A Design of RESTful Style Digital Gazetteer Service in Cloud Computing Environment

Song Gao[1], Hao Yu[1], Yong Gao [1], Yinle Sun[2]
1. Institute of Remote Sensing and Geographical Information Systems, Peking University, Beijing, China 100871
2. School of Geography and Remote Sensing Science, Beijing Normal University,
Beijing, China 100875
songgaogeo@pku.edu.cn

Abstract—**The geographical knowledge-informed digital gazetteer service (KIDGS) is a standard web service that provides unified XML-based access interfaces for various applications. In order to solve the problem of the raising data storage crabbed from millions of web pages and increasing query efficiency requirements for different levels of requests, we analyze the characteristics of cloud computing application and former architecture of KIDGS, and then design resources-oriented architecture of Cloud KIDGS in the RESTful style. Some key techniques in the process of MapReduce and design of RESTful services are also discussed. At last, a number of future development and their potentials for both commercial applications and scientific research are assessed.**

Keywords-cloud computing; digital cazetteer service; RSET; MapReduce

I. INTRODUCTION

A. Digital Gazetteers

Digital gazetteers manage place names, footprints, types, and aliases of geographical features. Several digital gazetteers have been developed such as the Alexandria Digital Library (ADL) gazetteer [1, 2], the Getty Thesaurus of Geographical Names (TGN) and the geographical knowledge-informed digital gazetteer service (KIDGS)[3]. The amount of entries of these digital gazetteers is giant. For example, ADL's recorded descriptions of named geographic places supported about 5.9 million features and TGN contains around 895,000 records, including nearly 1,115,000 names, place types, coordinates, and descriptive notes related to art and architecture. Also, The KIDGS at the Peking University which was designed to provide the foundation for textually represented geographical knowledge contain about 6,000,000 geographical terms including not only the common coordinated place names but also textual latently expressed locations such as IP address and telephone numbers. Digital gazetteers have proliferated in number and sophistication with the popularity of location-based services and web page searches.

TABLE I. DIGITAL GAZETTEERS COMPARISON

Digital Gazetteers	Content	Amount
ADL	place names, footprints, types, and alias	5,900,000
TGN	place names, types, coordinates, and descriptive notes related to art and architecture	895,000
KIDGS	place names, IP addresses, etc.	6,000,000

Supported by Research Fund from Major National S&T Program of China (Grant no. 2009ZX07528-004), National Natural Science Foundation of China (Grant no. 40928001, 40701134, 40771171), and National High-tech R&D Program of China (Grant no. 2007AA120502).

Rather than publishing geographical knowledge-informed digital gazetteer service as an open platform for end users, KIDGS focuses more on dealing with documents associated with locations and processing spatial assertions based on the web ontology language (OWL) support. It provides a unified XML-based access interface for various applications based on the web service definition language (WSDL) and the simple object access protocol (SOAP). Since calculating some spatial relations between two places sometimes are time consuming and many query requests may occur simultaneously, it will decrease the efficiency in executing queries. Unfortunately, the existing architecture of KIDGS is not appropriate enough to fulfill different levels of requirements from end users and also is not an open APIs platform for some advanced Web 2.0 applications such as REST, RSS, AJAX, etc.

B. Definition and Characteristics of Cloud Computing

Recently, cloud computing is a hot topic all over the world. It is considered as a sharing architecture of the IT trends, in which a third party provides highly scalable, reliable on-demand software, hardware, and infrastructure services with agile management capabilities [4], while users consumes and pays for it on demand. Cloud computing has many advantages such as high efficiency, on-demand availability, flexibility, dynamically-scalability, security and low-cost. Three core options comprise the service model within the cloud computing environment, that is, software as a service (SaaS)、platform as a service (PaaS) and Infrastructure as a Service (IaaS)[5]. Firstly, SaaS is an internet-based software application model to provide services rather than as traditional software, such as Saleforce.com. Secondly, PaaS is an application service platform on which developers can build and deploy their applications remotely such as Google App Engine Platform and Microsoft's Windows Azure platform. Thirdly, IaaS primarily encompasses the hardware and infrastructure for computing power, storage, operating system resources such as the Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Simple Storage Service (Amazon S3).

The architecture of cloud computing service and deployment models offer are a key research areas of geographical information systems (GIS) platform solutions. Unfortunately, researches on GIS application design in cloud computing environment are limited up to now. A stand Digital Gazetteer that focuses textually described geospatial features—points, lines, and polygons is the quintessential geographic information systems [6]. Therefore, the rapidly spread digital

gazetteers services for multiple levels of users on the Internet are expecting such a powerful large-scale distributed computing paradigms.

In this paper, we design a architecture of KIDGS, which considerates following points and problems：1) Designing a flexible architecture for KIDGS based on cloud environment to manage, abstract, and virtualize geospatial data, provide computing power for storage, queries, platforms, and services and be dynamically-scalable. 2) Meeting the increasing data storage requirements with limit hardware storage resources; 3) Mapping and reducing one query request of the gazetteer service; 4) Dealing with simultaneous queries; 5) Encapsulating an abstracting entity that delivers different levels of services to customers on demand on the Internet.

II. THE SYSTEM REQUIREMENTS

A. On-demand Data Storage Provision

The KIDGS manages two types of geographic features: artificial geographic features such as districts and phisical geographic features such as rivers. In addition to these features, some geographic clues are also associated with particular locations in geographical space, such as phone numbers, postcodes, and IP addresses. The KIDGS plays an important role as the basis of a range of services, including places-finding, spatial cognition, automatic georeferencing of text (geoparsing), and geographic information retrieval (GIR). With the motivation to represent plain text-based geographical knowledge better, KIDGS is going on to reference more than 100,000,000 web pages in the future and to solve the problem of vague places. The amount of place names data will be booming. There is another problem, the gazetteer services will stop if the data server is breaks down with flaws since there is only one data server. Traditionally, the data service center backup data frequently to another server in which make up some request deficiency. However, this strategy is fairly ineffective for high level and reliable data storage requirements of multi-users'. Furthermore, scalable storage is also needed because of the uncertainty in the demands of tenement for KIDGS. Cloud storage has a characteristic that many virtual instances can be "synched" to the cloud and distributed to other virtual computers, at the meanwhile the storage is delivered on demand. In this case, the actual gazetteer storage space can be thin provisioned and billed for based on actual usage so that the services can meet the capabilities required for different applications.

B. Single Query Efficiency and Simultaneous Queries

In KIDGS, the query execution strategies are composed of managing addresses and aliases, dealing with types, manipulating spatial relations and calculating matching degrees at each of above aspects. Since most spatial relation calculations and similarities' computing associated with names, types, and relations are time consuming, they will decrease the efficiency in executing a single query. Quite a few of query processes are searched by indexes on big joined table and each one only takes a few seconds to run on its own. However, when running them all as large script requests simultaneously, the server may freeze and halt for very long time.

If KIDGS is to be published as a basic service, its performance is still not good enough. Many applications such as GIR-oriented services require frequent access to gazetteers and fast response. The average of 10s for each treatment will greatly hinder the process of query. Therefore, increasing the executing rate and efficiency of query should be taken into consideration.

C. Multiple-level Servicess

With the fast development of LBS mobile applications and web search engines which are based on key words segmentation, such as Google, Microsoft Being and Baidu Inc., the needs of geographical information retrieval about tour places names and other related information increased greatly. However, web users usually still need select and pick useful items time after time from a mass of results, and some of the search results are little relative with which they really need. Since many fields information has certain relation, a set of "property assertions" that relate individuals to each other are knowledge-informed. Unfortunately, such assertions hardly could be asserted or searched by key words segmentation based.

Using the ontological model of common sense geographic world, the KIDGS provides an explicate framework for representing geographical knowledge representation, which is also an important aspect with growing requirements for web map search. For some flexible search requirements, the KIDGS should split up the original single service into small components and resources, each of which provides a general, scalable, and functionally independent service. This results in a highly flexible architectural framework which can serve as a vehicle. It is useful for further investigation of many issues relating to open geographical knowledge related hypermedia systems.

III. SYSTEM DESIGN

A. Architecture of Cloud KIDGS

In order to achieve above multiple requirements, we should consider three things: the efficiency of the response to query requirements on place names and spatial relations, convenient resource-based open APIs, and on demand scalable services. Before deploying and publishing the geographical knowledge-informed gazetteers services, we don't know how many users and requests there will be. With the cloud computing (Hadoop) framework, the open platform forms a reliable, scalable, distributed, and collaborated environment for providing geographical knowledge-informed gazetteer services.

Considering the difference between open GIS research platforms and enterprise business GIS platforms which prefer a private cloud environment, we think a hybrid cloud solution is commonly bound together. It is authored by proprietary technology which will only be embraced by enterprise computing in the future as standards are developed. The structure of Cloud KIDGS platform on vertical level can be divided into four layers(Figure 1)：the distributed data storage layer, the virtualized unified resource layer, the platform management layer, and the server application layer. They provide a set of uniform and simple RESTful style services.

1) Distributed Data Storage Layer: It contains the raw hardware level computer resources and distributed storage resources cluster, which provide storage of gazetteer metadata with HDFS and data bodies including the OWL knowledge base and spatial databases of all nodes. Each node of spatial database has a complete gazetteer database and cooperate with others when executing GET, PUT, UPDATE and DELETE tasks. For KIDGS, query GET tasks are more frequently used than others. Therefore, the GET operation is disassembled into many slave databases while the PUT operation will only be executed by master database. But all slave databases will refresh synchronously after the master update and return a uniform transparency interface for upper layer. To achieve load balance, all single gazetteer spatial database connections configure a maximum request number. When the request limit is reached, the new request will be automatically transferred to other available data nodes.

2) Virtualized Unified Resource Layer: This layer contains abstract gazetteer resources which may be pooled from any physical dissociative data storage and encapsulated by virtualization so that they can be exposed to upper layer (the platform management layer) and end users as the integrated uniform resources, which could be requested dynamically and scaled automatically. That means we do not need one high performance and reliable server machine such as IBM z series, but instead we can use a virtualized cluster of several common personal computers to substitute for it.

3) Platform Management Layer: It is a system administration and auto task deployment platform based on the framework of Hadoop, which provides functions such as security authentication, task schedule control, virtualized resource cataloging for collections of place name query APIs, geographical knowledge query APIs and geographical knowledge reasoning APIs. The important role of this layer is to decide when and how the services resources will be started processing and shut down by scripts, which controls the whole gazetteer services life cycle management. Moreover, the platform management also contains MapReduce programming model and an associated implementation for processing the spatial queries.

4) Server Application layer: The Server Application layer is the medium of interaction between users and the platform. Cloud KIDGS can be flexible accessed through standard HTTP requests based on Unified Resource Identifier (URIs) from various applications on different platforms.

B. Resources Oriented Architecture of Cloud KIDGS

In the cloud environment, the server can typically provide an interface that allows the deployment and management of virtual gazetteers get into their cloud platform. During the lifecycle of these gazetteer instances, the amount of resources allocated to these instances and the storage can be managed through these interfaces. Considering the main stream of high frequent simple web service request and convenient open APIs for further mashups application, this interface is based on Representational State Transfer (REST) HTTP operations. REST is a web application style for design and development, including a series of design criteria, and it determines how to make a well-defined process to move forward in the web and was first introduced by the Roy Thomas Fielding in 2000[7].Without the overhead of many similar protocols, the REST approach allows users to easily access their services. Each resource is uniquely addressed by using a Uniform Resource Identifier (URI) and is stateless, which is a set of operations based on HTTP such as create (POST), retrieve (GET), update (PUT) and delete (DELETE). And each HTTP request is complete and isolate.

Compared with the SOAP-based service, RESTful services are flexible and ease to use. Even without using any programming language, users can access RESTful services which have better performance, cache support, scalability, and statelessness [8]. The resources oriented architecture comprises resources, URIs, resource representations and the mutual linkage relationships between resources. In Cloud KIDGS, the gazetteer resource is addressable through URI, which represents both the name and the address of resources. For KIDGS URIs, we consider three important principles which include expressing the hierarchy with path variables, adding punctuation to eliminate misunderstandings, and using query variables for inputs of arithmetic.

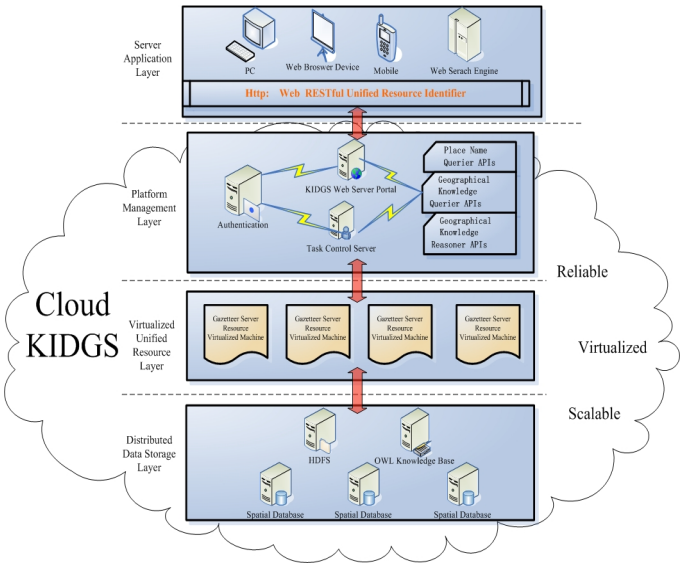


Figure 1. System architecture of Cloud KIDGS

```
http://geosoft/GPNS/geofeture/river/child1;child2
http://geosoft /GPNS/place?name=Beijing&type=city
http://geosoft /GPNS/search?type=
restaurants&reference=Beijing&spatialrelation&east
```

Figure 2. Name the resources' URIs in Cloud KIDGS

The working process of the cloud KIDGS platform is as follows: the end users from different devices or intermediate service enterprises send place name query requests through URIs. Then the requests is accepted by the platform management layer which completes the authentication, analyzes parameters, and processes the request with geographical Map or MapReduce computation via distributing them into the virtualized resourced pool. Finally, the results of processed place records with place name, type information

footprint, and match degree are returned to the client in an XML format or other structure documents (Figure 3).

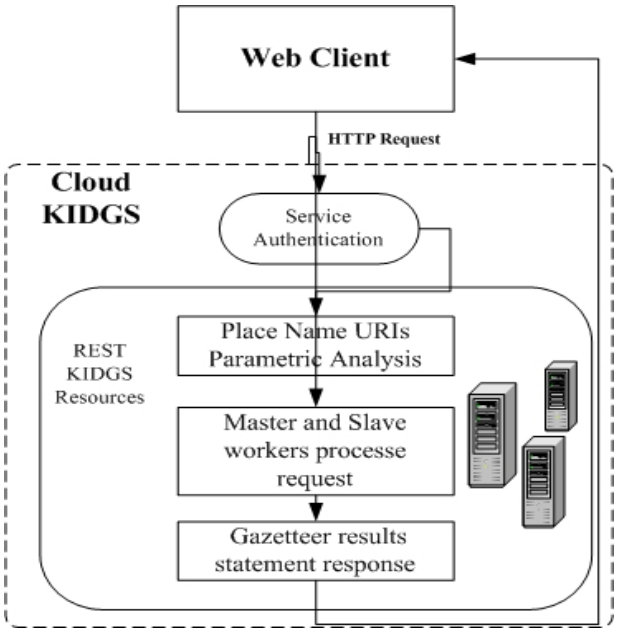


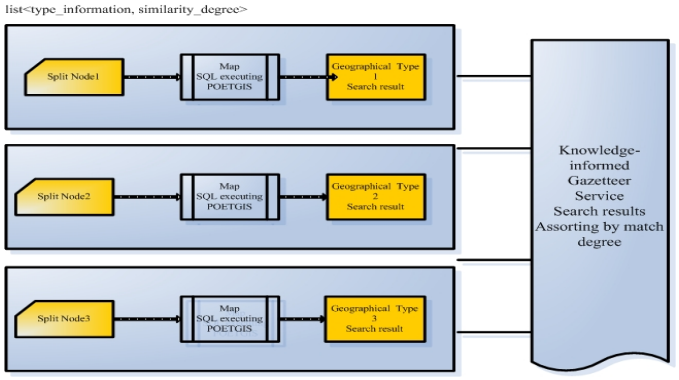
Figure 3. RESTful service response mechanism of cloud KIDGG

IV. KEY TECHNIQUES AND STRATEGIES

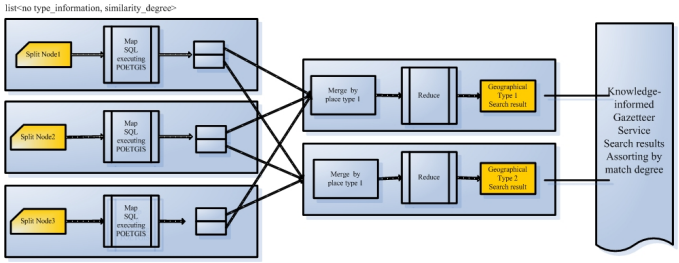
A. MapReduce Programming Model

Developers can specify a query task into the map and reduce process. In the map process, the function processes a key/value pair to generate some intermediate key/value pairs, while the reduce function merges all intermediate values associated with the same intermediate key in the MapReduce programming model[9]. In KIDGS, we take into account place name type postfixes in implementing a digital gazetteer. The postfixes represent the type of a place, and as a geographical knowledge expression which is stored in Protégé as a value property for each OWL ontology class. The metadata table contains a field which indicating whether place names are stored with postfix or not for a certain PostGIS database table. Therefore, there are two executive strategies cases when processing a place name query. One is that users query place names of a certain type or the place names with explicit type postfix such as "city", "mountain", "river" et.al. KIDGS will determine which related tables should be queried. The other one is that users do not provide any type information and the place names have no or a vague postfix or none, and then KIDGS will go through all tables or search among subtypes or sibling nodes to get all subtypes of the provided type referencing the ontological model of geographic knowledge.

For the first query case with type information and spatial relation, such as a query named "restaurants in the east of Peking University", the following sequence of query actions occurs in KIDGS(Figure 4(a)): 1) According to ontology tree of type found query place name type ("restaurants")and similar types(park, performance site, sport facility, etc. by the distance calculation. The task control server (Master) returned a list<type_information, similarity_degree>. 2) Finding the coordinates of reference place("Peking University") through a big category feature footprint table. 3) The master splits the input query list<type_information, similarity_degree> into M pieces of typically 8 megabytes (MB) per piece (configurable by the user via an optional parameter). This is the Map procedure. 4) Each slave spatial database node executes query in accordance with these different type by generating SQL statement with spatial relation, and then calculates the match degree. 5) Each node returns the summarized match results sorted by high to low match-degree and then transmits the results as a response resource interface to the upper layer.



(a) With no reduce tasks



(b) With multiple reduce tasks

Figure 4: MapReduce process cases in KIDGS (a) and (b)

For the second query case with no type information, such as the query "Huang Shan", which could be city name or natural feature, the following sequence of query actions is a little different(Figure 4(b)): 1) Because we need go through all spatial database tables , the input spatial data tables are split into M pieces of typically 16 megabytes (MB) per piece (configurable by the user via an optional parameter). 2) The master picks idle slave nodes as workers and assigns each one a map task for executing the basic query SQL and calculating the match degree for different type (such as "city", "town", "mountain" etc.) Then the locations of these buffered < type_information, query name >pairs in the memory are passed back to the master file , who is responsible for forwarding them to the reduce process workers. 3) When a reduce worker is noticed by the master with locations of buffered query pairs, it uses remote procedure calls to read the buffered data from the slave node spatial database map results. 4) The reduce worker summary the query results by <key, value> pairs, which is specifically <place name with type information, match degree >, such as <Huang Shan city, match

degree> and <Huang Shan mountain, match degree>. 5) Each reduce node transmits appropriate results of one type information as a response resource interface to the upper layer, which union all the types and summaries and provides the result for users. Whichever search occurs, the user should write similar code to the following MapReduce pseudo-code as shown in Figure 5:

```
map(String key, String value):
// key: Place Type Information Table
// value: query names
for each query name w in value:
Emit Intermediate(w, "1");
reduce(String key, Iterator values):
// key: a place name with type
// values: match degree
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

Figure 5: MapReduce pseudo-code

B. Resource Design for RESTful APIs

When an HTTP request reaches the KIDGS context, KIDGS will compare the request URI and resource configuration files in the URI template, then find the resources profile, and get the best match for the request to the gazetteer resource class.

The resource layer contains basic gazetteer resource base classes, the KIDGS implementation classes, the parameters parser and the computing processors. It is the core of RESTful services. Each KIDGS RESTful service resource has a unique URI address, User accesses to resources of the URIs with standard HTTP action (GET, PUT, POST, DELETE, and HEAD) for resources to operate. Resources layer as a core element of service is mainly responsible for the content of the HTTP request process. First, it extracts parameters from the HTTP request, and uses the correct parameters parser to parse java object parameters. Second, according to the parameters, it returns a response message from the URI to get the type of response form. Third, it chooses the right coder to response the result code. At last, it returns the results back through the HTTP client environment.

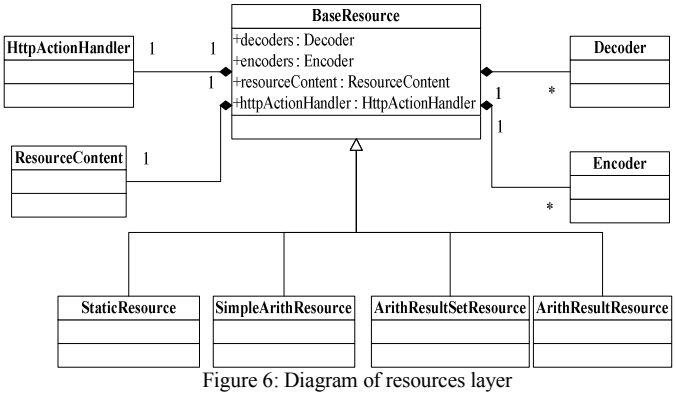


Figure 6: Diagram of resources layer

Figure 6 represent the diagram of resources classes. The BaseResource class is a root class on behalf of the resources to handle the HTTP request. Its handGet, handPut, handPost, handDelete, handHead methods are used to dealing with Get, Put, Post, Delete, Head of the request. There are four types of subclasses inherited the BaseResource class, including StaticResource class, SimpleArithResultResource class, ArithResultSetResource class and ArithResultResource class. 1) StaticResource class indicate the gazetteer data tables names, place names, dataset coordinates, and metadata. It supports logically get, post, put, and delete operation. 2) SimpleArithResultResource class is the resources of simple algorithm results, which through the URL query parameters to represent a specific algorithm result, such as:/Searchresult? name = Huang Shan & type =Mountain. This resource only supports GET and HEAD operation. 3) ArithResultSetResource class includes the complicated or compound resources of algorithms results sets, which are generally temporary resources will be automatically deleted after a request realization being realized. The algorithm parameters are transmitted by the request body. Such resources usually support only POST operation. This class supports spatial assertions consists of at least one spatial relation and reference object. 4) The ArithResultResource class is also algorithms results different from the SimpleArithResultResource class in that simple algorithms result resource is the result that algorithm parameters to get the information through real-time computing, while ArithResultResource obtains gazetteer search results from the cache.

In addition, the Encoder class can generate a gazetteer resource object presentation, and convert it to support different media types, different code languages, and different statements for character encode. A resource list Encoder is used to support the types of expression. This information is initialized when the resource is allocated the application. Encoder object can be extended to accept user's own statements, for example, a GMLEncoder can extend a place name query object with footprint to generate the GML expression resources.

Besides, the HttpActionHandler class is the Http action processor for resources. It contains all HTTP operations and interfaces which can realize custom Http action processing.

C. Security Strategy

Both data and system security problems are highly concerned for adoptions of the Saas platforms, the Grid computing and the Cloud computing environment. However, the security model for Cloud computing seems to be relatively simpler and less secure than the security model adopted by Grid computing in which only an authorized user can access resources [10]. For Cloud KIDGS security design, we can adopt some strategies as follows: (1) It relies on web forms over security socket layer (SSL) to create and manage account information for end-users.(2) Digital signatures are used to prevent tampering during gazetteer service transmission.(3) It use some data encryption algorithm for some sensitive geographical data and coordinates.(4) It make use of virtualized clusters for automatic regular backup.(5)An complete legible authorized configuration for service functions APIs and data is

taken into consideration.(6) For end-users, client-side input validation are needed for query request.

V. CONCLUTION AND DISCUSSION

In this paper, we address some emerging requirements of digital gazetteer services including on-demand spatial data storage provision, single query efficiency, simultaneous queries effectiveness and multiple level APIs of web application. Specifically, in order to solve with these problems, we design a scalable, reliable, virtualized architecture for KIDGS in the cloud computing environment, in which users can customize or mush up with flexible digital gazetteer services. It contains the server application layer, the platform management layer, the virtualized unified resource layer and the distributed data storage layer. Besides, we choose the resources-oriented RESTful stytle to deal with the multiple level requests. It supports standard Http operation requests with URIs. By encapsulating some complex geographical queries and spatial relation assertions into cached arithmetic resources, the service responses faster than before.

However, some key techniques for implementing cloud computing are not as mature as theoretic research. We will continue to study on how to configure a spatial-based gazetteer service and develop an appropriated platform for KIDGS.

ACKNOWLEDGEMENT

Thanks for Associate Professor Yu Liu at Peking University and Jun Xu at Resources and Environment Research Institute of Chinese Academy of Sciences giving us insightful suggestions and great help.

REFERENCES

[1] M.F. Goodchild and L.L. Hill, “Introduction to digital gazetteer research,” International Journal of Geographical Information Science. vol. 22(10), pp. 1039–1044, 2008.

[2] L.L. Hill, J. Frew and Q. Zheng, “Geographic names: the implementation of a gazetteer in a georeferenced digital library,” D-Lib Magazine. vol. 5, 1999. available online at: http://www.dlib.org/dlib/january99/hill/01hill.html.

[3] Y. Liu, R. Li, K. Chen, Y. Yuan, L. Huang and H. Yu, “KIDGS: A Geographical Knowledge-informed Digital Gazetteer Service,” in: Proceedings of 17th International Conference on Geoinformatics, Fairfax, VA, Aug. 12-14, 2009.

[4] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” in: Proceedings of IEEE Grid Computing Environments Workshop, pp. 1-10, 2008.

[5] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica and M. Zaharia, “Above the clouds: A berkeley view of cloud computing,” Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, 2009.

[6] J.T. Hastings, “Automated conflation of digital gazetteer data,” International Journal of Geographical Information Science. vol. 22(10), pp. 1109-1127, 2008.

[7] R.T. Fielding, “Architectural styles and the design of network-based software architectures,” PhD Dissertation. Department of Information and Computer Science, University of California, Irvine, 2000.

[8] L. Richardson and S. Ruby, RESTful web services. Sebastopol, CA: O'Reilly, 2007.

[9] J. Dean and S. Ghemawat, “Map Reduce: Simplified data processing on large clusters,” Communications of the ACM-Association for Computing Machinery-CACM. vol. 51(1), pp. 107-114, 2008.

[10] C. Kesselman and I. Foster, The Grid: Blueprint for a New Computing Infrastructure. San Francisco, CA: Morgan Kaufmann, 1999.