Third International Conference on Information Science and Technology
March 23-25, 2013; Yangzhou, Jiangsu, China

Hadoop-based Service Registry for Geographical Knowledge Service Cloud: Design and Implementation

Jianfeng Lin, Xiaozhu Wu, Chongcheng Chen and Yewei Liu

***Abstract*—The increasing maturity of Cloud computing technology brings a profound change to modern information service industry. With explosive growth of the web services, traditional service registration software encounters many problems on cloud service registration and discovery. In this paper, we design and implement a novel geographical knowledge service registry (GeoSC) to address the problems of organizing and managing vast amounts of heterogeneous knowledge services under cloud environment and support for fast, accurate service discovery. Based on open-source cloud computing framework - Hadoop stack, GeoSC adopted a non-relational data-storage model to support QoS-based and semantics-based service description, and provide efficient parallel service discovery. We carried out an elaborate comparison between GeoSC and traditional service registration software - JUDDI to validate web service registration and discovery performance improvement. The preliminary results indicated that, GeoSC performs better on service registration and discovery than JUDDI.**

I. Introduction

IN recent years, web services have become the primary means of network information exchange which promotes the prosperity and openness of web applications and APIs, because of their key features such as interoperability, reusability, deployability [1], [2]. Cloud computing redefines Web services to provide more than interoperating between different software applications but indicate those pattern that offer computing resource and computational framework as services [3], [4]. It leads explosive growth to the web services. Therefore an accurate and efficient cloud services discovery under the distributed heterogeneous environment has become a key issue.

Traditional services registries based on the UDDI specification encountered a bottleneck in performance in handling of massively distributed services discovery and Integration. Most of these services registries are highly complex and extremely difficult in specific implementation for users. The relational databases these registries used for querying among huge amounts of services from multiple sources take frequent "join" operation between tables which results in poor performance. Achieving low latency at high concurrency still is a huge challenge in massive multi-source service discovery in cloud era. Geographical cloud service registry described in this paper can be found as an effective cloud services solution to address the problems as stated effectively.

Geographic knowledge cloud is an intelligent computing platform which use cloud computing technologies as the foundation, integrating with the technologies such as artificial intelligence, knowledge management. It introduces spatial data mining (or knowledge discovery), distributed spatial decision support system technology into the cloud computing environment, making the cloud platform capable of deep spatial data processing and auxiliary spatial decision-making. Cloud Registry furnishes capabilities of geoscience services description, discovery, integration, service hosting etc. for data mining and knowledge discovery on the cloud platform.

This paper presents the Geographic Knowledge Service Registry Center (GeoSC), a new type of web services cloud services registry. GeoSC is designed to maintain cloud service registration information, to simplify the registration process of cloud services, to enhance service discovery capabilities, and to optimize the service discovery speed. In this paper, we described the technology based on virtualization and parallel computing framework to build the design and implementation of GeoSC. We run two experiments to compare GeoSC with JUDDI to demonstrate web service registration & discovery performance enhancement. The preliminary results indicated that GeoSC has much better performance than JUDDI.

The rest of this paper is organized as follows. Section 2 introduces the related open source technologies to build the geographic knowledge cloud registration center and the related work. Section 3 presents the system architecture of the registration center. Section 4 describes the implementation of the registry. Section 5 is devoted to the registration center of the cloud corresponding experimental results, and experimental comparison with Apache JUDDI [6]. Section 6 discusses our future work and concludes.

II. Related Work

Cloud computing provides a flexible, effective solution to meet the growing computing and storage demand of UCG website [7], [8]. The Geospatial data processing and mining can benefit from Cloud computing technology by utilizing its unlimited high-performance computing resources and storage on demand which can effectively promote the efficiency and

This work was supported partly by National Key Tech. R&D Program of China under Grant 2013BAH28F00, and by Fujian Sci. & Tech. program under Grant 2010I0008 and 2010HZ0004-1, and by EC FP7-2009-People-IRSES under Grant 247608.

J Lin, corresponding author, is with the Spatial Information Research Center of Fujian, Fuzhou University, Fuzhou, China (e-mail: ventlcc@gmail.com; phone: 13774506829).

C Chen is with the Spatial Information Research Center of Fujian, Fuzhou University, Fuzhou, China (e-mail: chencc@fzu.edu.cn).

X Wu is with the Spatial Information Research Center of Fujian, Fuzhou University, Fuzhou, China (e-mail: wxz@fzu.edu.cn).

Y Liu is with the Spatial Information Research Center of Fujian, Fuzhou University, Fuzhou, China (e-mail:liuyewei5@gmail.com).

ability of spatial knowledge discovery in large spatial dataset [9]. GeoKSCloud is a concrete realization of the geographical knowledge cloud platform, which is based on cloud computing technology to aggregate the wide distribution of various types of web services, including knowledge discovery services, spatial analysis, analysis services, integrated with geospatial ontology knowledge base, to provide highly scalable, extensible and intelligently geography problem-solving environment.

As a common method of web services of register, publish and discovery, UDDI had been widely used in both private and public web service centers [10]. Meanwhile GIS web services also use UDDI specification for service description and discovery. However, UDDI specification defines a four-tier XML Schema for providing the metadata of web services which actually makes its data structure cumbersome and complex. A lot of researches have been conducted on UDDI, a well-established service discovery protocol. The following is a brief outline of these works.

Ran introduced Web service QoS certifier to expand the UDDI model which added up two-tier authentication information and the publisher assertion structures to enhance the reliability of the service registry to discover services [11]. Based on QoS and customer performance metrics, Al-Masri et al proposed the Web Service Relevancy Function to build an intelligent web service registry [12]. M.Alrifai used the MIP (mixed the integer programming), a global optimization with local selection algorithm, to find the global constraint optimization services portfolio. Experimental results proved this method can efficiently improve discovery and registration ability of customer service [13]. Sycara proposed a services registry prototype named DAML-S/UDDI with enhanced capacity of semantic matching management services than the UDDI model while did not significantly reduce the performance of service registration [14]. Aguilera designed a semantic service matching discovery algorithm, it can retrieve semantic services and match any OWL entity directly without binding a specific ontology concept or a priori knowledge [15]. These efforts we hope to expand the UDDI protocol are compatible with the matching and retrieval of semantic information, they inevitably increase the UDDI complexity and do not bring a performance improvement, they are difficult to adapt to the massive service description, discovery requests.

Banerjee deployed a Distributed UDDI Deployment Engine on a grid platform to rendezvous multiple service registry and achieve a distributed service discovery mechanism [16]. Pastore implemented a service registry on the basis of Globus grid platform by integrating LDAP and UDDI. Some researchers also described the services submitted to the hierarchical database grid task search method [17]. However, the UDDI specification requires commercial entities to describe the service itself as it is standardized, complex patterns, at the same time it depends for sorting, viewing highly complex mechanisms of managed network services, the ultimate global registration center and not been successful [18].

III. THE ARCHITECTURE OF GEOSC

GeoSC follow the idea of open information system design, based on SOA architecture. It's standardized, modular, platform independent which providing a common language, unified communication and interaction way for heterogeneous environments. As shown in Fig. 1, GeoSC is designed into three-tier structure namely the virtual infrastructure layer, the cloud service engine layer and the core application layer. The main module to form different granularity of independent web services package is open to users. GeoSC prototype is implemented as five service center components by expanding the three-tiered architecture, including GeoControl Center, GeoAggregation Center, GeoService Center, GeoProblem Parser and portal of knowledge cloud (including the Web-based GeoKSPortal and mobile device-based GeoMobile Tool). Cloud service registry provides GeoSC the capability of geoscience service discovery, service registration and service hosting as the forerunner of data mining and knowledge discovery. Cloud service registry relies on its wide collection of geoscience services, as well as simple but efficient service discovery model to realize fast and accurate service discovery. It also supports services composition.

The virtual infrastructure layer is a dynamic, scalable virtual resources pool which provides dynamic services for various client applications on demand. The Cloud service engine layer is responsible for the core application layer to provide high availability, low latency, massively scalable distributed geo-management and analysis interface.

Cloud service engine layer built on top of the virtual infrastructure layer, provides service registry compute and storage components, such as service registration, service discovery, service hosted. The key point to improve the performance and scalability of the service registry is parallel framework.

GeoService Center, the core component of GeoKSCloud, supports registration and release of the third-party services (Geoscience process simulation and data mining services), and then generates directory system for algorithms and services. According to directory system, GeoService Center in turn taking over the search, mapping, combination and execution of problem-solving services for geoscience parser. GeoService Center has features for data safety backup, reliability, transparent file staging, and high fault tolerance, it utilizes HBase and HDFS to store registered service information and uploaded data. Parallel frameworks like MapReduce are used to provide computing capacity. The main modules of service registry system are independently packed as web services with various granularity. The GeoSC portal as well as Java wrapper library based on these service components facilitate different types of calls from heterogeneous platforms.

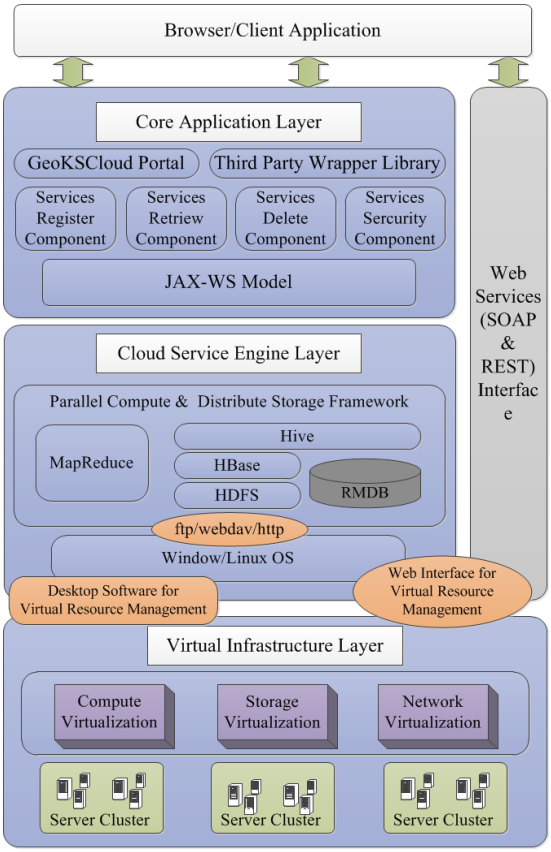

Fig. 1. Service registry architecture of GeoSC.

IV. IMPLEMENTATION

A. Data Storage

The design and implementation of data storage of registered services are based on APIs from the cloud service engine layer. Traditional computing can't accommodate high concurrent database read/write, efficient massive data storage/access, high scalability and high availability. Therefore, in 2000, the CAP theory was proposed by Brewer, who believed that a distributed system cant's meet all three needs (consistency, availability and partition fault tolerance) at the same time, but at least two of them. Gilbert then proved the whole conjecture in his experiment [19], [20]. Other scientists applied ACID with BASE transaction schema in distributed database, these systems achieving the eventual consistency to provide users high availability [21].

In order to deal with massive service information, cloud service engine adopts two strategies to store users' service information. Large files will be stored by data-flow accessing HDFS which follow the thought 'Write once and read many times is the most efficient mode'. Based on it, the realized middleware, which provide support for ftp and WEBDAV protocol, is encapsulated into web services, then transferred and managed as files. User applications can easily interact with service registry center through file system on Linux or Windows. In order to guarantee the high-performance processing, the large files will migrate to HDFS to take advantage of data locality and MapReduce will also be used properly. The integrity and reliability of service data stored in HDFS are ensured by redundant backup, and backups will not degrade the performance of the cloud services engine. Another strategy, a column-oriented database, based on HBase, integrating semantic, QoS, as well as classified information, is designed to adapt to the parallel environment, and provides service discovery capabilities. It meets the requirements of high concurrency, high availability, and non-relational network services (semantics). Semi-structured storage of the information of services classification and input/output semantics provide metadata for semantic-based service matching. QoS information, including service price, throughput, response time, reliability and quality, is properly stored and managed, and provides reference for SLA formulation.

QoS information indicators have different time-sensitivity. One is real-time QoS data such as service price, throughput, which change frequently overtime; The other is statistics such as the overall reliability of services and average availability, which gets from real-time QoS data. GeoService Center uses two column family QoSInstant and QoSStatistic to place real-time data and statistics data. We design this to reduce QoS information IO and improve speed of service response. Therefore, service users can combine the above mentioned QoS indicators and retrieve geoscience services according to their needs.

GeoService center allow registration and retrieveal of semantic web services, and utilizes a three column family (SPO, POS, OSP) RDF to store semi-structured RDF data. RDF is a W3C proposed network resource description standard, it labels network resources mainly by Subject, Predicate and Object [22], [23]. In addition, Registration center offers the Triple Pattern based SPARQL query and simple connection with BGP by the use of MapReduce. The data mode of Cloud service registry is using two denormalized tables to store the registered services description information, and defining the corresponding column family to standardize the SLA service evaluation based on QoS. Table I lists the information of published service entity and service entrance, service quality. In QosInstant column family, one or more columns can be used to define real-time (up to millisecond) service quality indicators. Table II is mainly used to describe service operation, service safety, and semantic information. Time stamp will record the service data versions of different periods, and the default data version that would be provided is the latest. However, client can check stamp index to assign historical data versions. In brief, the database is column free, and client and commercial entities can add specific columns according to their business needs.

HBase does not support JOIN operation between tables. Denormalized data mode can slove this problem, but it's likely to break consistency. In contrast, Materialized Views, which transfer the denormalized table of associated column family of GCServices and GCOperations to the table of GCView, can guarantee the final consistency of data Furthermore, Materialized Views also avoid the conflict when updating and editing user data, and effectively improve retrieve speed of complex services, but it will increase the

storage space.

B. References

GeoService Center based on parallelization service discovery model, implement and pack components of services of registration, discovery, delete, security, and then integrated with service interface of platform by adoption of JAX-WS model. What's more, the GeoSC portal and Java wrapper library is equipped to facilitate different types of calls from heterogeneous platforms.

TABLE I
GCOperation

RowKey	OperKey	Input: Type	Input: Count	Output	Time Stamp
M031124428	O110412067	String	3	String	T12409380 T12410217
M031453677	O110412068	Int	2	Double	T22532601

TABLE II
GCService

RowKey	ServiceName	Entity Neme	QoSInstant: response	Time Stamp
S110412067	HybirdDTIN	SIRC	10	T123834098 T125344097 T136600854
S110412068	Akima Equidistance	SIRC	12	T124054345 T125834911

The core application layer is responsible for packing computing and storage capability of service registry, and provides to client channel in forms of Web Service and GeoSC portal. We expose function and resource API of our cloud platform and a Java-based wrapper library, therefore the client applications can program directly by using these interfaces or classes; Service registry is integrated into GeoSC portal, and users can get functions of service registry, retrieve, call, hosting, and workflow design via WebUI.

Fig. 2 depicts the interaction between client/browser and the core application layer at various granularity. The service registry APIs are divided into three types: Function, Status Code and Data Type. *Function*, includes mainly single or multiple service operations such as registration, retrieve and hosting. The RetrievalAdvanced is not only a filter to different types of values, like keyword, category, QoS indicators type and corresponding values, but also the service retrieve of matching HBase database. If there exists default value, *RetrievalAdvanced* will delete the filter, and return client the value of the services that meet the remaining conditions. *Load* is used to import massive services to HBase through MapReduce framework. Load takes in charge of match data source input, and writes values (SequenceFiles, RTX, XML, WSDL) in corresponding columns of HBase. The Java Wrapper integrates *Load* and a Hive [24] project, in order to interaction between HBase and relational database.

Status Code collects the service and client response state as feedback information of service failure, and reflects the service availability and reliability.

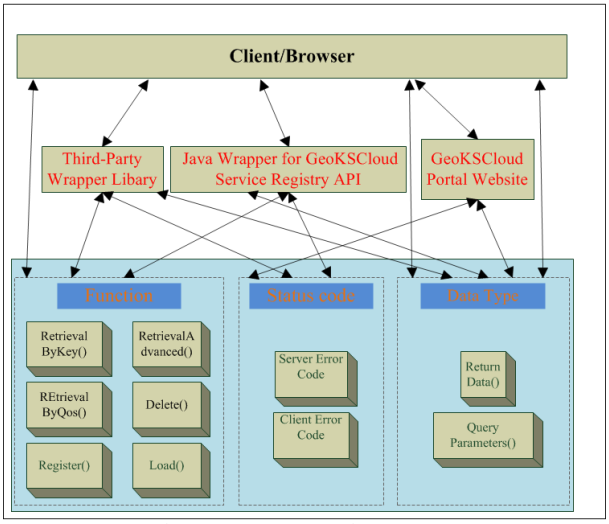

Fig. 2. GeoKSRC service wrapper.

```
<?xml version="1.0" encoding="gb2312"?>
  <serviceList pageCount="AllPages" currentPage="currentPage" pageSize="pageSize">
    <service id="ServiceKEY">
      <name>ServiceName</name>
      <description>ServiceDescription</description >
      <wsdl>WSDLAddress</wsdl>
      <score>UserScore</score>
      <classId>ClassID</classId>
    </service>
  </serviceList>
```

Fig. 3. Fragment of Services Retrieval XML

Data Type defines the input/output service data format, and users can access or manage service data only by implementing interface of *Data Type*.

The above sample code is used to invoke interface RetrievalByKey, and a service list that matches the keywords will be returned.

V. Experiment

The experimental environment is built on top of a four-node HP cluster of virtual machines and two-physical-node. Every virtual node with Quad-Core I7 CPU, 4G memory and 500G SATA hard disk. Operating system is Ubuntu Server 10.04 and its kernel version is 2.6.32. Hadoop version is 0.20.2 and HBase version is 0.20.6.In this cluster, one node as Hadoop master node, the remaining three nodes as Hadoop data node. We compare GeoSC with JUDDI 3.0.10, traditional UDDI software, which is open source implementation of UDDI standard version 3.

The experiments have two parts: services registration and services query operations. In first experiment, we had written same numbers of web services into both JUDDI and GeoSC. JUDDI using MySQL5.5, registration related to the operation on three tables; meanwhile write operation on GeoSC related to two HBase tables. As shown in Fig. 4: GeoSC has much

better write performances than JUDDI.

GeoSC reduces the table join operation by redesigning the web services registry model; and write performance is improved significantly by using parallel framework.

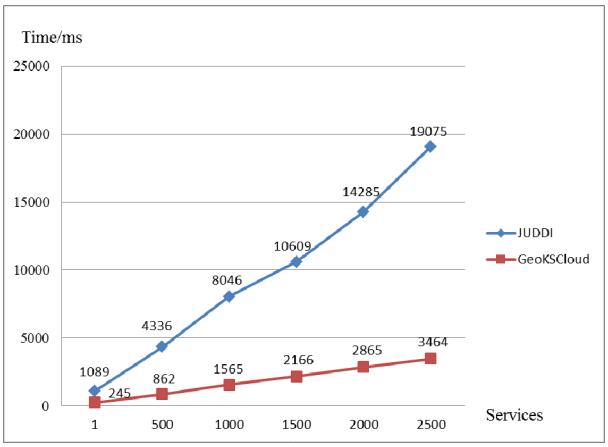

Fig. 4. Rate of services publish.

In the services query operation experiment, we had retrieved all corresponding web services of target keyword in both JUDDI and GeoSC, which matching rules are approximately matched with keyword. As shown in Fig. 5, GeoSC also has better read performance than JUDDI.

Due to web services data stored on GeoSC have been distributed on the clusters, and multiple machines can read at the same time while using MapReduce framework.

We implemented a web portal of GeoSC to provide users with a unified, intuitive, simple, user-friendly service description, registration, discovery interface which was developed on Html5 and CSS3 and other front-end technology. The web portal is shown in Fig. 6.

VI. CONCLUSION

Explosive web services development increasingly cause some processing issue of traditional registry which lack processing ability to meet with the demand of latency. This paper presents a novel cloud services registry-- GeoSC to deal with the massive numbers of web service description, registration and retrieveal. We design a new cloud service model and implement a new distributed cloud s services registry based on Hadoop technology stack. We carried out an elaborate comparison between GeoSC and open source UDDI software - JUDDI to validate cloud services data reading & writing performance. The preliminary results indicated that, GeoSC gains significant reading and writing performance improvement over JUDDI. The techniques and system in our work will provide a variety of users a powerful tool for further in-depth processing, and it has a broad application prospects.

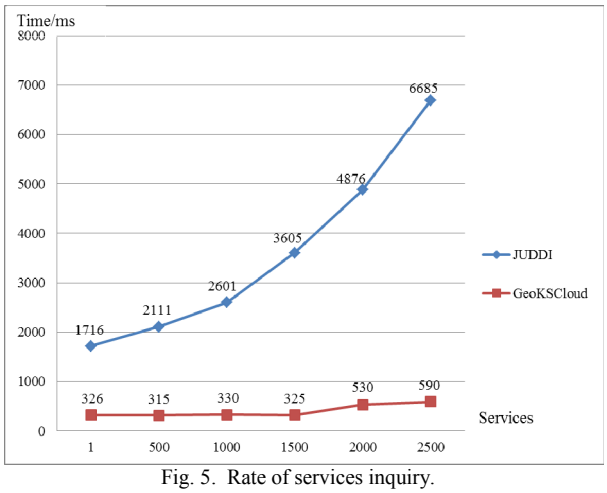

Fig. 5. Rate of services inquiry.

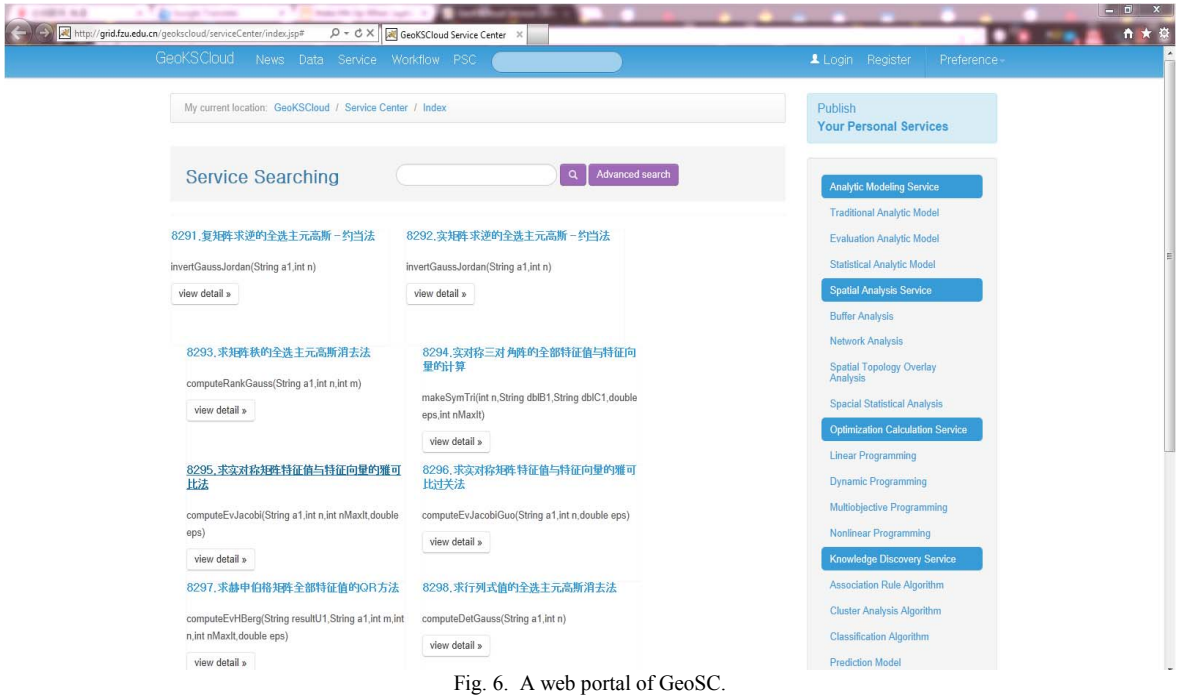

Fig. 6. A web portal of GeoSC.

REFERENCES

[1] B. P. Rimal, C. Eunmi, and L. Ian, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. Fifth International Joint Conference*, Seoul, pp. 44-51.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, *et al*., "Above the clouds: A berkeley view of cloud computing," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop*, 2008, pp. 1-10.

[4] Amazon. (2010, 2010-12-13). Amazon Web Services homepage [online]. Available: http://aws.amazon.com/.

[5] Wikipedia. (2011). Universal Description Discovery and Integration. Available: http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration.

[6] Apache. (2010). Apache JUDDI. Available: http://juddi.apache.org/.

[7] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the Cloud? An architectural map of the Cloud landscape," in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, Vancouver, 2009, pp. 23-31.

[8] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *Internet Computing, IEEE*, vol. 6, pp. 86-93, 2002.

[9] N. Alameh, "Service Chaining of Interoperable Geographic Information Web Services," *Internet Computing*, vol. 7, pp. 22-29, 2002.

[10] OASIS. (2010). UDDI 101. Available: http://uddi.xml.org/uddi-101.

[11] S. Ran, "A model for web services discovery with QoS," *ACM Sigecom Exchanges*, vol. 4, pp. 1-10, 2003.

[12] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proceedings of the 18th International Conference on World Wide Web*, Madrid, 2009, pp. 881-890.

[13] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated discovery, interaction and composition of Semantic Web Services," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 1, pp. 27-46.

[14] U. Aguilera, J. Abaitua, J. Diaz, D. Bujan, and D. Lopez-de-Ipina, "A semantic matching algorithm for discovery in UDDI," in *Icsc 2007: International Conference on Semantic Computing, Proceedings*, Irvine, 2007, pp. 751-758.

[15] Y. Yao, S. Su, and F. Yang, "Service matching based on semantic descriptions," in *International Conference on Internet and Web Applications and Services/Advanced International Conference*, Guadeloupe, 2006, pp. 126-126.

[16] S. Banerjee, S. Basu, S. Garg, S. J. Lee, P. Mullan, and P. Sharma, "Scalable grid service discovery based on UDDI," in *Proceedings of the 3rd International Workshop on Middleware for Grid Computing,* Grenoble, 2005, pp. 1-6.

[17] S. Pastore, "The service discovery methods issue: A web services UDDI specification framework integrated in a grid environment," *Journal of Network and Computer Applications*, vol. 31, pp. 93-107, Apr 2008.

[18] P. Krill. (2005). Microsoft, IBM, SAP discontinue UDDI registry effort. Available: http://www.infoworld.com/d/architecture/microsoft-ibm-sap-discontinue-uddi-registry-effort-777.

[19] E. A. Brewer, "Towards robust distributed systems," in *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, Portland, July 2000, vol. 19, pp. 7-10.

[20] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, pp. 51-59, 2002.

[21] D. Pritchett, "Base: An acid alternative," *Queue*, vol. 6, pp. 48-55, 2008.

[22] F. Manola, E. Miller, and B. McBride, "RDF primer," *W3C recommendation*, vol. 10, pp. 1-107, 2004.

[23] C. Franke, S. Morin, A. Chebotko, J. Abraham, and P. Brazier, "Distributed Semantic Web Data Management in HBase and MySQL Cluster," in *Cloud Computing (CLOUD)*, Washington DC, 2011, pp. 105-112.

[24] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, *et al*., "Hive: a warehousing solution over a map-reduce framework," *Proc. VLDB Endow.*, vol. 2, pp. 1626-1629, 2009.