estimate, modern enterprises will need to manage, on average, 50 times more information by the year 2020 [2], which is actually a great challenge for medium-sized and small enterprises. Thus, it can be seen that big data does move into enterprises.

Undoubtedly, big data brings big opportunities. The integration and analysis of big data can help enterprises glean deeper insights into the internal and external forces that affect their performance, anticipate development trends, and respond more quickly to changes. Until now, more and more companies have recognized that there is a lot of treasure contained in these huge datasets, indicating that big data adoption goes main stream in enterprises. According to IDC’s latest survey in January, 2014, there are 70% of enterprises have either deployed or are planning to deploy big data related projects and programs, and the expense of every enterprise is expected to reach, on average, 8 million US dollars in the coming year.

A coin has two sides. We have to admit that big data also poses big challenges. In general, big data owns the “4-V” characteristics of increasing volume (amount of data), velocity (speed of data in and out), variety (range of data types and sources) and veracity (uncertain or imprecise data) [3], which introduces unique computational and statistical challenges, including scalability and storage bottleneck, noise accumulation, spurious correlation, incidental endogeneity and measurement errors. These challenges and opportunities associated with big data necessitate rethinking many aspects of data management software. Moreover, with the popularity of cloud computing and mobile internet, current applications are required to be able to address both agile development requirements, and to handle massive structured and unstructured batch-oriented use cases.

B. Requirements

Considering the 4-V characteristics of big data [3] and the requirements for enterprises’ application development, in order to overcome the challenges big data caused, and to help enterprises seize the opportunities big data brings, these products are desired to provide the following functions:

- **Scalable Enough.** Data storage should always be considerably designed in data-intensive application development. Traditionally, we always preferably choose a relational database management system (RDBMS) in that several decades of successful application has proved that the system is reliable, effective and robust. Although it could be scaled up or scaled out to some extent, there are also many bottlenecks on its scale and speed when processing trillions of data [4, 5].

- **Multi-typed data supported.** Besides existing relational databases, documents, e-mails and attachment, photos and videos, and Internet search indexes, log files and social media become the primary data sources that need to be stored, processed and utilized by enterprises. It is difficult for those rigidly-defined and schema-based existing approaches used by relational databases to quickly incorporate these unstructured and semi-structured data [6], which requires the products designed for unstructured data as well as structured data.

- **Business driven and agile development.** In order to adapt to rapidly changing business requirements, or to develop applications for mobile electronic devices, data-intensive applications are no longer complex and all-in-one, and this requires the products should be designed to be more simpler and lighter than ever before, and could be adjusted rapidly to meet the needs of business development.

Therefore, we would like to design a platform which could pull together the data both within and outside an enterprise, provide easy-to-use facilities to store and process them, and offer services to make it faster and simpler to develop big data-oriented applications. Moreover, the platform could be deployed as a service in private clouds, and provides unified data access to unstructured data as well as structured data.

C. Merits and Demerits of Key-Value Store

In recent years, NoSQL (Not only SQL) becomes a rising star in the realm of database. As we know, a relational database is predefined by the layout of the tables and the fixed names and types of columns [5], whereas NoSQL offers a simpler data model for data storage and plentiful functionalities for query. Furthermore, NoSQL provides a significantly higher data throughput, and supports horizontal scalability and running on commodity hardware [7]. Therefore, it has been gaining great attention from multiple famous Internet service providers, who have to deal with vast amounts of unstructured data, like Google, Facebook and Twitter.

For different purposes of usage, there are several types of NoSQL databases, and the typical databases are Key-value
store, Document Database, Graph Database and Column-oriented Database. As the basic type of NoSQL database, Key-value store is famous for its simplicity. It employs a key-value pair model, and most of the data in it are stored as key value pairs, where a value is a data object itself and each object is identified and indexed by a unique key. Based on Key-Value store, Column-oriented Database store and data processing by columns instead of in rows can achieve high performance. Notable products in this field are Google’s BigTable [8] and HBase [9].

Despite the above-mentioned merits, it also has some shortcomings, and one of the most significant one is a lack of data model. The data stored in a key-value store is not application-independent due to such a shortcoming. These data should be modeled and maintained by specific applications, implying that other applications are unable to use the same data without learning the underlying data model. This hinders data accessing across different applications and increases the difficulty of programming for new data-intensive application development.

III. DATA POOL: A HYBRID DATA STORAGE

A. Unified Architecture of Data Storage

Data Pool is a storage facility of the application platform, which provides services for diversified and massive needs of data storage. Focusing on the situation and issues mentioned before, we designed a hybrid one to serve both structured and unstructured data with a unified access interface, since both current relational and NoSQL databases can’t meet the requirements.

![Figure 1. The architecture of Data Pool](image)

Unified Storage Interface

Key-Value Store Unit

Relational DB

KDFS (Kingdee Distributed File Service)

Figure 1. The architecture of Data Pool

Figure 1 depicts the hierarchical architecture of Data Pool which consists of four layers, namely, KDFS Layer, Relational Database Layer, Key-Value Store Layer and Unified Storage Interface Layer.

KDFS Layer is the infrastructure layer of Data Pool, which provides distributed file service for upper-level layers. Here we selected KDFS (Kingdee Distributed File Service) as the base to build the service, and KDFS is developed based on an open-source distributed file system, which has been optimized for the purpose of small-sized file storage and sharing.

Relational Database Layer uses a RDBMS to manage the models or metadata of applications. When initializing the storage of data, each application is required to submit its data models or metadata to the storage. These models and metadata are all stored in the RDBMS. The advantage of storing data models or metadata is to facilitate data access across different applications. Furthermore, this also facilitates data indexing because these data are stored as values in different partitions that follow the keys generated by the imported models and metadata.

Key-Value Store Layer includes a cluster of key-value data store servers running in several partitions. A partition acts as a basic computing and storage unit for data processing, and it is a sub-cluster of the storage that could be scaled up automatically in case of lower capacity. Following the popular Master/Slaves style, there exist at least two servers in a partition, where one server is selected as the master while the other server acts as a slave. One or more slaves should be configured to accomplish load sharing, provided that the data to be processed overloads.

Data Access Layer is supplied as a web service and provides a standard interface for unified storage. Applications of the platform that obtain authorization for their operational use, as will be explained in the coming section, can invoke the interface to store their models, metadata and data in the storage. Generally speaking, applications should initialize the storage through the interface when they first use the storage, so as to notify the storage how much of the data it would like to store.

B. Hybrid Approach of Unified Storage

Inspired by the Data Point Model (DPM) [10], an arising multidimensional data point structure standard drafted by European Central Bank and other associated financial institutions for financial data statistics and analysis, we proposed a key-value based data modeling approach. In this sub-section, we will introduce the modeling approach analogous to the DPM, so as to specify how to organize data models or metadata in the Data Pool. Figure 2 depicts an ER diagram that describes the meta-model of the data modeling approach. The meta-model is very simple, and includes only three core entities and several properties within them.

![Figure 2. Meta-model of the relational database in this paper](image)
The entity *App* is used to specify all applications developed on the platform. Each application is identified by an app id, and it is described by app name and other properties. Security key and access token are important properties of an application, and they are the necessary permits once an application wants to access its related or authorized data, or to invoke services on the platform. Although the details of these data objects are stored in the key-value store, their models or metadata should be persisted individually so that these data objects can be indexed and searched by other applications.

The entity *Basic Item* is used to describe the persistence of data objects in an application. Each data object is also identified by a key, and it is described by object name. An object name is only an abstract concept that can roughly describe a data object. For example, we use a concept *cat* to describe an animal that likes to catch mice, but if we want to know which type of such a cat is, more information is needed. Therefore, *Dimension* is such an entity to describe the detailed information about *Basic Item*. Generally speaking, more dimensions a data object has, the more specific the data object is. In this paper, dimensions are used to specify basic items, while types and other properties are utilized to specify dimensions. Here types could be *object*, *integer*, *varchar*, *text* and other common data types of rational databases.

Take for example the storage of forum posts. Let us suppose that a forum application developed based on our platform wants to store all posts in the Data Pool. For the sake of simplicity, the model of such posts has been simplified into three dimensions, including author, title and tag. Tag is the type of *text*, while both author and title are the type of *varchar*. The model looks like a three-dimensional reference system, as shown in Figure 3(a). Accordingly, all post data are stored in the key-value store unit of the Data Pool, following the key-value storage mechanism, that is, the data are stored in rows, and each row means a post of the forum application. It is worth noting that the keys of each post here are referenced from its relevant basic item and the column families are derived from relevant dimensions. Figure 3(b) shows that there are rows of posts, and every key as well as every column family in each row in the Key-Value Store unit can find their corresponding counterparts in the Relational DB unit.

Advantages of the key-value based data model are summarized as follows: (1) it supports data models and metadata storage, so that other applications or the third-party algorithms can access the data without providing additional information from their master applications; (2) the data from *Dimension* acting as the indexes of the Key-Value Store unit will facilitate value searching and processing.

IV. SERVICE ORIENTED OUTSOURCING PLATFORM ARCHITECTURE

Based on the Data Pool, in this section we proposed a service oriented platform architecture to build an outsourcing application platform. The primary characteristics of the architecture include: (1) it offers a business-independent framework for business-oriented data-intensive application development; (2) it supports data integration from existing enterprise applications and current popular social networking services; (3) it provides pluggable outsourcing services and algorithms for third-party engineers, enabling crowdsourcing software development.

Figure 4 depicts a concept model of the architecture, which is shown as a pie chart that consists of 4 key components. Besides the Data Pool, the kernel of the architecture, the architecture also includes Data Adapters, Algorithm Library and App Keeper, interacting with applications through standard application programming interfaces (APIs).

The Data Adapters provide various channels for different data sources to store data, and offer a series of commands to implement data operation, such as create, update, retrieve and delete. As mentioned before, for those heterogeneous and diverse data to be processed, the platform should provide the ability to integrate these data from different data sources. Under normal circumstances, using a standard interface of the Data Pool can meet most of storage needs, and a majority of systems can push their data into the Data Pool directly. For some legacy systems or external systems, such as...
Facebook or Twitter, specific channels are needed to pull data available from them and push these data into the Data Pool. Nowadays, our platform provides some frequently-used adapters, and any enterprises can develop a specialized adapter by themselves, if needed.

The Algorithm Library is used to provide various algorithms for online analytical processing and data mining based on the implementation of core algorithms. It frees the users of the platform from implementing data parsing and processing, metadata management and other tasks, that is, such tasks can be undertaken by the existing algorithms in the Algorithm Library. As an important component of the business independent platform that supports the vast majority of programming languages and data types, the Algorithm Library is designed to be pluggable so that it allows the plugins of necessary third-party algorithms gathered from the Internet.

Based on the Data Adapters and the Algorithm Library, users can develop a variety of business-oriented applications. On one hand, these applications can store their data in the Data Pool; on the other hand, these applications can also mine and analyze the data stored on the platform using those algorithms in the Algorithm Library. Note that, these operations should be authorized and monitored. The App Keeper is the controller of the applications developed on the platform. It provides authorization and authentication services for all applications, records every operation of the applications, and makes these operations compliance by audit. In summary, the main goal of the App Keeper is to ensure the safety and security of the data used on this platform.

In order to incorporate other systems or be integrated by more applications, we selected REST (Representational State Transfer) rather than WSDL (Web Services Definition Language) as standard API style to build an open platform. As a lightweight architecture style of Web services, REST is increasingly popular in recent years after it has been introduced and defined in 2000 by Roy Fielding in his doctoral dissertation at UC Irvine [11]. Basically all of the well-known sites or platforms, such as Google GAE (Google App Engine) and Amazon AWS (Amazon Web Services), have chosen Restful Web services as one of their standard open services. Therefore, the interfaces of the Data Pool and Data Adapters, the algorithms in the Algorithm Library and the functions of the App Keeper are all used or required to use Restful Web services as standard interfaces, so as to help developers facilitate developing data-intensive applications according to their own needs.

V. EXPERIMENT AND RESULT

A. Motivation to develop sentiment analysis service

Sentiment analysis, also known as opinion mining, for product reviews is a new research and application topic within the field of big data, which is beyond traditional business intelligence. Generally speaking, the aim of sentiment analysis is to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. Identifying a series of potential features in a large number of product reviews collected from a product forum or other sources, extracting opinion expressions about those features and expressing users’ emotions about the products will facilitate decision making on product design and development [12]. To the best of our knowledge, this technique has been successfully applied in many application areas. For instance, many useful learning algorithms and public APIs have been proposed to predict popular opinion (e.g., presidential election, emergency treatment, etc.) based on the data from Facebook, Twitter, and other social media sources.

As one of the biggest software vendors in China, Kingdee provides dozens of various enterprise management software and on-line services for large-, medium- and small-sized enterprises in the Asia-Pacific region. To further improve the quality of software services according to users’ feedbacks, we attempted to investigate sentiment analysis for six series of Kingdee products with the help of universities in China, which is time-consuming and fund-costing.

B. Data collection

Based on the application platform we proposed, we have recently developed a sentiment analysis application/service for our products. Unlike the investing in consumer support centre to gather user information, our application collects the information from product forum and social media inside and outside an enterprise. For this experiment, we take Kingdee Club [13] and Sina Weibo (micro-blog) [14] as examples, and we plan to mine the information of product defects and service improvements from them, and to cluster similar information for the related products.

Kingdee Club is an official product forum that provides the cyberspace for users themselves to communicate freely with each other based on a core BBS service. In many cases, Kingdee users would like to focus on particular questions of the related products, and their posts or replies always express their preferences, identify product defects, point out solutions to specific problems, or provide useful feedbacks. For some bestsellers, e.g., EAS (Enterprise Application
Suite), K/3 and KIS (Keep It Simple), thousands of posts and replies with short texts, described in natural language without any data structure, are created in one day, and their topics often range from different functional modules to UI appearance. For our experiment, we collected over 200 gigabytes (GB) data from the Kingdee Club.

Akin to a hybrid of Twitter and Facebook, Sina Weibo is the largest social networking website in China, which attracted hundreds of millions of people throughout the country to share the information about sentiment, thought, whereabouts and even their most personal interest, also including countless comments about our and competitor’s products. About 100 million messages are posted daily on Sina Weibo. For our experiment, we carefully gathered all the information related to the official Weibo of Kingdee from its birth to the present.

Figure 5. Business process of the sentiment analysis application for Kingdee products

C. Process of sentiment analysis service

Figure 5 presents the business process of the sentiment analysis application for Kingdee primary products, which includes three steps and each step is implemented based on a component of our platform. Especially for the step of data analysis, common algorithms and services, such as Chinese word segmentation, stop word removal, LDA (Latent Dirichlet Allocation)-based topic clustering, etc., can be found or plugged in the Algorithm Library. The application developed on the platform offers top hot words and emotional curve services for business decision makers in an enterprise, provides them novel business insights, and helps them diagnose technical problems and make the right decisions for those problems.

For example, in natural language processing, a topic model is a type of statistical model to find the abstract "topics" that occur in a collection of documents, and LDA is an example of topic model and was first presented as a graphical model for topic discovery [15]. In order to develop sentiment analysis service for those Kingdee users, we implemented an improved approximate Gibbs sampling algorithm for LDA [16] based on our experimental data. The pseudo-code of the simple algorithm is presented in Figure 6. It directly maps LDA to a distributed processor setting, where Gibbs sampling is performed over local data followed by a global update of topic counts. It is worth noting that the algorithm can simultaneously execute LDA Gibbs sampling on each of the $P$ processors when distributing the data and parameters across processors. Hence, it is suitable for dealing with the sheer amount of data in our experiment.

Algorithm Simultaneous LDA Gibbs sampling
input: the input of LDA-Gibbs algorithm, processor vector $p$
output: the output of LDA-Gibbs algorithm
repeat for each processor $p$ in parallel do
  Copy global counts: $N_{wk} \leftarrow N_{wk}$; Sample $z_{ij}$ locally: LDA-Gibbs-Iteration($x_{ij}$, $z_{ij}$, $N_{kjp}$, $N_{wk}$, $\alpha$, $\beta$);
end for
Synchronize;
Update global counts: $N_{wk} \leftarrow N_{wk} + \sum_p (N_{kjp} - N_{kjp});$
until termination criterion satisfied

Note that $N_{wk}$, $N_{kjp}$, $x_{ij}$, and $z_{ij}$ means the count of words assigned to a topic, the count of topics assigned in a document, the $i$th observed word in document $j$, and the topic assigned to $x_{ij}$, respectively.

Figure 6. Simultaneous LDA Gibbs sampling algorithm [16]

D. Experimental results

The source codes of the application were implemented in Java, and all experiments were performed on a cluster of a storage server and several application servers, which running Linux with Intel (R) Xeon (TM) 2 CPU E5-2609 (2.5 GHz), 128 GB RAM and 1.8 TB hard disk. These servers are connected by an internal Gigabit Ethernet of Kingdee.

Kingdee EAS software, a leading product for large-sized enterprises, was selected to be an experimental subject. We collected and processed the data of users’ sentiments about
this product within a month, and the line graph of the result is shown in Figure 7.

Figure 7. Sentiment analysis result for Kingdee EAS

It is obvious from the figure that there exist turning points in both indexes of positive emotion (blue line) and negative emotion (red line). Interestingly, the index of positive emotion (blue line) of users declined in the first week, and then increased in the subsequent three weeks. On the contrary, the index of negative emotion (red line) increased slightly in the first three weeks, while began to decline sharply in the fourth week. Besides, the list of Top 5 hot words indicates that users were satisfied with this product because of its high performance, better user experience, ease of use, and other innovations. However, some users were unsatisfied with it due to its low performance, inconvenient operation, occasional quality accidents and bugs. According to the graph, we conclude that our product operation team had observed these problems in the first two weeks and proposed various remedies for them. Fortunately, these remedies worked well and achieved the desired effect.

VI. RELATED WORK

Searching Google for the phrase “big data application platform” returns 54 million results, which indicates a tendency that more attention is being paid to big data application platform in industry. Engineers want to develop such platforms that provide frameworks, services or resources to make the development of applications much simpler and quicker to meet rapidly changing requirements. Investigating current so-called platforms for big data provided by some well-known vendors, we found that they could be classified into different categories.

The first category is known as hosting platform. AppFabric, founded by some former co-founders from Teradata, Yahoo, Facebook and Microsoft, belongs to this category. It allows developers to deploy, scale and manage big data apps in and outside the firewall [17]. The second category is known as integration platform. IBM launched a big data platform for enterprises not long ago, whose architecture is based on Apache Hadoop system and Data Warehouse [18]. IBM claimed that it is able to integrate and manage the full variety, velocity and volume of big data, and can provide a development environment for building new analytics applications in a secure manner. The third category is known as analysis platform. For example, LucidWorks [19] is the trusted name in Search, Discovery and Analytics, delivering the only enterprise-grade embedded. Moreover, LucidWorks gives developers the key to building business-critical search apps quickly, reliably, and at enterprise scale.

It is undeniable that the above-mentioned application platforms and other similar products or open source software could perform well within their respective application domains. Some of them even cover partial functionality of the proposed platform. However, there is a lack of an overall service-based solution and the relevant validations and demonstrations that can help guide enterprises to develop their own big data applications based on their current infrastructure and IT systems [20]. To the best of our knowledge, most of existing platforms are unable to support algorithms and applications outsourcing, and seldom of them are highly optimized for general use.

VII. CONCLUSION

To meet the requirements and business characteristics of modern enterprises in the era of Big Data, this paper proposed an outsourcing platform for data-intensive application development. Unlike the traditional relational database predominated business intelligence software, each component of our platform is designed for processing various and massive data, and the platform is also developed for the sake of open source. For the kernel of the platform, there is a hybrid database, named Data Pool, which is designed to store both structured and unstructured data with a unified access interface. Based on the Data Pool, the Data Adapters are used to connect other systems with different interfaces or protocols, to collect data from these systems,
and to store these data into the Data Pool; the Algorithm Library provides various algorithms as services for data processing, and it is designed to be pluggable so that the third-party software engineers can submit their customized algorithms to the library to serve other applications. Any applications authorized by the App Keeper can invoke APIs of the platform to develop data-intensive applications.

In order to validate the platform, this paper also developed a sentiment analysis application for Kingdee products based on services provided by the platform. The sentiment analysis application is used to identify a series of potential features in vast amounts of product comments from Kingdee Club and Weibo.com. We extracted the top five opinions about those features and expressed the emotions of users about the Kingdee EAS software, thus facilitating correct decision-making on product design and development. Experimental results prove that the platform can effectively store both structured and unstructured data in the key-value based hybrid database, and that its service-oriented outsourcing architecture does facilitate data-intensive application development.

ACKNOWLEDGMENT

This work is supported by the central grant funded Cloud Computing demonstration project of China, R&D and industrialization of the SME management cloud (Project No. [2011]2448), hosted by Kingdee Software (China) Company Ltd., under the direct of National Development and Reform Committee of China; the National High Technology Research and Development Program of China (863 Program), under the Grant No. 2012AA040915; the electronic information technology development foundation of Ministry of Information Industry (Grant No. GXBC [2011]506); the project of Guangdong Science and Technology under the Grant No. 2012B070200014; the Shenzhen high-tech projects under the Grant Nos. CXA201105060081A, CXZZ20120618164052978, and KC2012JSJ0021A; the National Science & Technology Pillar Program of China under the Grant No. 2012BAH07B01; the Youth Chenguang Project of Science and Technology of Wuhan City in China under the Grant No. 2014070404010232; the construction fund of National Engineering Research Center for Supporting Software of Enterprise Internet Services.