A Hadoop-based Approach for Efficient Web Service Management

Shangguang Wang*

National Engineering Laboratory for NGI Interconnection Devices
Beijing Jiaotong University
Haidian, Beijing, 10044, China
E-mail: sguang.wang@gmail.com
*Corresponding author

Wei Su

National Engineering Laboratory for NGI Interconnection Devices
Beijing Jiaotong University
Haidian, Beijing, 10044, China
E-mail: wsu@bjtu.edu.cn

Xilu Zhu

The Research Institution of China Mobile
China Mobile Limited
Changsha, Hunan, 425200 , China
E-mail: xilu.zhu@gmail.com

Hongke Zhang

National Engineering Laboratory for NGI Interconnection Devices
Beijing Jiaotong University
Haidian, Beijing, 10044, China
E-mail: hkzhang@bjtu.edu.cn

Abstract: In this paper, we propose a Hadoop-based approach for Web service management in Telecommunication and Internet domains. The basic idea of this approach is to adopt two components of Hadoop, HBase and MapReduce, to manage Web services. In HBase, we establish a HBase table based on request interface, and design a non-functional property index mechanism based on Qualify of Service (QoS) tree to discover Web services. In MapReduce, we employ it to search optimal Web services to satisfy users' complex Web service composition requirement. Experimental results show that our proposed approach can efficiently perform Web service discovery and composition for larger scale Web service management.

Keywords: Web service; Hadoop; HBase; MapReduce; Service management; Service composition.

Reference to this paper should be made as follows: Wang, S., Zhu, X. And Zhang, H. (2012) 'A Hadoop-based Approach for Efficient Web Service Management', Int. J. Web and Grid Services, Vol. , No. , pp.xx–xx.

Biographical notes: Shangguang Wang is a postdoctoral fellow at Beijing Jiaotong University. He received his Ph.D. degree in computer science and technology at Beijing University of Posts and Telecommunications in 2011. His research interests include service computing and cloud computing. He served as reviewer for Computer Journal; IET Software; Journal of Network and Computer Applications; International Journal of Web and Grid Services; International Journal of System Science; IEEE ICME 2011. He also served as session chair for IEEE GLOBECOM 2010 Workshop on Web and Pervasive Security, and ICICA 2010.

Wei Su is an associate professor at the National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University. His research interests include key theories and technologies for the next generation Internet. He currently is the group leader of the research project "Fundamental Research on Cognitive Services and Routing of Future Internet" which is supported by the National Natural Science Foundation of China.

Xilu Zhu is a senior researcher in The Research Institution of China Mobile. He received his Ph.D. degree in computer science and technology at Beijing University of Posts and Telecommunications in 2011. His research interests include service computing and cloud computing.

Hongke Zhang received his M.S. and Ph.D. degrees in Electrical and Communication Systems from University of Electronic Science and Technology of China in 1988 and 1992, respectively. From Sep. 1992 to June 1994, he was a postdoctoral fellow at Beijing Jiaotong University. In July 1994, he joined the School of Electronic and Information Engineering, Beijing Jiaotong University, where he is a professor. He has published more than 100 research papers in the areas of communications, computer networks and information theory. He is the holder of more than 50 Chinese patents and is the Chief Scientist of a National Basic Research Program of China (973 Program).

1 Introduction

With the emergence and widespread user of self-contained, self-describing modular application, Web services have been universally deployed and easily accessible on the Web (Guoquan et al., 2007). Web services have become a promising technology to design and build complex business applications out of atomic Web-based software components in service-oriented community (Wang, Sun and Yang, 2010). However, the huge amount of Web services and their ubiquitous usage to accomplish complex tasks may trigger the following two problems in Web service management (Zhu and Wang, 2011).

Firstly, as more and more customers delegate their task to Web service, one can expect a huge Web service repository to be fast and efficiently searched. However, because Web service advertisement information is a loose structure, it cannot be conveniently managed by the relational database. Moreover, some centralized models in traditional service management, such as UDDI platform, cannot easily scale to support a large number of Web services.

Secondly, although some artificial intelligence methods are widely used in Web service management, the performance still decrease as the large number of the same or different Web services involving during service discovery process. Moreover, in service management, some service composition algorithms based on the global optimization often merely consider the common QoS metrics, like availability, reliability, cost, response time, and reputation. They ignored other non-functional properties related to different business, like performance, compensation.

To improve the performance of Web service management, we propose a Hadoop-based approach for efficient management of large scale Web services, where Hadoop[1] can overcome the drawback which occurs in the traditional Web service management infrastructure. The two components of Hadoop, HBase and MapReduce, are integrated into our approach, where an HBase table and a non-functional property index mechanism are designed to strengthen the retrieving performance of the functional and non-functional properties of Web services. Moreover, a parallel Web service composition algorithm based on MapReduce is proposed to achieve the optimal service composition to satisfy users' requirement. Experimental results indicate that our approach significantly outperforms existing solutions in terms of efficiencies ensuring the function of service discovery and composition in service management infrastructure.

The rest of the paper is organized as follows. In the next section we discuss related work. We proposed our scheme for Web service management in Section 3. Experimental evaluations for comparing our solution against existing solutions are presented in Section 4. Finally, Section 5 gives conclusions and an outlook on possible continuations of our work.

2 Related Work

Web service management is a most important topic in the field of service-oriented computing, and lots of work has been done. The topic dealing with the large amount of Web services attracts many researchers' attention, especially, service discovery and composition.

Web service discovery is an important issue for Web service management. Web service discovery based on the input and output interfaces had been studied in (Di Martino, 2009, Di Martino and Moscato, 2008, Di Martino and Venticinque, 2008, Li and Horrocks, 2003, Srinivasan, Paolucci and Sycara, 2005). To enhance the efficiency of Web service discovery, Bellur and Kulkarni (2007) proposed a semantic search algorithm based on the concept of matching bipartite graphs in UDDI, for which proposal flags discrete degrees of matches using formal logic concepts of equivalence, subclass etc. and thus lends itself to an automated invocation of the discovered Web service. Zhang, Zheng and Lyu (2010) proposed a novel Web service search engine to expressively find expected Web services. The engine ranks the publicly available Web services not only by functional similarities to users' queries, but also by non-functional QoS characteristics of Web services. Di Martino (2009) presented an approach to semantic-based Web service discovery and a prototypical tool based on syntactic and structural schema matching. The approach is based on matching an input ontology, describing a service request, to Web services descriptions at the 'syntactic level' through Web Services Description Language or, at the semantic level, through service ontology described with languages. The

[1] http://hadoop.apache.org

different input schemas, WSDL descriptions, Ontology Web Language (OWL), OWL-S, WSMO, SWSF and WSDL-S components are represented in a uniform way by means of directed rooted graphs. Dong et al. (2009) proposed a semantic service retrieval engine which incorporates a novel semantic QoS evaluation methodology. The salient feature of this approach setting it apart from any other QoS evaluation methodologies is that the evaluation of the service and the subsequent quantification of QoS are a combination of subjective and objective QoS measures. Another salient feature of the approach is that it enables the domain-specific ranking of services.

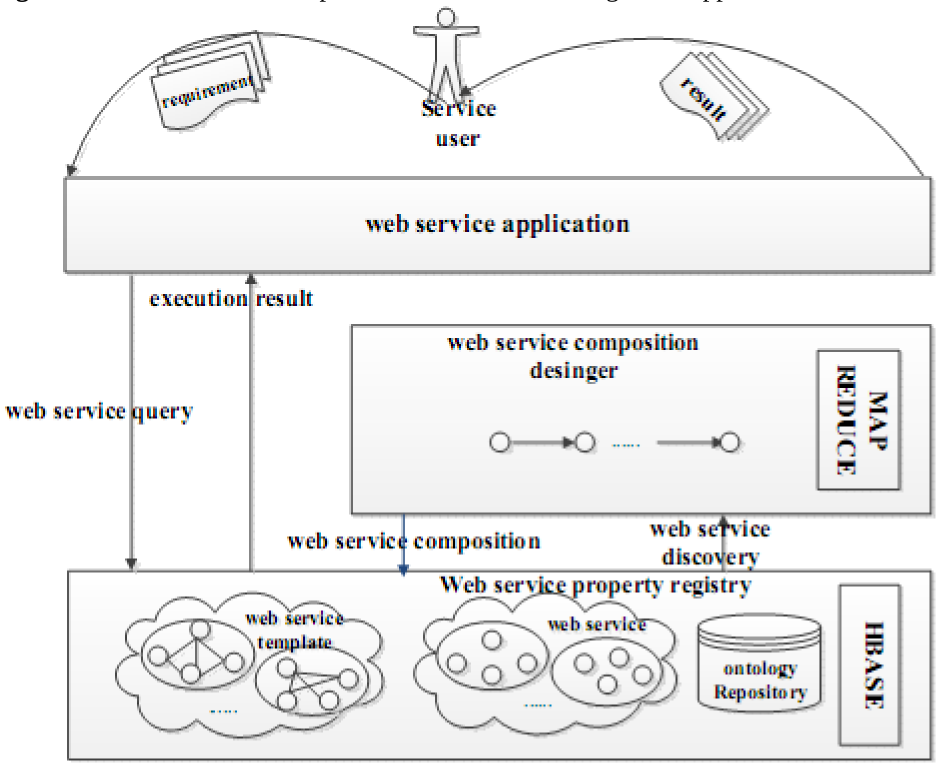
Constantinescu, Binder and Faltings (2005) proposed an extensible directory system for service discovery. The directory is organized as a special balanced search tree, where nodes also contain an “interesection” discriminator, in contrast to current systems which usually provide only a “union” discriminator. This kind of discriminator is used for early pruning in the case of negated queries and for providing tighter lower bounds in the case of numerical functions. For efficient search, the initial user query is automatically transformed into a query exploiting the internal directory structure (lower and upper bounds). A best first search technique is used for the lazy creation of the result set. Similar to this work, Skoutas et al. (2008) used the index-based mechanism for Web service discovery, and extends this to P2P network. The traditional service management techniques and frameworks are inadequate for handling new-generation integrated applications and cross-enterprise business processes and services (Kryvinska, Auer and Strauss, 2008). The Service Delivery Platform (SDP) is the proper, but still not obvious, choice for such management architecture. Hence, Kryvinska et al. (2008) identified both the current boundaries of SDP and the concept's importance. They classify and analyse the components and the relations between the elements of this management framework, and visualise the results of the analysis through the reference architecture. Based on the representation of the service management framework, they provided a rational view of the whole service delivery environment. Moreover, to overcome the problem resulted from huge number, a distributed model is applied into this area. For example, Tang, Xu and Dwarkadas (2003) distributed Web service advertisement in CAN network based on the semantics generated by Latent Semantic Indexing. Banaei-Kashani, Chen and Shahabi (2004), and Emekci et al. (2004) proposed a distributed framework for service discovery. (Gibelin and Makpangou, 2005) proposed a distributed QoS registry architecture in the non-functional property management. Li et al. (2007) proposed a Napster-like P2P system to support distributed QoS register.

Composing simple Web services into a complex one also cannot be ignored in Web service management research. Based on the tree or graph structure, some outstanding algorithms (Haiming, Farong and Chang, 2008, Shiaa, Fladmark and Thiell, 2008, Zhou, Huang and Wang, 2007) are proposed to improve the composition efficiency. They focus on avoiding the unrestricted growth of the tree depth. Pathak et al.(2006) adopted parallel strategy for Web service composition in the case of parallel composition. Hennig and Balke (2010) focused on the question of how to harness the power of multi-core architectures to build composition frameworks providing the actual scalability needed for life science applications. Using the running example of finding workflows in metabolic networks for the area of systems biology, the authors proposed two novel parallelization techniques. The optimizations are specifically tailored to the needs of the service composition problem, and their basic paradigms of course can also be exploited in general application scenarios. Blanco et al. (2012) proposed sampling-based techniques to accurately estimate QoS values that will be used in a hybrid composer named PT-SAM, to identify the compositions that best meet a user request. PT-SAM adapts a Petri-Net unfolding and uses a utility function defined on the QoS estimates, to guide the composer. By looking into the coordination of Web services following their acceptance to participate in a composition scenario, Yahyaoui et al. (2010) identified two types of behaviours associated with component Web services: Operational and control behaviours. These behaviours are used to specify composite Web services that are built upon component Web services. In term of orchestration a composite Web service could be either centralized or P2P. Moreover, to support composite Web services coordination per type of orchestration schema, various types of messages are exchanged between these Web services. Eid et al. (2008) given a survey of a representative set of these systems to capture the requirements and challenges of these composition systems. Moreover, they developed a reference model for describing the functional structure and evaluating the performance of dynamic Web service composition systems based on existing dynamic Web service composition platforms and prototypes. There are also some other service composition schemes (Cardinale, 2011, Eid, Alamri and El Saddik, 2008, Ukey et al., 2010) and this section will not introduce in detail.

Unlike traditional Web service management technology, our Hadhoop-based approach can seamlessly integrate with discovery and composition mechanism, and is more suitable to manage the loose-structure information of Web services, especially, lager-scale Web service data.

3 Hadoop-based Web Service Management

Figure 1 Procedures of Hadoop-based Web service management approach



As shown in Fig. 1, our proposed Hadoop-based approach mainly contains two components, i.e., HBase and MapReduce.

HBase stores the functional and non-functional properties of Web services, and maintains Web service advertisement and the ontology tree, which contributes to construct the function property table. Then we design a HBase table and propose a non-functional property index mechanism. Because HBase is suitable to loose-structure data (each data row has a sortable key and each data column is varying), it can effectively manage Web services with different properties.

MapReduce can participate into retrieving the non-functional and functional interfaces which satisfy users' requirements from HBase. MapReduce can also select the best matched Web services among them, and concurrently filter the unqualified-QoS Web services. Based on MapReduce, we propose a parallel Web service composition algorithm, which can find an optimal solution for users' complex requirement within Web service composition.

3.1 *HBase for Web Service Storage*

HBase is a robust, scalable and distributed store being capable of hosting billions of Web services' advertisement information. It plays an important role in Web service discovery and composition. Hence, an efficient index strategy is significantly important for processing large amount of Web services. In this section, we propose an integrating index mechanism with HBase to retrieve properties by storing non-functional property and functional property of Web service.

3.1.1 Store non-functional property of Web services

QoS requirement is a necessary part in Web service discovery and composition. QoS is widely employed for describing non-functional characteristics of Web services (Zheng, Zhang and Lyu, 2010). However, there is no mechanism for HBase to support a complex query (e.g., a service user requires the metrics of reliability, availability and reputation is no less than 50.) in non-functional property index. In addition, because Web services in different domain may face different QoS requirements, local QoS optimum cannot be ignored. Hence, to strengthen HBase, we introduce QoS tree index.

Before we construct QoS tree, for easy illustration, we intend to quantify the QoS metrics and map them into normalized space. Actually, there are three types of relationships among quantified QoS, i.e., strong dominance, dominance and ε-weak dominance.

Definition 1(Strong Dominance): $QoS_1 = (q_1, q_2 \cdots q_m)$ strongly dominate $QoS_2 = (q_1^{'}, q_2^{'} \cdots q_m^{'})$, denoted as $QoS_1 \succ QoS_2$, if $\forall i \in \{1, \cdots, m\} q_i > q_i^{'}$.

Definition 2(Dominance): $QoS_1 = (q_1, q_2 \cdots q_m)$ dominate $QoS_2 = (q_1^{'}, q_2^{'} \cdots q_m^{'})$, denoted as $QoS_1 \succeq QoS_2$, if $\exists i \in \{1, \cdots, m\} q_i = q_i^{'}$.

Definition 3(ε -weak dominance): $QoS_1 = (q_1, q_2 \cdots q_m)$ ε-weakly dominate $QoS_2 = (q_1^{'}, q_2^{'} \cdots q_m^{'})$, denoted as $QoS_1 \succ QoS_2$, If $\exists i \in \{1, \cdots, m\} q_i \geq q_i^{'}$, where the number of $q_i \geq q_i^{'}$ is ε ($\varepsilon < m$).

According to these definitions, Fig. 2 describes the procedure of QoS index tree. In the $node_{QoS}$ structure, the $node_{QoS^{'}}$ managed by *LeftChild* is strongly dominated or dominated by the $node_{QoS}$, and the $node_{QoS^{'}}$ managed by *RightChild* is ε-weakly dominated by the $node_{QoS}$. Hence, the relation between parent and its child depends on the node is left child or right child. QoS denotes the non-functional vector ($q_1, q_2 \cdots q_n$). Val is the average value of QoS vector, $\frac{1}{n}\sum_{i=1}^{n} q_i$.

As shown in Fig. 2, according to definition of the QoS node, the left and right child can be considered as index node to reduce the searching cost due to the transitive relation. To simplify the index, we construct a virtual index, which is diagonal of the hypercube of the quantified QoS dimension, as the main index. That is, $(0,\cdots,0) \prec (1,\cdots,1) \prec\cdots\prec (100,\cdots,100)$ is introduced. The detailed algorithm of joining and locating the qualified QoS is in (Zhu and Wang, 2010). Hence, as shown in Table 1 we design an HBase table in terms of the QoS tree.

Figure 2 Procedures of QoS index tree

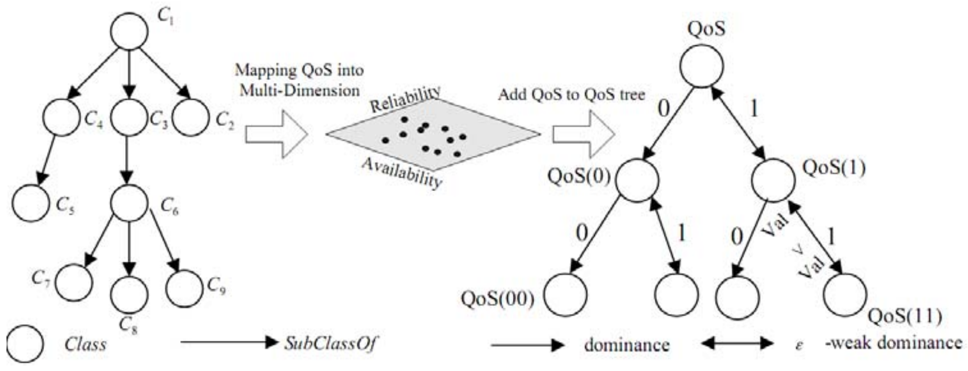


Table 1 QoS Index Table

Row Key	Time Stamp	Relationship		Web Service	
travel90151	T1	Qos	QoS metrics	Serv1	DomName
		Parent	QoSId	Serv2	DomName
		Lchild	QoSId	Serv3	DomName
		Rchild	QoSId	Serv4	DomName
		Parent inbst	QoSId	Serv5	DomName

In Table 1, the row key in HBase is the combination of domain and the encoded QoSId, such as (travel90151), where travel represents the Web services' domain and 90151 contains the main index value and the sequence number in this main index. The column family relationship describes the QoSId's relationship in QoS tree. In addition, the leftmost node and the rightmost node can be constructed as a binary search tree to reduce the time cost. Thus, the relationship column for the index nodes stores immediate family member in the binary search tree. The column family Web service includes urls of Web service where their QoS is equal to QoS metrics in relationship column. By using the row key, we can concurrently retrieve the qualified QoS in different domain to achieve performance improvement.

3.1.2 Store functional property of Web services

Besides the non-functional property management, the functional property can also be stored in the HBase. For Web service composition, the common effective method relies on matching interface based on semantic.

More specifically, there are four different scenarios in the interface matching process, which need to be considered (we assume that I denotes request interface and O denotes the response interface): 1) the concepts of O and I are the same, Sim(I=O); 2) I subsumes O, Sim(I<O); 3) O subsumes I, Sim(I>O); 4) O is not directly related to I, Sim($I \neq O$).. The degree of the similarity Sim(I,O) decreases from the scenario 1) to 4), thus, we can use a generic criterion for accessing the similarity degree between interface I and interface O by the following:

$$Similarity(I,O) = \frac{\{C \mid C \in C_I \wedge C \mid C \in C_O\}}{\max(\{C \mid C \in C_I, C \mid C \in C_O\})} . \tag{1}$$

In fact, ontology is a modeling tool for concept C, and ontology can be also represented graphically by a tree structure. Thus, the ontology table structure is similar to Table 2, and the column family relationship includes the concept's immediate family in ontology tree. In addition, it includes the other two types of column family, the request interface and the response interface, where they store the interface instance belonging to this concept. The basic method for finding matched Web service relies on the interface. Hence, we create an interface index based on request interface for all Web services, which is convenient for searching the matched interface among Web services and can reduce the time cost of redundant interface comparison. As shown in Table 3, we design an interface index table.

Table 2 Ontology Table

Row Key	Time Stamp	Request Interface		Response Interface		Relationship	
conp	T1	Req1	Inst1	Res1	Inst1	Parent	Conp1
		Req2	Inst2	Res2	Inst2	Child	Conp2
		Req3	Inst3	Res3	Inst3	Brother	Conp3

Table 3 Interface index table

Row Key	Time Stamp	Web Service	Matched Interface	
conpInst	T1	DomName	Instance	Inst1
			Similarity	0.5
			Service Name	DomName
			Other Interface	conpInst

In Table 3, the row key is constructed by the combination of concept and interface id, which helps to concurrently retrieve interfaces belonging to the same concept, and column family Web service is used to filter QoS-unqualified Web service by matching column family Web services in QoS table. The matched response interface instance and Web service name that are stored in the *matched Response* column family can be obtained from column family *response interface* in ontology table. The last column family *Request Inteface* stores the request that belongs to the Web service producing the response interface in the *matched Response*. The *Request Inteface* can be obtained from the table, which mainly manages the interface information. After fetching all matched interfaces, the next step is to find the matching relation among Web services by using Web service match algorithm (called SBMWS) as shown in Algorithm 1.

Algorithm 1 *SBMWS*
Input: *Interface requirement*
Output: *best matched Web service*

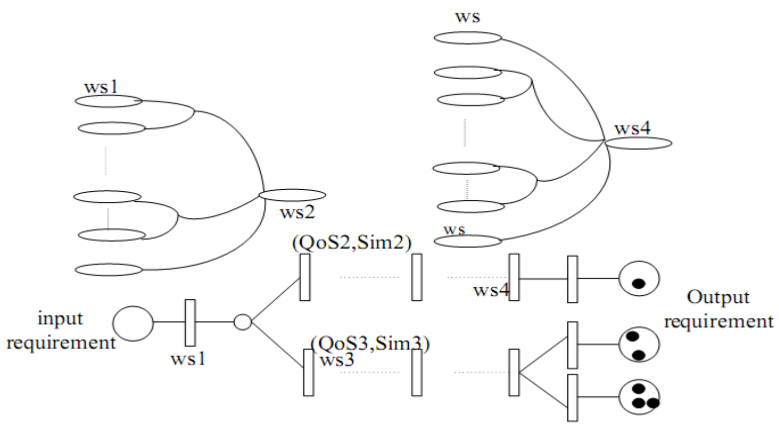
```
1.  Function Mapper(key,value)
2.      Ouput(wsn, matched Interface);
3.  EndMapper;
4.  Function Reducer(wsn, matched Interfaces)
5.    For(intfs: matched Interfaces));
6.      For(rintfs:require Interfaces)
7.          getBestMatchedCombination( );
8.    Output(Null, matchedResult);
9.  EndReducer
```

According to Algorithm 1, the worst case of the time complexity is about O (n^2). The function Mapper () is used to obtain the matched interface among Web services, and then the algorithm can find the best matched among services since it's possible that the response interfaces of a single Web service cannot satisfy all request interfaces. Hence, the parallel process is more contributable to improve the efficiency since the time complexity of the serial process must multiply the number of Web service O (n^2*WS).

3.2 MapReduce for Web Service Composition

By using HBase and MapReduce, we can efficiently and reliably retrieve the composition related information from the Web service repository. To accomplish the complex task, each simple Web service must be composed together with QoS reaching the global optimization. Figure 3 presents the procedure of Web service composition.

Figure 3 Procedures of Web Service Composition



We use the Petri net to describe the Web service composition graph, and the token in this graph denotes the customer's response requirement. The response place can be considered as the start point. Any Web service which produces one or more response requirement can be added into the search path. Hence, the end point stands for a Web service which satisfies the request requirement. The weight of the edges among Web services can be measured by two metrics, Web service similarity and QoS. Then an evaluation method of Web service composition (called WSC) is given as

$$score(cws) = (QoS_{all}, Sim_{all}) = \omega_1 QoS_{all} + \omega_2 Sim_{all} \tag{2}$$

Where QoS_{all} can be defined as $\sum_{i=1}^{n} QoS(ws_i)$, $\min(QoS_i)$ in consequence and concurrent models, the Sim_{all} can be represented by $\frac{1}{n}\sum_{i=1}^{n-1} Sim(ws_i, ws_{i+1})$.

According to (2), we propose a parallel Web service composition that contains Single-source Shortest Path (SSSP) algorithm and Multi-Source Shortest Path (MSSP) algorithm. SSSP is to find the optimal solution where the time complexity is O (n+v). MSSP is to reduce the time cost.

In SSSP and MSSP algorithms, we adopt two different type of parallel strategy. SSSP adopts the BSF method, where the Mapper function is to compute weight sum to the incoming vertex v_i, and Reducer function is to select the maximum weight. Time complexity of SSSP is the diameter of the network, i.e., O (n). However, because the mechanism of MapReduce results in more IO cost, and experiments show that its performance is far worse than the serial algorithm, we consider this inverted search process as the multi-source search (since any Web service or their combination producing all the response requirement can be considered as a source). Hence, we can distribute them to different computer and find the shortest path simultaneously, and then its time complexity reduces to O ((n+v)/P).

Algorithm 2 SSSP

```
Function Mapper($v_1$, $val_1$)
  $\forall v_i, v_i \in (v_1, v_i)$
  If (dist($v_1$)+dist($v_1$,$v_i$)¡dist($v_i$))
    setDist($v_i$);
    addPath($k_1 + k_i$);
  End If
  Output($v_i$,$v_i$.info)
End Mapper
Function Reducer($v_2$,$vals$)
  While($v_i \in vals$)
      If ($v_i.dist$ ¡$v_2.dist$)
      $v_2.dist$=$v_i.dist$;
      $v_2.ssp$=$v_i.ssp$;
  End If
  Output($v_2$, $v_2.info$)
End Reducer
```

Algorithm 3 MSSP

```
Function Mapper($v_1$, $val_1$)
/*v1 is start point*/
  $\forall v_i, v_i \in (v_1, v_i)$
  Output($v_i$,$v_i$.info)
  End Mapper
Function Reducer(k2,vs)
  while($v_i \in v_all$)
    $\forall v_j, v_j \in (v_i, v_j)$
    If ($v_j$.dist ¡dist($v_i$,$v_j$))
      setDist($v_j$);
      setSSP($v_j$);
    End If
  $v_{all}$=$v_{all}$-$v_i$
  Output(Null,MSSP)
End Reducer
```

4 Performance Evaluation

In this section, we investigate the performance of our approach by comparing it with existing approaches. The performance metrics include the query cost, IO (discovery or match) cost and Web service composition cost.

4.1 Experiment Setup

In order to evaluate our Hadoop-based approach, we have implemented it based on the Hadoop platform[1]. For these experiments, we use a distributed cluster with CPU 2GHz, RAM 8GB and Linux operating system.

All algorithms have been implemented in Java version 1.6. We randomly generated composition requests and reported the average time of several runs by using the dataset from generator of WS-Challenge 2008[2] and WSBen[3].

4.2 Query Cost

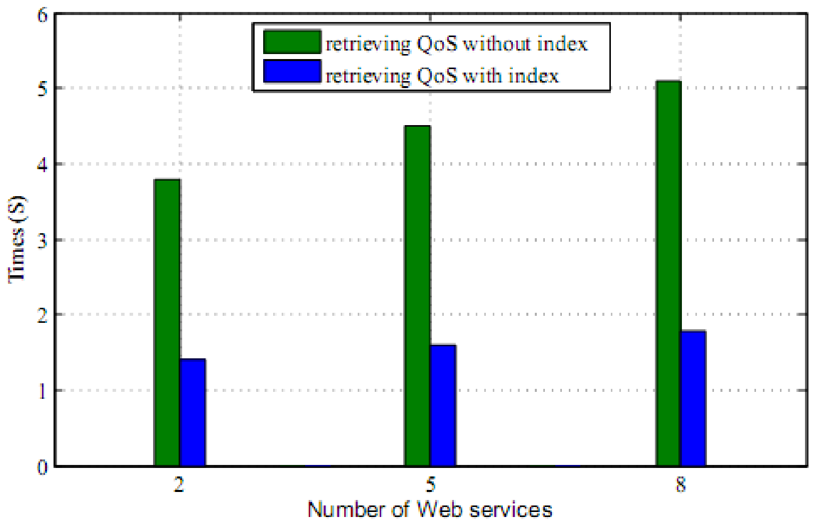
Fig. 4 shows the query cost when we retrieve the functional and non-functional properties from HBase with or without using index mechanism.

From Fig. 4(a), the QoS query cost maintains a low level, since the strong-dominance or dominance relation in the QoS tree contributes to reduce the unnecessary comparison. In addition, the time consumption doesn't grow as the QoS dimension increases, since the comparison times in QoS tree doesn't grow as the QoS dimension increase. On the contrary, the time cost increases as the dimension increases when retrieving the QoS without using index mechanism. Hence, the QoS index tree contributes to reduce time consumption when retrieving QoS from HBase. Fig. 4(b) shows the time consumption of

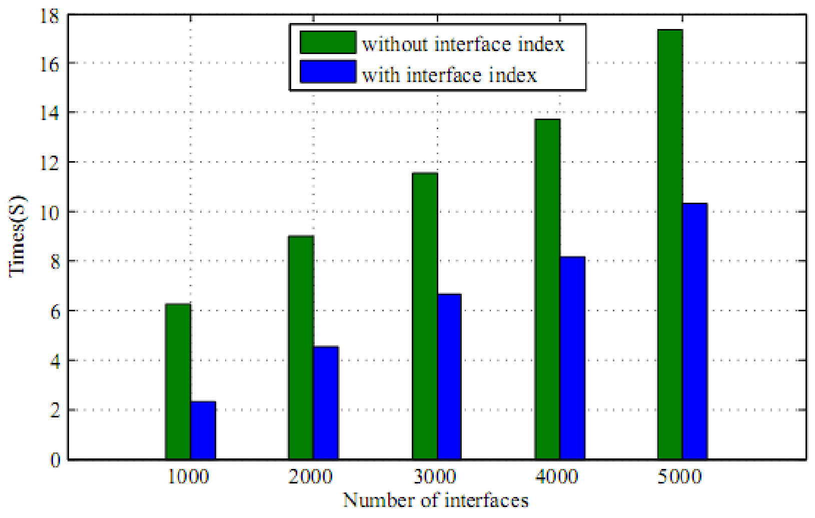
[1] http://hadoop.apache.org
[2] http://cec2008.cs.georgetown.edu/wsc08
[3] http://www2.ie.psu.edu/Kumara/Research/lisq/indexfiles/wsben/WSBen.htm

retrieving the matched interface from HBase and the WSDL files generated by WSBen. The latter will pay more IO cost of reading files and repeatedly searching files to find matched interfaces, while the interface-based index merely produce more storage cost. The experimental results demonstrate that interface-based index is less query cost than that of merely rely on WSDL files.

Figure 4 Query cost



(a) Query cost w.r.t the number of Web services



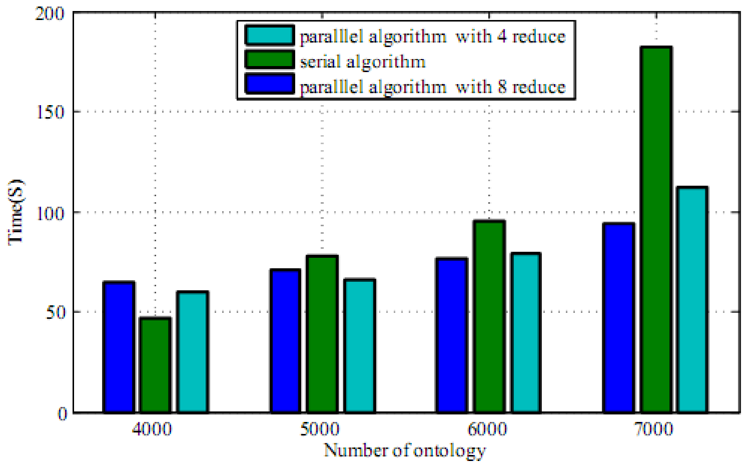
(b) Query cost w.r.t the number of interfaces

4.3 IO Cost

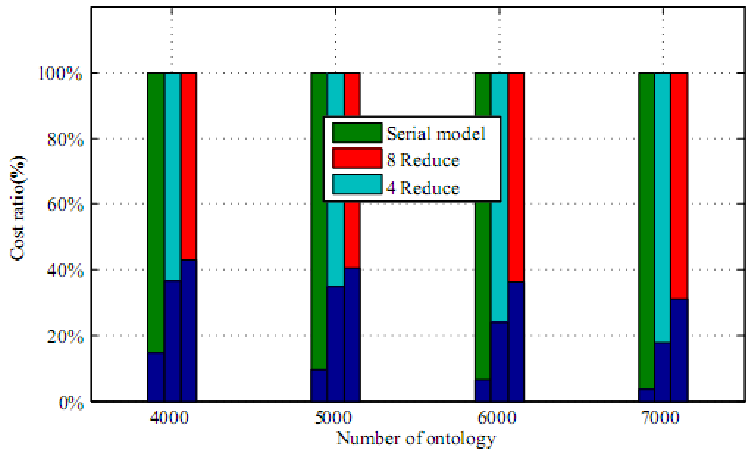
In this experiment, we generate a test set which includes 2000 Web services with different ontology number by using generator in WS-challenge 2008.

As shown in Fig. 5, we investigate the improvement of our approach computation time to find the best matched among the large-scale Web services. In Fig. 5(a), with the increasing of the ontology number, the time consumption grows slowly in parallel algorithm. On the contrary, the serial algorithm reaches to 180 when the Web services with 7000 ontology. Moreover, Fig. 5(b) reflects that the IO cost during the whole process. The additional IO cost results from the higher communication and data movement in cluster, which makes the parallel algorithm take more consumption, especially, when this is with more Reduce number involving the computing. However, although it brings more extra IO cost, the performance can be further imported as the more computers involves in this process. Hence, our approach is still efficient.

Figure 5 IO cost



(a) IO cost w.r.t the number of ontology



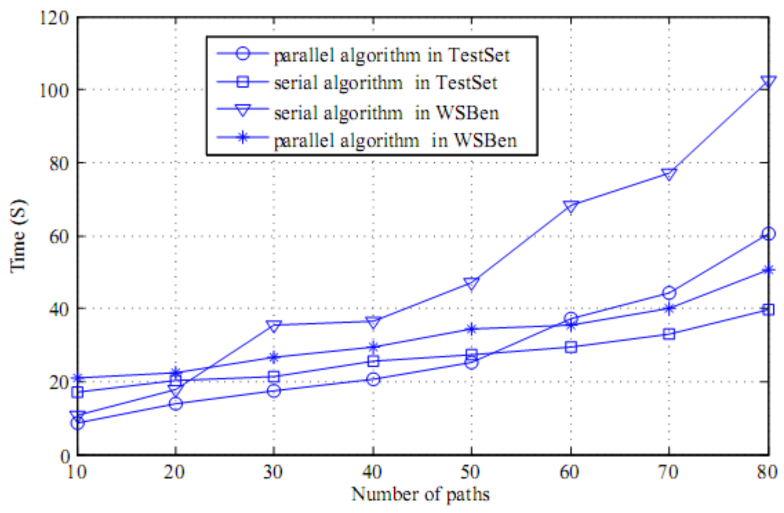
(b) IO cost ratio w.r.t the number of ontology

4.4 Web service composition cost

In this section, we investigate Web service composition cost in two types of datasets (WS-Challenge 2008 and WSBen) to find the optimal solution with different number of paths (Web service composition paths). In this experiment, we add more response interfaces to get more output combination.

As shown in Fig. 6, the serial algorithm still outperforms the parallel algorithm, when applying the algorithm to a small number of paths. Although the parallel algorithm distributes the searching process to each computer, the cost of data movement and communication overpass the computing consumption because of the Mapreduce mechanism. As the number of paths increase, although the iterative search the best solution process lower the performance of serial process, our proposed parallel algorithm still apparently reduce the time cost of Web service composition.

Figure 6 Web service composition cost



5 Conclusion

Due to the huge amount of Web services, existing approaches cannot manage the huge amount of Web services in distributed Web environment efficiently. In this paper, we propose a Hadoop-based approach for Web service management. In our approach, we design an HBase table and propose a non-functional property index mechanism to strengthen the retrieving performance of the functional and non-functional properties of Web services. Moreover, based on MapReduce we propose a parallel algorithm to achieve the optimal service composition for satisfying users' requirement. Experimental results show that our approach has a good performance in managing large-scale Web services.

Our future work deals with the dependability (Huang et al., 2011) of the service system work when deploying our scheme in “BigCloud” system[1] (a cloud computing platform) .

Acknowledgements

The work presented in this study is supported by NSFC (61202435), NSFC (61272521), SRFDP (20110005130001), CPSF (2011M500226) and NSFC (100263).

[1] http://221.183.16.16:8080

References

Banaei-Kashani, F., Chen, C.-C. & Shahabi, C. (2004) 'WSPDS: Web Services peer-to-peer Discovery Service'. Proceedings of Proceedings of the International Conference on Internet Computing, IC'04, June 21, 2004 - June 24, 2004, 2004 Las Vegas, NV, United states.pp. 733-39.

Bellur, U. & Kulkarni, R. (2007) 'Improved matchmaking algorithm for semantic web services based on bipartite graph matching'. Proceedings of 2007 IEEE International Conference on Web Services, ICWS 2007, July 9, 2007 - July 13, 2007, 2007 Salt Lake City, UT, United states.pp. 86-93.

Blanco, E., Cardinale, Y. & Vidal, M. E. (2012) 'Experiences of sampling-based approaches for estimating QoS parameters in the Web Service composition problem', International Journal of Web and Grid Services, Vol.8, No.1, pp. 1-30.

Cardinale, Y. (2011) 'CPN-TWS: a coloured petri-net approach for transactional-QoS driven Web Service composition', International Journal of Web and Grid Services, Vol.7, No.1, pp. 91-115.

Constantinescu, I., Binder, W. & Faltings, B. (2005) 'Flexible and efficient matchmaking and ranking in service directories'. Proceedings of 2005 IEEE International Conference on Web Services, ICWS 2005, July 11, 2005 - July 15, 2005, 2005 Orlando, FL, United states.pp. 5-12.

Di Martino, B. (2009) 'Semantic web services discovery based on structural ontology matching', International Journal of Web and Grid Services, Vol.5, No.1, pp. 46-65.

Di Martino, B. & Moscato, F. (2008) 'Foreword: Semantic web and semantic information management', International Journal of Web and Grid Services, Vol.4, No.3, pp. 247-50.

Di Martino, B. & Venticinque, S. (2008) 'Advance in web/grid information and services discovery and management', International Journal of Web and Grid Services, Vol.4, No.4, pp. 353-56.

Dong, H., Hussain, F. K. & Chang, E. (2009) 'A QoS-based service retrieval methodology for digital ecosystems', International Journal of Web and Grid Services, Vol.5, No.3, pp. 261-83.

Eid, M., Alamri, A. & El Saddik, A. (2008) 'A reference model for dynamic web service composition systems', International Journal of Web and Grid Services, Vol.4, No.2, pp. 149-68.

Emekci, F., Sahin, O. D., Agrawal, D. & El Abbadi, A. (2004) 'A peer-to-peer framework for Web service discovery with ranking'. Proceedings of Proceedings - IEEE International Conference on Web Services, ICWS 2004, June 6, 2004 - June 9, 2004, 2004 San Diego, CA, United states.pp. 192-99.

Gibelin, N. & Makpangou, M. (2005) 'Efficient and transparent web-services selection'. Proceedings of the 3rd International Conference on Service-Oriented Computing, ICSOC 2005, December 12, 2005 - December 15, 2005, 2005 Amsterdam, Netherlands.pp. 527-32.

Guoquan, W., Jun, W., Xiaoqiang, Q. & Lei, L. (2007) 'A Bayesian network based Qos assessment model for web services'. Proceedings of 2007 IEEE International Conference on Services Computing, SCC 2007, July 9, 2007 - July 13, 2007, 2007 Salt Lake City, UT, United states.pp. 498-505.

Haiming, T., Farong, Z. & Chang, Y. (2008) 'A tree-based method of Web service composition'. Proceedings of the 8th IEEE International Conference on Web Services, ICWS 2008, September 23, 2008 - September 26, 2008, 2008 Beijing, China.pp. 768-70.

Hennig, P. & Balke, W.-T. (2010) 'Highly scalable web service composition using binary tree-based parallelization'. Proceedings of the 8th IEEE International Conference on Web Services, ICWS 2010, July 5, 2010 - July 10, 2010, 2010 Miami, FL, United states.pp. 123-30.

Huang, J., Lin, C., Kong, X. & Zhu, Y. (2011) 'Modeling and analysis of dependability attributes of service computing systems'. Proceedings of 2011 IEEE International Conference on Services Computing, SCC 2011, July 4, 2011 - July 9, 2011, 2011 Washington, DC, United states.pp. 184-91.

Kryvinska, N., Auer, L. & Strauss, C. (2008) 'Managing an increased service heterogeneity in a converged enterprise infrastructure with SOA', International Journal of Web and Grid Services, Vol.4, No.4, pp. 440-60.

Li, F., Yang, F., Shuang, K. & Su, S. (2007) 'Q-peer: A decentralized QoS registry architecture for Web services'. Proceedings of the 5th International Conference on Service-Oriented Computing, ICSOC 2007, September 17, 2007 - September 20, 2007, 2007 Vienna, Austria.pp. 145-56.

Li, L. & Horrocks, I. (2003) 'A Software Framework for Matchmaking based on Semantic Web Technology'. Proceedings of 12th International World Wide Web Conference 20-24 May 2003 Budapest, Hungary.pp. 331-39.

Pathak, J., Basu, S., Lutz, R. & Honavar, V. (2006) 'Parallel Web service composition in MoSCoE: A choreography-based approach'. Proceedings of the 4th European Conference on Web Services, ECOWS 2006, December 4, 2006 - December 6, 2006, 2006 Zurich, Switzerland.pp. 3-12.

Shiaa, M. M., Fladmark, J. O. & Thiell, B. (2008) 'An incremental graph-based approach to automatic service composition'. Proceedings of 2008 IEEE International Conference on Services Computing, SCC 2008, July 7, 2008 - July 11, 2008, 2008 Honolulu, HI, United states.pp. 397-404.

Skoutas, D., Sacharidis, D., Kantere, V. & Sellis, T. (2008) 'Efficient Semantic Web service discovery in centralized and P2P environments'. Proceedings of the 7th International Semantic Web Conference, ISWC 2008, October 26, 2008 - October 30, 2008, 2008 Karlsruhe, Germany.pp. 583-98.

Srinivasan, N., Paolucci, M. & Sycara, K. (2005) 'An efficient algorithm for OWL-S based semantic search in UDDI'. Proceedings of First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, July 6, 2004 - July 6, 2004, 2005 San Diego, CA, United states.pp. 96-100.

Tang, C., Xu, Z. & Dwarkadas, S. (2003) 'Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks'. Proceedings of ACM Conference on Computer Communications, SIGCOMM 2003, August 25, 2003 - August 29, 2003, 2003 Karlsruhe, Germany.pp. 175-86.

Ukey, N., Niyogi, R., Singh, K., Milani, A. & Poggioni, V. (2010) 'A Bidirectional Heuristic Search for web service composition with costs', International Journal of Web and Grid Services, Vol.6, No.2, pp. 160-75.

Wang, S. G., Sun, Q. B. & Yang, F. C. (2010) 'Towards Web Service selection based on QoS estimation', International Journal of Web and Grid Services, Vol.6, No.4, pp. 424-43.

Yahyaoui, H., Maamar, Z. & Boukadi, K. (2010) 'A framework to coordinate web services in composition scenarios', International Journal of Web and Grid Services, Vol.6, No.2, pp. 95-123.

Zhang, Y., Zheng, Z. & Lyu, M. R. (2010) 'WSExpress: A QoS-aware search engine for Web services'. Proceedings of 2010 IEEE 8th International Conference on Web Services, ICWS 2010, July 5, 2010 - July 10, 2010, 2010 Miami, FL, United states.pp. 91-98.

Zheng, Z., Zhang, Y. & Lyu, M. R. (2010) 'Distributed QoS evaluation for real-world Web services'. Proceedings of 2010 IEEE 8th International Conference on Web Services, ICWS 2010, July 5, 2010 - July 10, 2010, 2010 Miami, FL, United states.pp. 83-90

Zhou, A., Huang, S. & Wang, X. (2007) 'BITS: A binary tree based web service composition system', International Journal of Web Services Research, Vol.4, No.1, pp. 40-58.

Zhu, X. & Wang, B. (2010) 'A distributed quality of service index framework'. Proceedings of 2010 IEEE Asia-Pacific Services Computing Conference, APSCC 2010, December 6, 2010 - December 10, 2010, 2010 Hangzhou, China.pp. 123-30.

Zhu, X. L. & Wang, B. (2011) 'Web service management based on Hadoop'. Proceedings of 8th International Conference on Service Systems and Service Management, ICSSSM'11, June 25, 2011 - June 27, 2011, 2011 Tianjin, China.pp.1-6.