International Journal of Database Theory and Application
Vol.9, No.10 (2016), pp.327-336
http://dx.doi.org/10.14257/ijdta.2016.9.10.28

Research on Service Component Retrieval with Hadoop

Wen Si[1], Rong Tan[1] and Lu Bing[1]

[1]*College of Information and Computer Science, Shanghai Business School, 201400, Shanghai, P. R. China*
siw@sbs.edu.cn, tanrong529@gmail.com, betty20006@163.com

Abstract

It remains both important and difficult for service-oriented computing to retrieve accurately and efficiently the service components fulfilling the client's need. This paper intends to present a method for use in retrieving the personalized service components based on Hadoop architecture. It extends the semantic ontology of service components, and proposes the methods for building and inquiring the index files based on Lucene. By utilizing the HDFS and adopting the QoS computation and personalized recommendation techniques, the working principles of Map and Reduce of Hadoop are expanded. Experiment results show that the proposed method is applicable in realizing the retrieval of service components in distributed computing environment.

Keywords*: Service Component, Hadoop, Lucene, QoS computing, Personalized Recommendation*

1. Introduction

Service-oriented computing (SOC) and service-oriented architecture (SOA) which utilize Web services to dynamically construct loosely coupled service applications, greatly promote the development of e-commerce, finance, telecommunications and other fields [1]. In order to meet the complex business needs, a number of services are required to be combined as a whole.

However, it is difficult to quickly retrieve the appropriate services by traditional methods such as the keyword query in the database, the ontology [2-3] query and the agent [4] query. While keyword query is a simple literal matching algorithm and can be integrated conveniently, the service retrieval requests are too restrictive, and the retrieval time is generally high resulting in low efficiency. Querying by agents can search a large number of distributed service components, the precision rate is still not very ideal since the semantic service cannot be described by the agents. Although using ontologies are capable of retrieving the semantically similar service components, it costs much time to construct and traverse the ontology files with traditional methods. With the more the files to be indexed, the longer the time costs which is a geometric growth.

This paper proposes a personalized service components retrieval method based on Hadoop [5-6]. The ontologies are used to describe the semantics of the service components including interface parameters, return values, service component domain ontology, and service performance ontology. By using the means of reversed indexing of Lucene [7], the service component ontologies are made. Furthermore, based on the Hadoop distributed file system, the index files are stored. When the users submit the personalized service requests, the task will be divided into hundreds of sub tasks parallel executed by MapReduce of Hadoop. The experiments show that it can improve the efficiency of retrieving target service components.

ISSN: 2005-4270 IJDTA

2. Index File Construction for Service Component

2.1. Semantic Description of Service Component

Five views of upper ontology are given to describe the semantics of service components, as shown in Figure1, that are QualityModel, InteractionModel, BaseInfor, CategoryModel, BusinessModel.

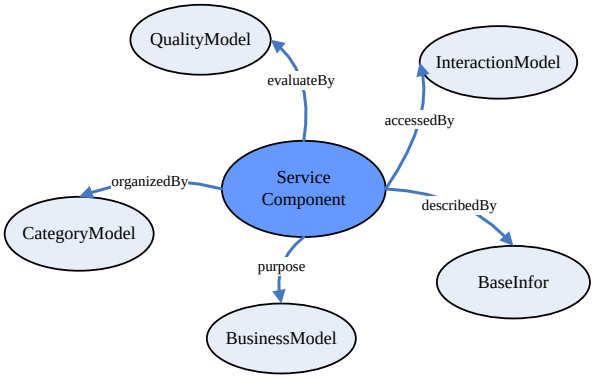


Figure 1. The Upper Level Ontology View of Service Component

BusinessModel describes the usage of service components, and is defined by a three-variable combination (S, Func, Owner), where S denotes the current service component, Func denotes the functional description of service, Owner denotes the owner of current service component. QualityModel describes the quality information of service component, and is defined by an eight-variable combination (S, Time, Cost, Reliability, Availability, Reputation, Compensation, Penalty), where Time denotes the response time; Cost denotes the execution cost for using the current component; Reliability denotes the possibility that the component is called by a procedure in the right way; Availability denotes the possibility that a service can be accessed; Reputation denotes the evaluation of the current service component given by past clients; Compensation denote the compensation action offered by the service component once it cannot fulfill the client's expectation; Penalty denotes the penalty that must be paid if the use of current service component is declined for some reason.

CategoryModel supplies a way to service components organization, which becomes the prerequisite for the management of service component library. CategoryModel is defined by (S, DR, S'), where S denotes the service component; DR denotes relations of ontology; S' denotes the service component that has a child-father relation with S.

BaseInfor describes the functionality specifications of the service component, including the input and output, constraints, dependency, *etc.* BaseInfor is defined by referring to OWL-S semantic service description as (S, Resource, Service Profile, Service Model, Service Grounding), where Resource binds up itself with Service via attribute Provides, and describes semantics of service within the ontology, *e.g.* the category, the hierarchical dependency, *etc*. Service Profile binds up itself with Service via attribute Presents, and describes the fundamental information of service, *e.g.* the organization that supplies the service, the service functionality (including input and output information, parameters, prerequisites and post-conditions), the attribute description of service characteristics (including service category, extra-parameters, Qos, service address and extensible listed information, *etc.*). Service Model binds up itself with Service via attribute describe By, and describes the service details and working procedures. Service Grounding binds up itself with Service via attribute supports, and defines the manner that client gets access to the current service.

InteractionModel describes how the services supplied by components can be obtained, and the model of relations with components. InteractionModel is defined as (S, Invoke, Port, URL), where S denotes the current service component, Invoke belongs to the set {One-Way, Notification, Request-Response, Solicit-Response}, and denotes the interactive mode of service callback; Port belongs to the set {in, out, in-out, out, in}, and denotes the interface type; URL denotes the location of service.

2.2. Index Building Process for Service Component

Figure2 illustrates the building process of service components index. Firstly, Web service and its semantic description ontology deployed in each server are interpreted and then extracted by Xpath to obtain the information of semantic attribute of service components. Secondly, the semantic attribute is written into Field and Document by InderWriter (based on Lucene), respectively. Finally, the retrieved files are stored in HDFS and the results are returned to servers, respectively.

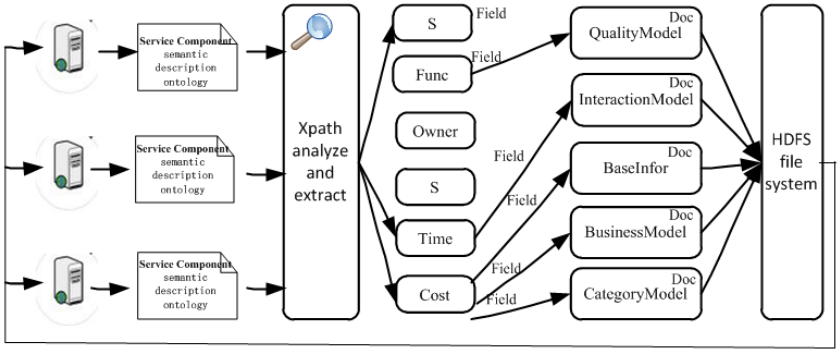


Figure 2. The Index Building Process

Xpath as a language for finding information in XML document can be used to navigate in XML documents via elements and attribute. By using path expression, nodes in an XML document can be selected. , with / represents to select node from the root node, // represents to select node from the current node that matches selection requirement, without caring about the current node's position, . represents the currently selected node, .. represents the parent node of the current selected node, @ represents the attribute of the current selected node.

This paper provides a method using Xpath to analyze and extract the semantic information semantic service components. Taking the following pseudo-code of QualityModel for example:

```
QualityModel qualityModel=new QualityModel;
qualityModel.S □/ QualityModel@S ;
qualityModel.Time □ / QualityModel@Time ;
qualityModel.Cost □ / QualityModel@Cost ;
qualityModel.Reliability□/QualityModel@Reliability
…….
```

As the upper 5 levels of ontology are concerned, multiple documents should be designated to store different semantic views of service components during the formulation of index files. Codes of Lucene are listed below:

```
Document  DocQualityModel= new Document();
Document  DocInteractionModel= new Document();
Document  DocBaseInfor= new Document();
Document  DocCategoryModel= new Document();
Document  DocBusinessModel= new Document();
```

Multiple Fields may also be designated in each Document to represent the semantic information. Codes for QualityModel are defined as:

```
Field FieldPath = new Field("S",qualityModel.S,Field.Store.YES, Field.retrieval.NO);
Field FieldPath = new Field("Time", qualityModel. Time,Field.Store.YES,
Field.retrieval.NO);
Field FieldPath = new Field("Cost",qualityModel.S,Field. Cost tore.YES,
Field.retrieval.NO);
Field FieldPath = new Field("Reliability",qualityModel.Reliability, Field.Store.YES,
Field.retrieval.NO);
Field FieldPath = new Field("Availblility",qualityModel.Availblility,Field.Store.YES,
Field.retrieval.NO)
Field FieldPath = new Field("Reputation",qualityModel.Reputation,Field.Store.YES,
Field.retrieval.NO);
Field FieldPath = new
Field("Compensation",qualityModel.Compensation,Field.Store.YES, Field.retrieval.NO);
Field FieldPath = new Field("Penalty",qualityModel.Penalty,Field.Store.YES,
Field.retrieval.NO);
```

Information would be written into corresponding documents through RetrievalWriter as soon as these operations are finished. Document regarded as a unit to be retrieved is an object to which every file should be converted for retrieval purpose. The index file can be stored in common PCs with a HDFS format, which requires neither server of high costs, nor PCs of high performance. Although unstable, unreliable and prone to be ineffective in performance, the adverse impacts brought by PC would be overcome by the fault-tolerance capability of Hadoop. Major strategies include:

1) for case of storage failure, Hadoop supplies multiple backup versions of user data for failure nodes' recovery from disasters.

2) for case of distributive computing failure, Hadoop supplies configurable computation schedule for backup tasks to finishing or terminating the distributive computing.

3. Index File Construction for Service Component

Distributed retrieval technique can be realized via Hadoop with high retrieval accuracy and high response speed to client's inquiry. A Map-Reduce mode within Hadoop architecture consist of two major procedures: the one that process primitive data and generate intermediate results, termed as Map procedure, and the one that generate final results, termed as Reduce procedure. Functions in Map procedure accepts a bunch of data and transfers it to lists composed of keys and corresponding values. Functions in Reduce procedure accept the generated list and make further optimization on the list's size, according to Key/Value pair.

Therefore, there should be at least a Map function, a Reduce function and a Run function for entry purpose. Task configurations can be made in the Run function, including the input/output pathway of Map function, the input/output format of Reduce function, and the computing strategy (type of Map function for input files, type of Reduce function for output files).

According to personalized retrieval requirement, the retrieval procedures within Hadoop architecture can be extended as the followings:

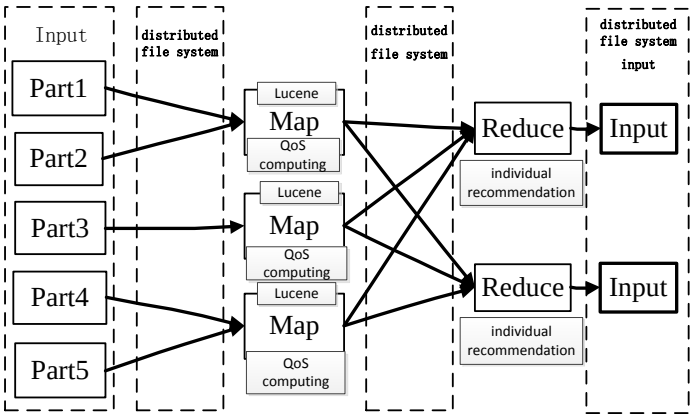


Figure 3. The Map-Reduce Based Service Component Retrieval Process

The retrieval procedures for service components are based on the extended mode of Map-Reduce. The descriptions of required service components are submitted by the clients and split into several inquiry fields. Inquiry requests are then sent out in a distributed manner, and proper service components would be retrieved by Luence within each server according the formulated index file. The service component once found would be transformed into Key/Value format, and would be processed in the manner of QoS computing. The Key/Value pair would be read and processed statistically. Finally, the processed data would be used in returning the personalized service set to clients.

The description file for service components submitted by clients can be defined in a BNF format:

```
Requirement-List::=(Quality||Interaction||BaseInfor||Category||Business)Requriment-List
```

Where Quality, Interaction, BaseInfor, Category, Business corresponds to views of QualityModel, InteractionModel, BaseInfor, CategoryModel, BusinessModel, respectively.

Define Quality function for QoS computing, which is used after Reduce procedure, as the following:

```
Quality:=(Time||Cost||Reliability||Availblility||Reputation||Compensation||Penalty);
Time=0…100;
Cost=0…100;
Reputation=0…100;
Compensation=0…100;
Penalty::=0…100;
Reliability::= 0…1;
Availability::= 0…1;
```

From the definition, the values of Time, Cost, Reputation, Compensation, Penalty in Quality function are integers, Reliability and Availability are float values between 0 and 1.

3.1. Map Process

The retrieval process of Luence is split into multiple tasks, carried out in designated servers for index files, respectively. A new retrieval searcher should be instantiated as the following:

```
path∈{QualityPath||InteractionPath||BaseInforPath||CategoryPath||BusinessPath};
retrievalSearcher searcher = new retrievalSearcher(HDFS.path);
```

where HDFS.path represents to retrieve existing pathways.

Query can be set up according to the submitted request and includes the following categories: TermQuery, BooleanQuery, RangeQuery, PrefixQuery, PhraseQuery, PharsePrefixQuery, fuzzyQuery, WildcardQuery.

Take the view of QualityModel to make illustration of using RangeQuery to inquire service components subjected to QoS requirement. The following codes represent inquiring service components with response time between 10~20 seconds:

```
Term beginTime = new Term("Time", "10");
Term endTime = new Term("Time", "20");
RangeQuery query=new RangeQuery(beginTime, endTime, false);
Hits hits = searcher.search(query);
```

Once the retrieval process completes, data will be output in a manner of <KEY, VALUE>, with data format Key1 : Value1 ; Key2 : Value2. Formulate the corresponding value of VALUE as:

$QoS::=[q_T, q_C,\ldots, q_{PE}]$,

where elements q_T, $q_C,\ldots$, q_{PE} correspond to attributes Time, Cost, Reliability, Availability, Reputation, Compensation, Penalty, respectively.

Weight of each element would be computed via an expectancy equation, given as the following:

$$q_T = \frac{Min\{q_{Ti}\} + 4*\sum_{i=1}^{n} q_{Ti} / n + Max\{q_{Ti}\}}{6} \quad (1)$$

where Min{} represents to choose the minimal response time in q_{Ti}, $Max\{\}$ represents to choose the maximal response time in q_{Ti}, $\sum_{i=1}^{n} q_{Ti} / n$ represents the mean value of response time.

After $QoS::=[q_T, q_C,\ldots, q_{PE}]$ is derived, a weight value will be assigned to each attribute, and the value of QoS will be computed. Let Time, Cost, Reliability, Availability, Reputation, Compensation, Penalty have weights in 1 to 7, respectively, then the computing equation would be given as:

$$QoS = \sqrt{\frac{q_T + 2*q_C + 3*q_{RE} + 4*q_{AV} + 5*q_{RP} + 6*q_{CO} + 7*q_{PE}}{7}} \quad (2)$$

After the QoS computing process, retrieval results of Map procedure fulfilling clients' requirements are derived. Since the computing process is distributive, the inquiry results should be subjected to further integration.

3.2. Reduce Process

The basis of the Reduce process is simply divided into several parts of the index which cannot be effectively classified, resulting in reducing the efficiency of the query. Therefore, on the basis of Reduce, this paper extends the process of personalized recommendation. Reduce procedure acquires intermediate computing results from TaskTracker nodes occupied by multiple Map procedures, and then dispatches the results to the service component sharing the same Reduce procedure.

Taking client's record as the premise, service component' response time expectancy, execution expense need, reliability, availability, credit degree, and compensation rate can be evaluated, with 2 denoting positive deviation, 1 denoting fulfillment, and -1 denoting negative deviation. The personalized evaluation equation is given as $\sum_{i=1}^{n} req_i$, and the recommended weights are given as $\sum_{i=1}^{n} req \times QoS$. The Reduce procedure processes statistically the Key/Value list and returns Top-k service components

4. Service Component Retrieval in Hadoop Architecture

In the performance experiments, we use the J. Meter [8] Apache as a tool to test the performance. Figure 4 is a two control group test, with the scale of 1000 service component library executing query requests. The left picture is the traditional retrieval method based on database of service component, and the right one is the method we proposed. While the average response time of the traditional method is up to 201ms, our method is only 29ms. It shows that it is stable in the case of large scale service component retrieval.

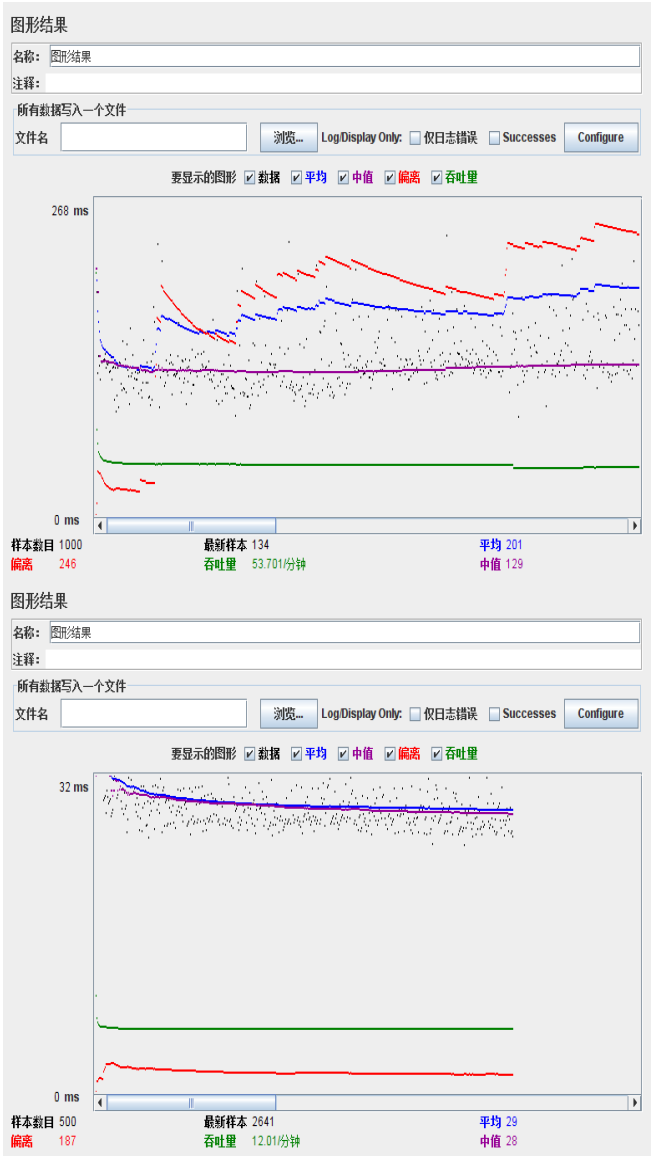


Figure 4. Experiment Results

5. Summary and Prospective Research

This paper discussed the retrieval problem of service component. A personalized service component retrieval method based on Hadoop is given, which uses the semantic description of the attributes and features of the service component in generating more understandable and formal semantics. With the help of the Lucene search engine to

complete the service component index files and query operations. By extending MapReduce mode of Hadoop, the operation of the distributed processing service component is more efficient, and the purpose of the personalized recommendation is realized. The next stage of the research objectives of this paper includes:

1) to research and to explore based on on-the-fly method of query patterns, and gradually expand the query method in the process. In the Hadoop framework, according to several queries, if you can determine the current index files on the server is inappropriate service component, you can end the current query operation.

2) to research the dynamic updating and merging model of service component index file.

Acknowledgements

This research was sponsored by Shanghai Natural Science Fund (No.14ZR1429800, 15ZR1430000) and Young University Teachers Training Plan of Shanghai Municipality under Grant (No.SXY12003). Experimental Course Reform Project for Commercial Application of Internet of Things (No. 16-12066-3343-04).

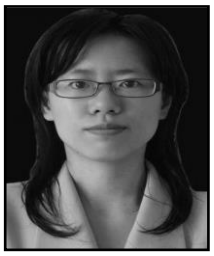
References

[1] L. F. Gui and Z. X. Chang, "Hadoop-based Measurement Report Parsing and Optimization", International Conference on Computer Science And Communication Engineering, **(2015)**, pp. 219-223.
[2] R. O. Herrero and V. A. Alberto, "Tuning small analytics on Big Data: Data partitioning and secondary indexes in the Hadoop ecosystem", Information Systems, vol. 54, **(2015)**, pp. 336-356.
[3] W. Weining, Z. Weijian and C. Chengjia, "An efficient image aesthetic analysis system using Hadoop", Signal Processing-Image Communication, vol. 39, **(2015)**, pp. 499-508.
[4] X. Xu, P. Fang and J. Xiao, "Hadoop distributed B2B platform based personalized recommending method, involves forming buyer identity knowledge base, storing calculation result into distributed data warehouse, and obtaining recommendation result", patent number CN103886487-A. patentee: Focus Technology Co. Ltd., **(2014)**.
[5] X. Runqun, L. Junzhou and D. Fang, "Optimizing data placement in heterogeneous Hadoop clusters", Cluster Computing-The Journal of Networks Software Tools and Applications, vol. 18, no. 4, **(2015)**, pp. 1465-1480.
[6] N. P. Gopalan, "Modified Delay Scheduling: A Heuristic Approach for Hadoop Scheduling to Improve Fairness and Response Time", Parallel Processing Letters, vol. 25, no. 4, **(2015)**.
[7] N. Paolo, "A Hadoop based platform for natural language processing of web pages and documents", Journal of Visual Languages and Computing, vol. 31, **(2015)**, pp. 130-138.
[8] Apache. JMeter. http://jakarta.apache.org/jmeter/

Authors

Wen Si, is an associate professor in department of information and computer science of Shanghai Business School. His research area includes Biomedical engineering and Internet of things technologies

Rong Tan, is an instructor in department of information and computer science of Shanghai Business School. His research area includes Spatial-Temporal Database, NoSQL and Computer Networks.



Lu Bing, is an instructor in department of information and computer science of Shanghai Business School. Her research area includes Signal processing and Biomedical engineering.