Building Service Composition based on Statistics of the Services Use

Fedorov Roman

Matrosov Institute for System Dynamics and Control Theory
of Siberian Branch of Russian Academy of Sciences, Irkutsk 664033, Russia
fedorov@icc.ru

Abstract. A distributed computing environment is being formed within the geoportal IDSTU SB RAS, which is mainly used for processing spatial data. Processing services that implement the WPS standard, and data storage services, a catalog of WPS services, and service execution systems and their compositions have been developed. In the system for executing services statistics data is collected on user-made service calls. The analysis of the obtained statistical data is carried out in order to identify the data transfer between services calls and build service compositions.

Keywords: SOA, WPS, spatial data processing, semantic network, composition of services.

1 Introduction

Now the number of Web services [1] is actively growing, providing processing and publication of data. For example, remote sensing data services, spatial data processing and publishing services. Storage and publication services are creating. In particular, these are cloud data storages, such as Internet services: Dropbox, OneDrive, Google Drive, iCloud, Yandex.Disk, Cloud Mail.Ru, etc. The developed services greatly simplify applying various algorithms and data acquisition, excluding the often complicated process of installing and configuring software. Standards and specifications have been developed and applied that unify the call of services, the transmission and reception of data, for example, the OGC WPS spatial data processing standard [2], the set of REST specifications [3]. Service catalogs are created that contain meta-information and allow you to search for necessary services. The developed services are actively used by users to solve many problems.

Often, the solution of a problem may require repeating a certain sequence of services calls, where the user enters the same parameter values for each service. To automate the user's work, it is necessary to create a composition of Web services [4]. The standards for setting service compositions are BPEL (Business Process Execution Language), BPMN (Business Process Modeling Notation), BPML (Business Process Management Language), and XPDL (XML Process Definition Language). In addi-

tion, programming languages are actively used to define service compositions. There are a number of workflow systems that allow you to create service compositions and have achieved significant results in various subject areas: Pegasus [5], Kepler [6], Swift [7], KNIME [8], Taverna [9], Galaxy [10], Trident [11] and Triana [12], Everest (Mathcloud) [13], CLAVIRE [14]. For processing spatial data, Geo-Processing Workflows is actively used [15]. The generally accepted standard for determining the composition of services is Directed acyclic graph (DAG) [16, 17], in which the vertices are the service calls, and the arcs are the dependencies between services according to the data. Creating a service composition is a complex process that requires programming skills from the user. The authors propose a method that automates the creation of service compositions based on statistical data of the services use by the user. This method allows you to connect single service calls to each other, to determine the dependencies from the data.

2 Statistics collection

Within the framework of the geoportal IDSTU SB RAS, a distributed computing environment is being formed, which is mainly used for processing spatial data. Processing services that implement the WPS standard, and data storage services, a catalog of WPS services, and service execution systems have been developed. The interaction between processing services and data storage services is implemented, which allows simplifying the use of services by the user. In the system for executing services and their compositions, statistics of service calls is collected. The statistics data includes the name of the service and its address, values of input and output parameters, service execution time, successful execution, execution errors, etc.

3 Definitions

We introduce the following notation necessary to describe the operation of the method.

s = <name, I, O> is the service, where «name» is the name of the service, I is the set of input parameters, O is the set of output parameters of the service. Below we will denote s.I and s.O parameters that belong to a particular service. A user service call has input parameter values specified by the user and output parameter values.

$t = <V_I, V_O>$ - service s call with values V_I, V_O of parameters s.I and s.O.

Log is the set of successful service calls made by users. The cause of the unsuccessful service call may be incorrect parameter values. Therefore, statistics are filtered and only successful calls are left. Based on the Log data, it is necessary to form a set of service compositions in the form of DAG.

4 Defining data transfer between two service calls

To recognize a service composition, it is necessary to determine the presence of data transfer between two service calls. The determination of the relationship between two service calls can be made based on the analysis of parameter values. Parameters of service call can be divided into input and output. It is assumed that if the value of the output parameter matches the value of the input parameter, then there may be data communication between service calls. Parameter values can be string, numerical type, or resource URIs. Usually these are URLs to files, data services, etc. Each URI used to receive data uniquely identifies the transmitted data. Accordingly, using URI parameters, data transferring from one service call to another can be determined. Analysis of string and numerical parameters is more difficult, because of matching the values may be casual. Currently, string and numeric parameters are not considered.

Here is an algorithm for checking data transfer between two service calls.

```
Input: ti, tj - two service calls
Output: flag - connection between service calls and by what parameters
function islinked
    flag <- False
    for each vm in ti.VO:
        for each vn in tj.VI:
            If (typeof vm = file and typeof vn = file and vm = vn) then
                flag <- True
                vn.linkedwith <- vm
            end if
        end for
    end for
return flag
```

This algorithm compares the output parameters of one service call with the input parameters of another service call. If the URL values of the parameters match, the connection between service calls is established. In addition, it is determined by what parameters they are connected.

5 Building service compositions based on statistics

To analyze the set of successful calls to Log services, the following algorithm is used, at the output of which we obtain a general directed acyclic graph DAG_{gen}, where calls to t_i services are connected by data. Additionally, we obtain the values of the parameters of services entered by the user.

```
Input: Log
Output: list - semantic relationships between services
for each t_i in Log:
  for each t_j in Log:
```

```
    If islinked (ti, tj) then
        add (ti, tj) into list
  end for
end for
```

A user can perform a data-connected sequence of service calls that forms a connected subgraph D_i (graph connectivity components) in DAG_{gen} that is not connected to other subgraphs. Each such subgraph D_i is a special case of the implementation of the composition of services. To search for D_i subgraphs, the width search algorithm is used. Moreover, the complexity of the search is linear. It depends from the sum of the number of vertices and the number of edges of the graph DAG_{gen}.

The user can often repeat the same sequence of service calls. This leads to the creation of several D_i subgraphs in which calls of the same services are made, but the parameter values differ, i.e. creating isomorphic subgraphs D_i, considering only services and data transfer. Checking for isomorphism of graphs is carried out by traversing oriented subgraphs.

As a result of the execution of the algorithm, we obtain several sets of isomorphic $IDAG_i$ subgraphs. The number of D_j subgraphs in $IDAG_i$ is an estimate of the frequency of use of this services composition. Next, filtering isomorphic $IDAG_i$ subgraphs is carried out, for which the number of vertices is more than one and there is at least one call to the service for publishing data. The resulting composition of services is shown in Fig. one.

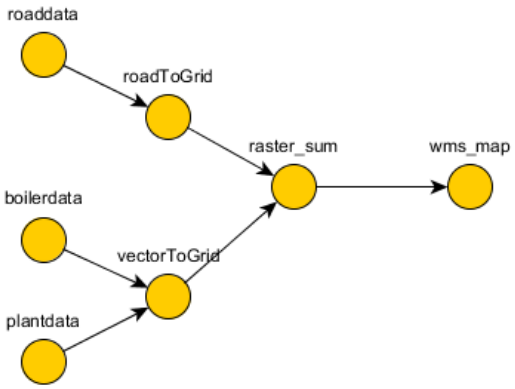

Fig. 1. The resulting services composition.

6 Conclusion

The proposed method, in contrast to existing approaches, uses statistics to build service compositions. To build a composition, the user only needs to sequentially perform the necessary services. The method automatically connects single service calls to each other, determines data dependencies. The application of the method is focused on the automation of frequently repeated user actions.

This work was supported in part by the Russian Federal Property Fund (grant 18-07-00758-a, 17-57-44006-mong-a, 17-47-380007-r), the program of the Presidium of the Russian Academy of Sciences No. 2, the program “Information and telecommunication platform for digital monitoring of Lake Baikal, based on end-to-end technologies”, Shared Equipment Centers of ISDCT SB RAS.

References

1. Grimm, S. Ontologies and the Semantic Web / S. Grimm, A. Abecker, J. Volker, R. Studer // Handbook of semantic web technologies: foundations and technologies, VOLS 1 and 2. — 2011. —P. 507-579.
2. Schut, P. OpenGIS ® Web Processing Service / P. Schut // Open Geospatial Consortium. — 2007. — № 6. — P. 1–3.
3. Pautasso, C. RESTful Web service composition with BPEL for REST / C. Pautasso // Data knowledge. — 2009. — Vol. 68, № 9. — P. 851–866.
4. Hoffmann, J. Web Service Composition / J. Hoffmann, I. Weber // Encyclopedia of Social Network Analysis and Mining. — Springer-Verlag, 2014.
5. Deelman, E. Pegasus, a workflow management system for science automation / E. Deelman, K. Vahi, G. Juve // Future Generation Computer Systems 46C. P. 17–35.
6. Ludäscher, B. Scientific Workflow Management and the Kepler System / B. Ludäscher, Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, Y. Zhao // Special Issue: Workflow in Grid Systems. Concurrency and Computation: Practice & Experience. 2006. Vol. 18(10). P. 1039-1065.
7. Wilde, M. Swift: A language for distributed parallel scripting / M. Wilde, M. Hategan, J.M. Wozniak // Parallel Computing. 2011. Vol. 37(9). P. 633–652.
8. Berthold, M.R. The konstanz information miner / M.R. Berthold, N. Cebron, F. Dill // SIGKDD Explorations 11. 2009. P. 26–31.
9. Wolstencroft, K. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud / K. Wolstencroft, R. Haines, D. Fellows // Nucleic Acids Research. 2013. Vol. 41(W1). P. 557–561.
10. Blankenberg, D. Galaxy: A Web-Based Genome Analysis Tool for Experimentalists. / D. Blankenberg, G.V. Kuster, N. Coraor. Wiley. 2010.
11. Simmhan, Y. Building the trident scientific workflow workbench for data management in the cloud / Y. Simmhan, R. Barga, C. Ingen // Advanced Engineering Computing and Applications in Sciences (ADVCOMP). 2009.
12. Churches, D. Programming scientific and distributed workflow with Triana services: Research articles / D. Churches, G. Gombas, A. Harrison // Concurrency and Computation: Practice and Experience. Vol. 18(10). P. 1021–1037
13. Smirnov, S. Integration and Combined Use of Distributed Computing Resources with Everest / S. Smirnov, O. Sukhoroslov, S. Volkov // Procedia Computer Science. 2016. Vol. 101. P. 359-368.
14. Boukhanovsky, A.V. CLAVIRE: Perspective Technology for Second Generation Cloud Computing / A.V. Boukhanovsky, V.N. Vasilev, V.N. Vinogradov, D.Y. Smirnov, S.A. Sukhorukov, T.G. Yapparov // Scientific and technical journal «Priborostroenie». 2011. Vol. 54.
15. Chen, NC. Geo-processing workflow driven wildfire hot pixel detection under sensor web environment / N.C. Chen, L.P. Di, G.N. Yu, J.Y. Gong // Computers & geosciences. 2010. Vol. 36, №: 3. P. 362-372.

16. Kwok, Y.-K. Static scheduling algorithms for allocating directed task graphs to multiprocessors / Y.-K. Kwok, I. Ahmad // ACM Computing Surveys. 1999. Vol. 31, № 4. P. 406–471.
17. Xie, G. A High-Performance DAG Task Scheduling Algorithm for Heterogeneous Networked Embedded Systems / G. Xie, R. Li, X. Xiao, Y. Chen // Proc. of IEEE 28th International Conference Advanced Information Networking and Applications. 2014. P. 1011–1016.