Available online at www.sciencedirect.com
ScienceDirect
Nuclear Science and Techniques 19 (2008) 74–78
NUCLEAR SCIENCE AND TECHNIQUES

Web services interface to EPICS channel access

DUAN Lei[1,2] SHEN Liren[1,*]

[1] *Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China*

[2] *Graduate University of the Chinese Academy of Sciences, Beijing 100049, China*

Abstract Web services is used in Experimental Physics and Industrial Control System (EPICS). Combined with EPICS Channel Access protocol, Web services' high usability, platform independence and language independence can be used to design a fully transparent and uniform software interface layer, which helps us complete channel data acquisition, modification and monitoring functions. This software interface layer, a cross-platform of cross-language, has good interoperability and reusability.

Key words EPICS, Channel access, Web services, Middleware

CLC numbers TP273^{+}.5, TP393

1 Introduction

Experimental Physics and Industrial Control System (EPICS) is a set of Open Source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for large scientific facilities[1], including the Shanghai Synchrotron Radiation Facility (SSRF).

Using EPICS, with its perfect function, development of the control system can be done in a convenient process, and stability and reliability of the devices can be greatly improved. However, the developers using different platforms and languages should possess profound knowledge on all kinds of the development. This complicates the development process and the source code maintenance, let alone difficulties in high-level applications to similar equipment.

In order to improve portability, interoperability and reusability of high-level application in EPICS, we decided to use some middleware platforms, i.e. component model of Java-based J2EE (SUN), COM/DCOM (Microsoft) and CORBA (OMG). As each platform has its own internal protocol, many "Information islands" will be built up. In order to obtain a wider scope of cross-platform and cross-language sharing, we use Web services technology, and establish a fully transparent and unified software interface layer on EPICS. In this paper, we report a Web services interface to EPICS channel access, which provides another way for high-level application developers to access the bottom channels.

2 EPICS channel access (CA) protocol

The EPICS control system consists of Operator Interface (OPI), I/O Controller (IOC) and Device Controller. A software bus between OPI and IOC, the Channel Access (CA) protocol, is an application layer protocol based on client/server (C/S) model, and it is based on the TCP/IP network protocol[2]. Process Variable (PV) is the basic unit for control. A channel is established when PV connects to the OPI client application by channel access protocol. The connecting process between CA client and CA server is shown in Fig.1[3].

The CA client sends name search requests to a list of server destination addresses to determine the IP

* Corresponding author. *E-mail address:* shenliren@sinap.ac.cn

Received date: 2008-01-08

address of the server on which the channels Process Variable resides. These server destination addresses can be IP unicast addresses or IP broadcast addresses. If one of the servers reachable by the address list knows the IP address of a CA server that can serve one or more specified Process Variable, it sends back a response containing the CA server's IP address and port number to the CA client. The CA client and CA server establish a TCP, which can be reused. This means that several PVs on the same server use the same TCP connection.

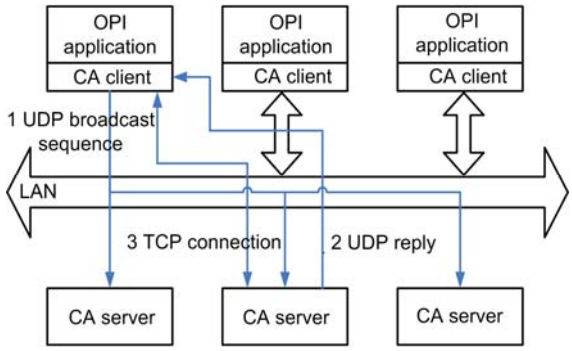

Fig.1 The search and connect procedure between CA client and CA server.

3 Web services introduction

The main objective of Web services is to establish a generic platform-independent and language-independent technology layer based on different platforms. And applications from various platforms implement their connectivity and integration by the technology layer. Web services solve the problem of limited interoperability, thereby enhance and expand functions of the distributed computing. Theoretically, Web services allow two or more software components to communicate with each other, regardless of the component using whatever technology and deploying on whatever platform. In addition, the Web services-based applications are easier to debug, because Web services use the text-based communication protocol (such as HTTP), rather than binary communication protocol used in DCOM and CORBA. To achieve a complete Web services system requires a series of protocols to support. The entire Web services Technology System protocol stack is given in Table 1.

Table 1 Web services protocols

Web services protocols based on XML(eXtensible markup language)				Existing network protocols	
Service workflow	Service discovery/integration	Service description	Call service	Data transmission	Internet
WSFL (Web services flow language)	UDDI (Universal description, discovery and integration)	WSDL (Web services description language)	SOAP (Simple object access protocol)	HTTP, FTP, SMTP	Ipv4, Ipv6

4 Design and implementation

4.1 System software architecture

We use J2EE as platform to develop Web services, and choose Sun Java System Application Server as the application server to deploy Web services. Because Java is the main development language on server-side, we need to use JCA (Java Channel Access)[4]. JCA is a CA library provided to Java. It establishes the connection to Channel Access by calling CA API, and satisfies the demand of Java developers. Therefore, the system calls JCA API directly to communicate with the CA server[5]. The system software architecture is shown in Fig.2.

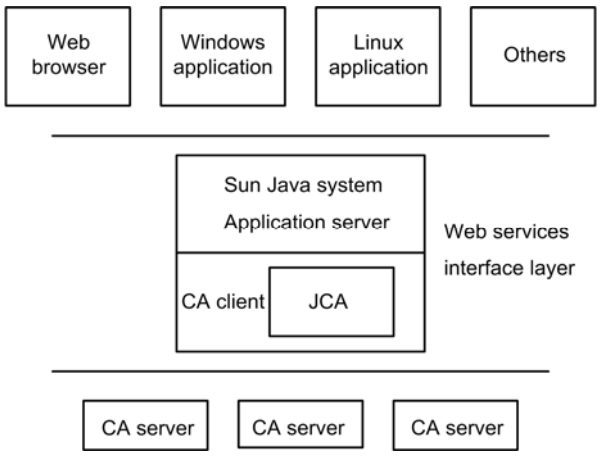

Fig.2 System software architecture.

4.2 Interface design

The work steps of CA protocol (here using JCA API) are as follows:

(1) Use JCALibrary.getInstance() to initialize CA.

(2) Use JCALibrary.createContext() to establish context.

(3) Use context.creatChannel() to search channel, and use context.pendIO() to do overtime processing.

(4) Do getting, setting, monitoring and other operations through the channel.

(5) Use channel.destroy() to destroy the channel, and use context.destroy() to destroy the context.

In order to acquire a channel's status and accomplish the operations through a channel, we developed Web services interfaces as shown in Table 2[6].

Here, NetBeans 5.5 is used as the integrated development environment, and the various Web services interfaces will be deployed in the Sun Java System Application Server[7]. Fig.3 shows the invoked procedure.

Table 2 Web services interface

Interface names	Interface definitions	Functional description
caGetService	DBR caGetService(string channel)	Get PV value
caMonitorService	string caMonitorService(string channel, int timeout)	Monitor PV value
caPutService	string caPutService(string channel, DBR value)	Set PV value
caStateService	string caStateService(string channel)	Get channel status
caInfoService	string caStateService(string channel)	Get channel information

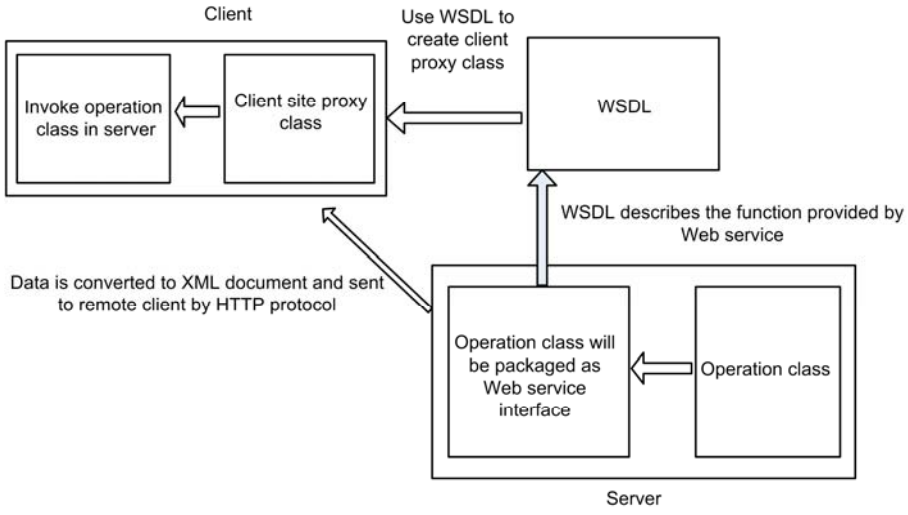

Fig.3 The invoked procedure of Web services interface.

Javax.jws.WebMethod, javax.jws.WebParam and javax.jws.WebService are used when adding web service operation to server's source code in NetBeans5.5. These packages help us to create Web services. The following shows how to use the three packages:

```
@WebService()
public class CAWebServices {
@WebMethod
public String caGetService(@WebParam(name = "channelname") String channelname) {
// Use JCA API to get the value and return result as the previous work steps of CA protocol
}
```

4.3 The synchronous and asynchronous Web services client

We have two calling modes in the development of Web services client:

1) Synchronous call. The service-request threads will be waiting for request results from the server. And the service-request threads will be in a wait state all the time if the server processing time is too long.

2) Asynchronous call. If the server processing

time is too long, the service-request threads will deal with other services after sending requests. And the service-request threads will be notified to response for the processing results after receiving Web services results.

In actual systems, the monitoring of the PV needs some time. Therefore we use asynchronous call to develop the client for caMonitorService interface. At the same time, asynchronous Web service clients consume Web services either through the "polling" approach or the "callback" approach. In order to reduce network load, we are using "callback" approach, namely, the servers notify clients only when the value of PV changes, and return the changed value.

4.4 Tests and comparison

4.4.1 Web services tests

In order to test the performance of Web services, we use SOATest from Parasoft Company (originally called SOATTest). It uses WSDL, which describes the service and notifies the address, to control visits to Web services. The test PC is P4 2.4GHz, 512MB, Windows XP Professional. Testing time is 10 minutes, and the number of virtual users uses bell-type division. The number is from 0 to 100, and then from 100 down to 0. Web services test results is shown in Fig.4.

The test PC is not only used as server, but also to create a large number of virtual users. From Fig.4 statistics, the average time is 2799 ms.

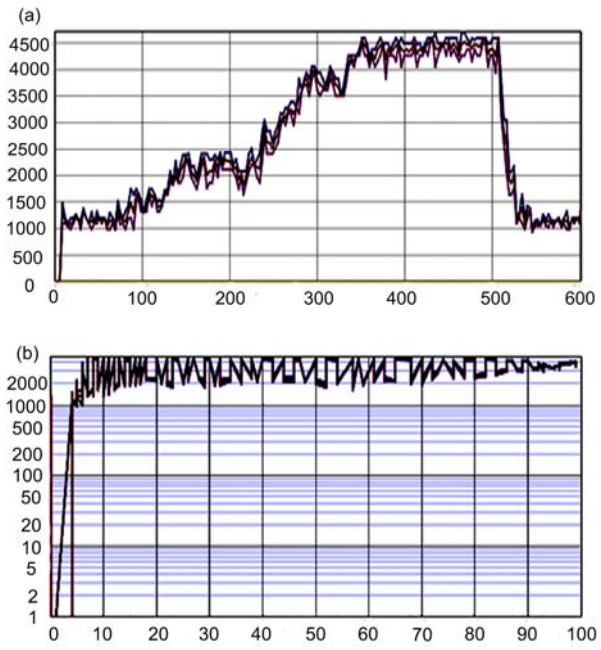

Fig.4 Web services test result (X axis: Tests Completion Time) (a) and Web services test result (X axis: Virsual Users) (b).

4.4.2 Comparison between Web services interface and JCA

In order to compare Web services interface and JCA performance, we choose the same software and hardware platforms, and use two PCs, PC A and PC B, to establish the test environment. In web services test, PC A is used to create IOC and Web services server, while PC B is used to create Web services client. In JCA test, PC A is used to create IOC and JCA client as shown in Table 3.

Table 3 Software and hardware platforms

	PC A	PC B
Hardware configuration	P4 2.4GHz, 512MB DDR	P4 1.4GHz, 384MB SDR
OS	Windows XP Professional	Windows XP Professional
Software development platform	NetBeans 5.5	NetBeans 5.5
Web services test	IOC, Web services server	Web services client
JCA test	IOC, JCA client	

In the comparison, NetBeans5.5 is used to develop Web services client and JCA client. Both read 100 PV through their corresponding interfaces, and calculate the average time of reading one PV (Table 4).

Table 4 Test results

	Average time
Web services interface	0.10867s
JCA	0.06303s

5 Conclusion

With the extensive application of EPICS, the corresponding high-level application software becomes more and more, resulting in a lot of different CA interfaces for various platforms and development languages, such as ActiveX, Matlab, PHP, and other specially designed CA interfaces. All these CA

interfaces have a certain limitations that can only be used in one platform and language. But the emergence of Web services provides us with a cross-platform and cross-language method. The CA interfaces based on Web services will enable high-level software developers to ignore the bottom details of protocols and choose any platform or development language. This increases software portability and reusability. Besides, the changes of bottom protocols or API will not affect the upper application software.

In this paper, the design of interfaces based on Web services provides another common channel access approach to high-level software developers, and helps developers in different development environment to design EPICS high-level application.

References

1 EPICS home page: http://www.aps.anl.gov/epics.

2 Hill J O. EPICS R3.14 channel access reference manual. http://www.apl.anl.gov/epics/base/R3-14/6-docs/CAref.html.

3 Evans K. Introduction to channel access clients. http://aps.anl.gov/aod/bcda/epicsgettingstarted/developtools/introductionchannelaccess.html.

4 Java channel access home page: http://jca.cosylab.com.

5 Furukawa K, Satoh M. Mejuev I, *et al.* A java-based EPICS archive viewer with SOAP interface for data retrieval. ICALEPCS'03, Gyeongju, Korea, 2003.

6 NetBeans home page: http://ww.netbeans.org/kb.

7 Portmann G, Corbett J, Terebilo A. Middle layer software manual for accelerator physics. LBNL internal report, LSAP-302, 2005.