Available online at www.sciencedirect.com
ScienceDirect

Nuclear Science and Techniques, Vol.18, No.4 (2007) 193–197

NUCLEAR SCIENCE AND TECHNIQUES

Web Services interface of SSRF archive data analysis system

LI Lin SHEN Liren* ZHU Qing WAN Tianmin

(Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China)

Abstract Accelerator database stores various static parameters and real-time data of accelerator. SSRF (Shanghai Synchrotron Radiation Facility) adopts relational database to save the data. We developed a data retrieval system based on XML Web Services for accessing the archive data. It includes a bottom layer interface and an interface applicable for accelerator physics. Client samples exemplifying how to consume the interface are given. The users can browse, retrieve and plot data by the client samples. Also, we give a method to test its stability. The test result and performance are described.

Key words XML, Web Service, Interface, Database, .NET, EPICS, AO

CLC numbers TL503.6, TP311.13

1 Introduction

Since the mid 90's, Experimental Physics and Industrial Control System (EPICS) [1] has been the most popular tool-kit for accelerator control systems. The control system of Shanghai Synchrotron Radiation Facility (SSRF) [2] saves data in relational database via EPICS Channel Access (CA) [1]. Relational database is suitable for distributed network computing. However, applying it to a system of accelerators involves an enormous volume of data in various data types, limited granularity of data acquisition, and diverse demand types of top layer users, etc.

Web Service, as the distributed middleware technology based on Internet Web, is now widely used as programming model, and multi-layered distributed system is becoming the trend of database development. A Web Services-based archive data analysis system was designed for SSRF, with a distributed software architecture of three logical tiers: data, business object and user interface to make the platform independent, scalable and flexible. In this paper, we describe the Web Services interface for data access and exemplify how to consume the interface to meet the demands of different users for data retrieval and management.

2 Web Services overview

A Web Service provides a standard means of interaction between different software applications, running on a variety of platforms. It is defined as *a software system designed to support the machine-to-machine interaction over a network* by the W3C (World Wide Web Consortium) Working Group. It has an interface described in a machine-processable format (specifically WSDL, Web Service Description Language). Other systems interact with the Web Service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML (eXtensible Markup Language) serialization in conjunction with other Web-related standards [3]. Web Services can be used internally by a single application or exposed externally over the Internet for use by any number of applications. Because it is accessible through a standard interface, a Web Service allows homogeneous and heterogeneous systems to work together as a single web of computation, which also enables a new era of distributed application development.

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

Received date: 2007-03-14

3 System design

The database system is configured on Microsoft .NET Framework 2.0 and SQL Server 2005, with infrastructure and tools to support XML and Web Services. The system has a distributed architecture with three layers of the data, the business object, and the user interface, which improve the scalability and the reuse performances. Maintenances of the system and enhancements to the solution can be done with ease. Fig.1 illustrates the architecture of the applications.

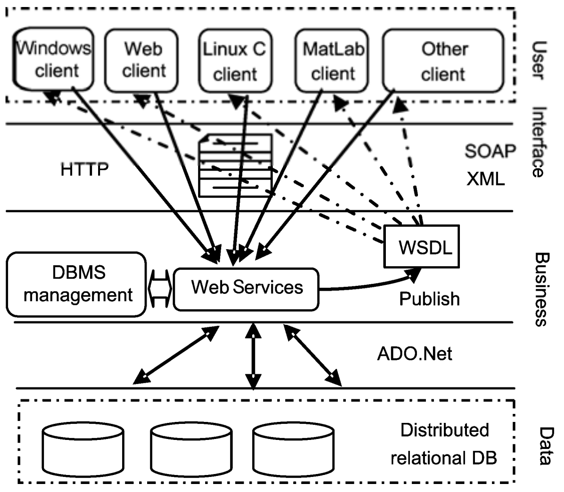


Fig.1 General architecture.

3.1 Data layer

The data layer is a database in SQL Server. SSRF's database stores all the static parameters and real-time values on IOC (Input/Output Controller) by a special archive engine via CA. The data are saved as the XML format. In this way, it is easy to extend the system, and just a modification of the XML Schema files is needed when types and quantities of data vary. Functionally, XML type can reduce the complexity of data and give facilities for data analysis.

3.2 Business layer

The business layer is a middle layer where the business logic of the solution is implemented. Meanwhile, the Web Services interface and database management is realized. The primary .NET technology involved at this layer is the ADO.NET (ActiveX Data Objects). The Web Services interface utilizes the ADO.NET as the data access component and the result is encapsulated as XML. Clients as .NET can call the interface directly; however, other clients may generate proxy classes according to WSDL files.

3.3 User interface layer

The user interface layer is the client that invokes Web Services interface exposed from business layer. This layer is responsible for interacting with the user. The clients can be a traditional Windows form, a Web forms page, the C/C++ applications under UNIX or the physical software like Matlab or Labview. For each client, we give an example to show how to consume Web Services and analyze the results. One can choose the client as he/she likes or even develop data analysis tools of his/her own. The Windows form and Web pages are based on .NET platform. Client of UNIX is proposed and implemented by gSOAP [4,5].

4 System realization

4.1 Interface design

There are two main types of interfaces, i.e. bottom layer interface and interface applicable for accelerator physics.

4.1.1 Bottom layer interface

Table 1 shows a few of the bottom layer interfaces.

Table 1 Bottom layer interface

Name	Description	Parameters	Result type
GetChannelInfo.asmx	Return information by channel name	Channelname	String
ReturnHttpVars.asmx	Return parameters of server and clients	Null	String
RetrievalData.asmx	Retrieve data by channel name and interval	Channelname starttime endtime	XML
UsrSqlQuery.asmx	Execute SQL commands by users	Usr pwd sqlstring	XML

4.1.2 Interface for accelerator physics

AO (Accelerator Object) is a part of an accelerator control middle layer, the Matlab Middle Layer, which is developed by Spear3 Laboratory together with ALS Laboratory. The AO is a kind of Matlab structure containing attributes for each Family (device list, channel names, etc.), hardware-to-physics conversion factors, range information, etc [6]. The AO resides in a hidden memory location for application data associated with the Matlab command window. Users of accelerator physics get accustomed to the AO because of its intuitive combination of physical elements. AO has been used in a number of synchrotron facilities, such as ALS, Spear3 and NSLS of USA; CLS of Canada; PLS of Korea; Diamond of England; Soleil of France; and ASP of Australia, plus a few more facilities that are planning to use AO.

What makes the AO so appealing for us to build an AO table is that the structure of the AO accords with XML format naturally. The AO table includes three columns: ID, Family Name and Data Structure in XML format. Web Services interfaces similar to operations of the AO were designed and implemented. They are: 1) the interface to return all information of all families without parameters; 2) the interface to return values of a family by family name and interval; 3) the interface to perform operation like AO. As an example, 'ao.QD.Monitor' returns the data of monitoring a dipole magnet. Therefore, physics users who are familiar with the AO can obtain data conveniently.

4.2 Client samples

Clients, through a Web browser or stand-alone application, get access to the Web Services and send out their request to the Web Services server. The server validates the request and processes it, interacts with the database and returns the results in XML format. For data users of SSRF, an analysis and plotting tool for historical data, called 'HistoryPlot', will be the most widely used. We developed Windows form and Web pages as a sample version of the HistoryPlot thoroughly. Functions of the two applications are:

■ To retrieve data out of multi-conditions such as channel names, interval, group family.

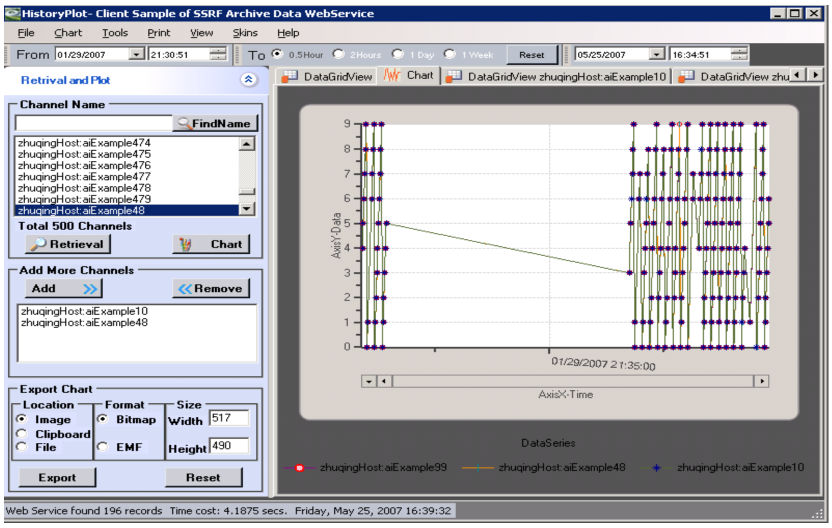
■ To list data results by sortable tables and print the tables.

■ To transform the tables to Excel or PDF format.

■ To plot data like CGIExport and WinBrowser of Channel Archiver [7].

■ To zoom the plot coordinates; save pictures in different types; print the plot; plot three-dimensional pictures; change plot type into line, bar, or step style; and save the plot data in XML format.

Fig.2 shows a typical screen of the applications.


HistoryPlot- Client Sample of SSRF Archive Data WebService

From 01/29/2007 21:30:51 To [✓ 0.5Hour] 2Hours 1 Day 1 Week Reset 05/25/2007 16:34:51

Retrival and Plot

Channel Name

zhuqingHost:aiExample474
zhuqingHost:aiExample475
zhuqingHost:aiExample476
zhuqingHost:aiExample477
zhuqingHost:aiExample478
zhuqingHost:aiExample479
zhuqingHost:aiExample48

Total 500 Channels

Add More Channels

zhuqingHost:aiExample10
zhuqingHost:aiExample48

Export Chart

Location: [✓ Image] Clipboard File

Format: [✓ Bitmap] EMF

Size: Width 517 Height 490

Web Service found 196 records Time cost: 4.1875 secs. Friday, May 25, 2007 16:39:32

Fig.2 Screen to run Windows form application.

5 Tests and results

At first, we found that the Web Services was difficult to test due to the fact that it involves distributed applications with numerous runtime behaviors. Some testing technologies such as design-by-contract were proposed. We adopt SOAPtest 3.0 [8] to test the performance of the Web Services.

SOAPtest provides an entire suite of Web service-focused testing tools, which enable us to use a consistent tool to prove the Web Services from WSDL validation to performance testing without building or updating testing clients themselves. A key feature of SOAPtest is that it provides a wide array of data sources the tool supports, and it supports test data of any ODBC/JDBC connectable database.

The test was realized on a computer of Pentium 4 processor in 2.4GHz and 512 Mb RAM, operating under Windows XP Professional, with IIS 6.0 Web server, ASP.NET 2.0.50727 and HTTP1.0. We tested a Web Service named RetrievalData in the scenario where 100 registers requested the Web Service in an hour on the Intranet, and there were 6105878 records in total in the SSRF database. According to the Bell curve, the user number increased from 5 to 100 and returned to 5 gradually. This is useful for expected usage testing, which determines whether performance problems occur with normal load patterns. We used a data table shown in Table 2 as the test data source. The result is shown in Table 3 and Fig.3.

Table 2 Data table as test data source

Parameters	Execution time	Records found	Response size
cn= x, s=11/21/200613:49:20,e=11/21/2006 14:58:30	<1ms	1374	443kB
cn=x, s=11/21/200613:50:20,e=11/21/2006 14:05:00	<1ms	875	283.5kB
cn=x, s=11/21/200613:30:20,e=11/21/2006 13:50:20	<1ms	61	21.2kB

Table 3 Testing result of Web Services

Test name	Avg. time /ms	Std. deviation /ms	Run count	Failure count	Avg. request size /byte	Avg. response size /byte
RetrievalData	6721	10166	15894	1105	684	40877

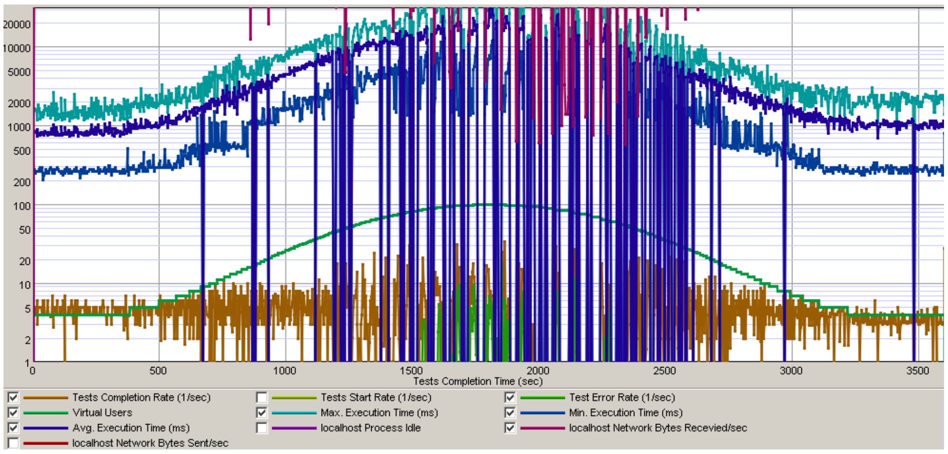


Fig.3 Testing result of Web Services.

As can be seen, the average time to retrieve and transmit data of 39.92 kilobytes from database to 100 users is 6721 ms. This proves that SOAP transmission is feasible under massive data conditions and, more important, the performance is efficient.

On the Windows client side, we tested the response time of HistoryPlot. The time was the sum of the followings: the time of clients sending their queries, the time consuming in response and querying data in the database on the server side, and the time of sending the results back and binding to the GUI. HistroyPlot was installed on a computer of Intel Xeon MP in 3.66GHz and 8GB RAM, operating under Windows Server 2003R2. The Web Service is called over a local area network. Fig.4 shows a testing result.

Actually, there will be almost 60,000 channels in SSRF and the total quantity of real-time data records amount to several millions. From Fig.4, the time to retrieve 1400 records from 5.3 million records and response to the HistroyPlot is 1.3 s. The performance of clients is sufficient to meet the needs of users.

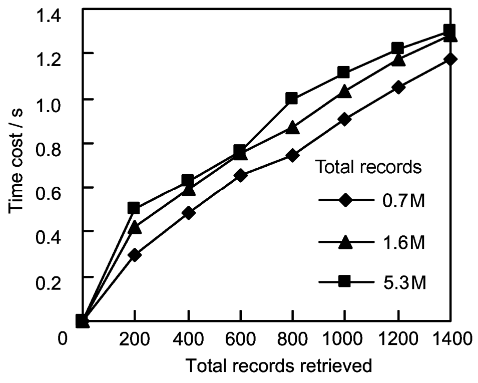


Fig.4 Windows client response time.

6 Conclusion

The archive data analysis system of SSRF was designed and implemented to provide the interface to gain and analyze archive data for users, including bottom layer interface and interface applicable for accelerator physics. A specific client application named HistoryPlot was programmed to browse, retrieve and plot data.

XML and Web Services technology provide a standard means of interoperating between different software, running on a variety of platforms, using XML as an open standard for data exchange via common Internet protocols. Applying Web Services to design the middleware interface of data access was motivated by the fact that SSRF database has an enormous volume of data in various data types, limited granularity of data acquisition, and diverse demand types of top layer users. We established a distributed application architecture of Web Services from server to clients, realized the Web Services interface for data acquisition, and gave examples of how to consume the interface. Performance of the system from the server to the client has been satisfactory.

References

1 Dalesio B. EPICS Overview, http://www.aps.anl.gov/epics/docs/training.php.

2 SSRF Introduction. http://ssrf.sinap.ac.cn/english/1/ Introduction.htm.

3 W3C. org. Web Services Architecture. http://www.w3.org/TR/ws-arch/.

4 gSOAP Toolkit. http://sourceforge.net/projects/gsoap2/.

5 http://www.cs.fsu.edu/~engelen/soap.htm.

6 Portmann G, Corbett J, Terebilo A. Middle layer software manual for accelerator physics. LBNL Internal Report, LSAP-302, 2005.

7 Graber T. Channel Archiver, http://www.aps.anl.gov/xfd/bcda/epicsgettingstarted/starttools/The_EPICS_Channel_Archiver.ppt.

8 SOAPTest. http://www.parasoft.com/jsp/products/release.jsp? articleId=1563&product=SOAP.