Int. J. Communications, Network and System Sciences, 2016, 9, 152-159
Published Online May 2016 in SciRes. http://www.scirp.org/journal/ijcns
http://dx.doi.org/10.4236/ijcns.2016.95014

Scientific Research Publishing

A Cross-System Invocation Platform Based on Distributed Network Performance Measurement System

Zhenwen Xue[1], Yuehui Jin[1], Tan Yang[1,2]

[1]State Key Laboratory of Networking and Switching Technology, University of Posts and Telecommunications, Beijing, China
[2]School of Software Engineering Beijing, University of Posts and Telecommunications, Beijing, China
Email: Xuezhenwen728@gmail.com, yhjin@bupt.edu.cn, tyang@bupt.edu.cn

Received 12 April 2016; accepted 24 May 2016; published 30 May 2016

Abstract

The Distributed Network Performance Measurement Sys-tem provides functions to derive performance indices of networks and services, which are significant for Network Management System. To make these two systems cooperate, we realize this cross-system invocation platform, using Web Service, a mechanism which allows two systems to exchange data over the internet through publishing interfaces [1]. There are several mature Web Service frameworks, Apache Axis2, Apache CXF etc. In this paper we choose Apache Axis2 to achieve the objective that the Network Management System can invocate the net-work performance measurement functions via the Web Services.

Keywords

Network Measurement, Cross-System, Web Service, Apache Axis2

1. Introduction

TANC is a distributed network performance measurement system, and provides a method and device which is used to measure the IP network. It is divided into two subsystems, *i.e*. a cloud monitoring platform and measuring probes. **Figure 1** illustrates the overall architecture of the system.

The cloud monitoring platform which consists of a management platform (Arbiter), several Data Analyzers (DA) and a Load Balancer (LB), uses cloud computing technology to complete the performance data integration, consistency of treatment and other processes. Among those components, the management platform is the core control platform, which is responsible for managing the dynamic probe deployment, adjustment and measuring tasks. The main task of DA is to collect raw measured data from different probes, In order to balance the amount of DA to the probe based on the load information of DA.

Measuring probe subsystem consists of probes deployed in large-scale network. Probe implements a variety of measuring tasks of network performance index, collects the resource information itself at the same time and reports it to the cloud monitoring platform in real time. During the collection process, the management platform

How to cite this paper: Xue, Z.W., Jin, Y.H. and Yang, T. (2016) A Cross-System Invocation Platform Based on Distributed Network Performance Measurement System. *Int. J. Communications, Network and System Sciences*, **9**, 152-159.
http://dx.doi.org/10.4236/ijcns.2016.95014

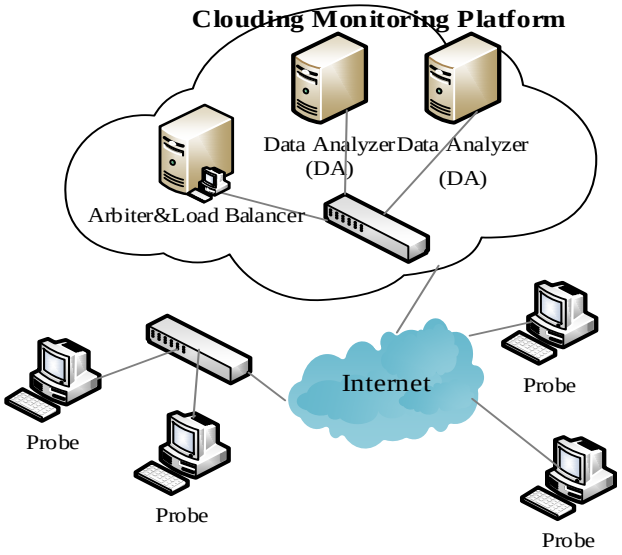

Figure 1. The distributed network performance measurement system.

will adjust the location of probes flexibly relay on the resource status of probes [2].

The rest of the paper is organized as follows: Section 2 introduces the design of the invocation platform of the TANC system. Section 3 describes the interfaces which should be provided for the Network Management System, and the process of the service publication. Section 4 presents the methods which we use to evaluate web service. We propose problems for future and other works of our study in Conclusion.

2. Platform Design

The cross-system invocation platform provides network performance measurement functions of the TANC system. The available indices includes time delay, packet loss and jitter rate etc. The Network Management System or other systems invocate the open interfaces of the platform. On one hand, the platform can make full use of the integrated measurement functions, reuse the software, improve the efficiency of TANC system and exchange the data across systems. On the other hand, it can promote the loose coupling of internal system. For example, the web of the TANC system can call the interfaces of the platform, rather than directly execute the reading and writing operations to the database. **Figure 2** shows the invocation platform architecture.

Web Service is a method of communication that allows two different software systems to exchange data over the internet. Different software can be built using different programming languages, and hence there is a need for a method of data exchange that does not depend upon a particular programming language. However, most types of software can interpret XML tags, and Web Service use XML files for data exchange [3]. Thus Web service is a good choice to use to implement the cross-system invocation.

Axis2 and CXF are two Web Service frameworks both belong to the Apache Software Foundation. Axis2 comes from the well-known Axis1.x series, and is a total rewrite of Axis from the ground up. It uses a new modular architecture that allows its functionality to be more easily extended. CXF is literally the offspring of the XFire and Celtix projects, and has been extensively retooled. The chief differences between two frameworks are as follows:

CXF has support for WS-Addressing, WS-Policy, WS-Security, and WS-I BasicProfile. Axis2 support each of these except for WS-Policy, which will supported in an upcoming version. CXF was written with Spring in mind, but Axis2 is not. Axis2 supports a wider range of data bindings, including XMLBeans, JiBX, JaxMe and JaxBRI as well as its own native data binding, ADB. Note that support for JaxME and JaxBRI are still considered experimental in Axis2 1.2. CXF currently supports only JAXB and Aegis, support for XMLBeans, JiBX and Castor will come in CXF 2.1.Axis2 supports multiple languages. There is a C/C++ version available in addition to Java version [4]-[6].

In this paper we choose Apache Axis as the frameworks to publish the service.

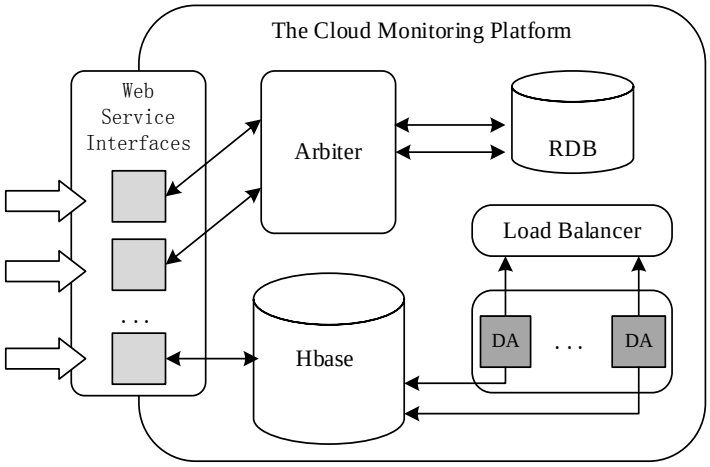

Figure 2. Invocation platform architecture.

3. Interface Definition and Service Publication

3.1. Interface Definition

The main functions of TANC system include task deployment, task state information query, and the return of the task result etc. the platform needs to provide the interfaces of related functions, thus the Network Management System or other system can call the interfaces in the process to complete related operations. **Figure 3** shows the interface invocation flow chart.

Before deploying task, it is necessary to request the topology information, probes information and task types.

The network topology information query interface requires administrator ID and password as parameters, and returns all the networks the system have built up. Requestor can choose which network need to be measured via the information requestor get. The information includes network topology name and its ID etc.

When requestor need to deploy single link performance measurement task, the probes information query interface should be invocated first, which is used for service requestor to get the available probes in a network, and it need network topology ID as a parameter. The information is stored in the system database. If ID is valid, the system will query the database based on the parameter and return a string array containing IP of available probes.

The task types query interface provides requestor with the network performance indices which TANC can measure. It does not need any input parameters and if the invocation succeed, the interface will return the available types.

The task which requestor can deploy includes single link performance measurement task and network performance measurement task. The single link performance measurement task requires two probe IPs, task type, task start time and end time as parameters, then the system checks the validity and generates a task record if all parameters are valid. The network performance measurement task requires network topology ID instead of probe IPs as parameter.

Task status query interface is used when the requestor needs to know the status of a certain tasks, and returns to requestor the task status at the moment. The status includes task been deployed, task into the waiting queue, task executing, task success, failed to carry out task. When requestor invokes the interface, the task ID need to be provided as a parameter, then the system returns a string representation of the status after querying the database.

By calling the task result query interface, requestor can view the results of the completed tasks, and make the judgment of network performance base on the analysis results. This interface also requires a task ID as interface parameter, and returns the task result data as a JSON format string. The task data is stored in the Hbase after the DA receives from the probes and analyses it.

Other interfaces can be designed and implemented on the invocation platform, which can make the invocation more efficient. The above several interfaces provide the most basic functions to complete a network performance

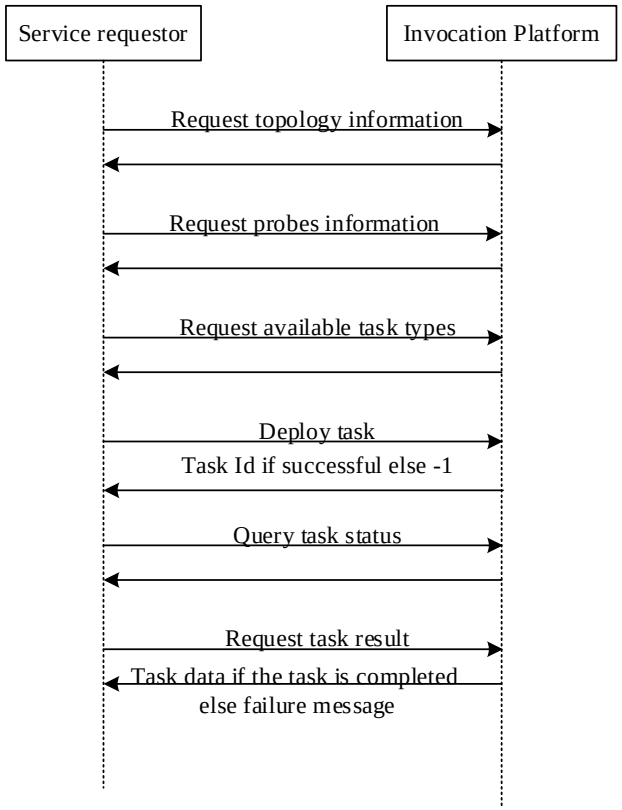

Figure 3. Interface invocation flow chart.

measurement task.

3.2. Service Publication

Axis2 has taken an approach that makes it in many ways resemble an application server in miniature. Axis2 comes packaged with a WAR that can be deployed on a servlet container such as Tomcat that is designed to make web services easier to manage and deploy. The Axis2 Web Administration module allows Axis2 to be configured dynamically. While applications are running, new services can be uploaded, activated or deactivated and their parameters may be changed. The administration UI also allows modules to be enabled on one or more running services. The only downside to using the UI for these purposes is the fact that configuration changes made through it are not persistent. They go away when the servlet container is restarted. Axis2 lends itself towards web services that stand alone, independent of other applications, and offers a wide variety of functionality and a good model for adding more functionality as time goes on through its modular architecture [4] [6].

Comparing with Apache Axis2, Most configuration of Apache CXF is done via the API instead of XML files. Spring integration is heavily emphasized, including support for Spring 2.0 and APIs of CXF. Spring configuration mirror one another fairly closely. CXF emphasizes code-first design, using simple APIs to make development of services from existing applications easier [5] [6].

In publishing the web service, the organization provides a service description containing the interface and implementation details including its data types, operations, binding information and network location. This definition is constructed using WSDL [7]. We publish web service by means of Eclipse in combination with Apache Axis2, and use two Eclipse plugins, the Service Archive Wizard as well as the Code Generator Wizard. Service Archive Wizard is used to package service code as suffix .aar file, and Code Generator Wizard is used to generate the WSDL file and parse the WSDL file to generate the client code. We just need to put the two plug-in jar packages in Eclipse directory under the plugins folder, restart the Eclipse. Then we need get an axis2.war file via unzipping axis2-1.6.2-war.zip, and put it into the webapps file of tomcat, start the tomcat.

Take the interface file GetMapInfoByUserID.Java as an example. Service code should be packaged into .aar

file, here we use the above mentioned Axis2 Service Archiver plug-in. After a series of steps, the corresponding GetMapInfoByUserID.aar file is generated. Then the file needs to be put in tomcat/webapps/Axis2/ WEB_INF/ services/folder below. Other class file and configuration files need to be put in tomcat/webapps/Axis2/ WEB_INF/classes. Restart the tomcat. And then visit http://localhost:8080/axis2/, click on Services, and a web service which is related with getting all network topology information is shown. Click on the web service and the web page displays the related WSDL information.

Figure 4 introduces the structure of the WSDL file. The WSDL file is a simple XML document, which contains a series of definition that describes a web service. The type element defines the data types of web service. For maximum platform neutrality, WSDL uses XML Schema syntax to define data types. The message element defines the data elements of an operation. Each message is composed of one or more parts which can be compared to the parameters of a function in a traditional programming language. The portType element is the most important WSDL element. It describes a web service and its operations that can be performed as well as other related information. The binding element defines the message format and protocol details for each port [8].

4. Service Interface Evaluation

After getting the published Service WSDL file, it is necessary to test that if it can provide normal services, one method is by using the Axis2 Code Generator plugin to generate the client code, but another popular method is by SoapUI which is an open source test tools via soap/HTTP to check and call. Apache JMeter is used to test the performance of the interfaces.

4.1. Client Code

Axis2 Code Generator plugin can generate client code easily and automatically according to the WSDL file, thus we can simply call the generated Code to complete the web service invocation. At the same time, we can also try to invoke the service interface on other platforms, with the Android development platform as an example, it is practicable to invocate the interface with the help of the ksoap2, a client develop library which is more suitable for the Android mobile platform.

Figure 5 describes the Android Ksoap2 sends web service request to the server to get all network topology information in the system.

4.2. SoapUI

SoapUI is an open-source web service testing application for service-oriented architectures (SOA) and repre-

```
<definitions>

<types>
        Definition of types
</types>

<message>
        Definition of a message
</message>

<portType>
        Definition of a port
</portType>

<binding>
        Definition of a binding
</binding>

</definitions>
```

Figure 4. The structure of WSDL file.

```
public class GetMapInfoByUserId {
    public ArrayList<String> execute(String userid) {
        SoapObject request = new
SoapObject("http://webService",
"getMapInfoByUserID");
        request.addProperty("userid", userid);
        SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
        envelope.bodyOut = request;
       HttpTransportSE ht = new
HttpTransportSE("http://10.108.101.25:8080/axis2/serv
ices/GetMapInfoByUserID?wsdl");
      try {
 ht.call(null, envelope);
      } catch (HttpResponseException e) {
            e.printStackTrace();
 } catch (IOException e) {
            e.printStackTrace();
       } catch (XmlPullParserException e) {

            e.printStackTrace();}
       ArrayList<String> list =
new ArrayList<String>();
       Gson gson = new Gson();
       try {
          if(envelope.getResponse()!= null){
             Object object =
(Object)envelope.getResponse();
             String jsonString = object.toString();
             list = gson.fromJson(jsonString,
new TypeToken<ArrayList<String>>(){}
.getType());
          }else{
             list = null;
          }
       } catch (Exception e) {
         e.printStackTrace();}
      return list;
    }
}
```

Figure 5. The request by android Ksoap2.

sentational state transfers (REST). Its functionality covers web service inspection, invoking, development, simulation and mocking, functional testing, load and compliance testing. SoapUI can test SOAP and REST web services, JMS, AMF, as well as make any HTTP(S) and JDBC calls [9]. We can use SoapUI software or SoapUI plug-in in Eclipse to complete our testing. A new project need to be created based on the URL of the WSDL file, and the test result of network topology information query interface is shown as follows.

Figure 6 and **Figure 7** describes the success of the request and response operations using SoapUI to test the basic information of the network topology respectively. For example, the requester need to provide the User ID, here is admin, as a parameter, then the interface return the information as a JSON string, which contains the ID information of network topology and its name.

4.3. Performance Evaluation

JMeter, which is developed by Apache, is applied to the performance evaluation of Web applications widely. It

can be used in the simulation under the condition of heavy load performance of the entire server. The test of JMeter, including the creation of loops and thread group, uses default delay to simulate continuous requests to the server, and the thread group is designed to simulate concurrent load [10].

We choose three different thread numbers, 5, 50, and 500 to test the performance of the network topology information query interface. Throughput is the number of requests the server can deal with per minute. Average is the total running time divided by the number of requests sent to the server. Median represents the time, which half of the server response time less than it [10]. From **Table 1**, the throughput of 5 threads is less than half of 50 and 500 threads, which two has a similar value, about 5500 per minute. Both average and Median is increased dramatically as the number of the concurrent threads increases.

5. Conclusion

In this paper, we choose the Web Service technology to implement the cross-system invocation platform based

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/enve
 lope/"            xmlns:web="http://webService">
 <soapenv:Header/>
       <soapenv:Body>
          <web:getMapInfoByUserID>
            <!--Optional:-->
         <web:userid>admin</web:userid>
        </web:getMapInfoByUserID>
      </soapenv:Body>
</soapenv:Envelope>
```

Figure 6. The request of SoapUI.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envel
ope/">
<soapenv:Body>
    <ns:getMapInfoByUserIDResponse
xmlns:ns="http://webService">

<ns:return>["0_liveserver","1_base","6_test","9_111
11","10_PlanetLab","10006_rzz","10008_EssayTest
2","10010_test3","10011_test4","10012_test5","1001
7_EssayTest4","10020_22","10021_ACO","10024_M
atLabTest1","10033_teststate","10037_testtest""100
41_unnamed","10042_1","10044_ccc","10045_cccc",
"10046_cc2","10047_xiami","10048_newbase","1004
9_tututu"]
</ns:return>
    </ns:getMapInfoByUserIDResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 7. The response of SoapUI.

Table 1. The interface performance evaluation.

Threads	Throughput/min	Average/ms	Median/ms
5	2676.182	62	66
50	5436.752	363	147
500	5702.41	3392	1123

on a distributed network performance measurement system TANC. Web service is an independent, loosely coupled, self-contained platform based on the programmable Web application. Different applications running on different environments can exchange data or integrate by web service without using additional third party system. We define basic interfaces required to complete a measurement task, including deploying task, task status view, task results back, etc. Apache Axis2 is a popular Web Service framework, and we use it to publish the service and generate the corresponding WSDL file. The services are tested by two ways, the client code test and a convenient test tool, SoapUI. And in the end we use Apache JMeter to evaluate the performance of the interfaces. More interfaces of the TANC system need to be published in the future in order to make the invocation more efficient.

Acknowledgements

This work was sponsored by the 973 Project of China (No.2009CB320505), the 863 Project of China (No.2011AA01A102) and the Fundamental Research Funds for the Central Universities of China (No. 2014RC0501).

References

[1] https://en.wikipedia.org/wiki/Web_service

[2] Chen, X., Jin, Y.H. and Yang, T. A Low Coupling Analysis Method for Large-Scale Network Performance. *Proceedings of* 2013 5*th IEEE International Conference on Broadband Network & Multimedia Technology*. http://dx.doi.org/10.1109/icbnmt.2013.6823906

[3] Papazoglou, M.P. (2009) Web Servies Prinples and Technology. China Machine Press.

[4] http://axis.apache.org/axis2/java/core/

[5] http://cxf.apache.org/docs/index.html

[6] Townsend, B. (2007) Axis, Axis2 and CXF: Surveying the WS Landscape. TheServerSide.com.

[7] McGregor, C. and Schiefer, J. (2003) A Framework for Analyzing and Measuring Business Performance with Web Services. *Proceedings of the IEEE International Conference on E-Commerce* (*CEC*'03).

[8] http://www.w3school.com.cn/wsdl/wsdl_documents.asp

[9] http://www.soapui.org/about-soapui/what-is-soapui-.html

[10] Nevedrov, D. (2006) The Application of JMeter in WebService Performance Test. dev2dev.