XML-BASED INTEGRATION FOR VMES, AN APPLICATION SERVER AND RDBMS

T. Tanabe, T. Masuoka and M. Kase, RIKEN, Saitama, Japan

Abstract

Extensible Markup Language (XML) is becoming a standard data format for computers. In our test system XML is used to set/get parameters in Common Object Request Broker Architecture (CORBA) objects running on VMEs from a DB via an application server. Interoperable Naming Service (INS) is utilized to have different ORBs communicate with each other. The description of the system and a comparison of performance with different configurations are presented.

1 INTRODUCTION

CORBA is the standard distributed object architecture for an open software bus, which allows object components created by different OSs and languages to interoperate. It would make integration with legacy controls easier, which is suited for RIBF project [1]as it is an expansion of existing facilities. XML has become a standard data format for computers, which allows one to use a standard parser (XML processor) to manipulate the data. It is now being used for the integration of various enterprise applications. In CORBA, a XML value type will also be incorporated in the near future.

We have attempted to update our CORBA based control schemes using XMLs in order to separate the data for setups, equipment properties and attributes from previously hardcoded programs. XMLs are also employed to set an interface between applications and DB. There are two ways in terms of the flow of data: from equipment to the DB, and from the DB to equipment. XML is used only for the latter case as the data collection requires much faster and larger data flow.

2 DATA COLLECTION FROM EQUIPMENT OBJECTS

In our previously constructed system, a Java based CORBA client directly receive data from C++ based CORBA servers running in VMEs. In this system, an application server (AS) is utilized to collect the data from CORBA servers and store them to a RDBMS. Figure 1 shows a schematic diagram of the system. A list of the used hardware and its operating system are listed in Table 1 and the used software and its version are shown in Table 2, respectively.

Table 1: Used hardware and operating systems.

Computer Hardware	Operating System
Host WS (SPARC Ultra10 366MHz, 512MB RAM)	Solaris2.6
Application Server (P4 1.7GHz, 512MB RAM)	Windows2000
DB Server (P3 933MHz, 512MB RAM)	Miracle Linux
VME (MVME2400, 128MB RAM)	VxWorks

Table2: Used software and its version.

Software	Version
Apache	1.3.20
Tomcat	3.2.3
Java	1.3.1
VisiBroker (VME)	4.10
ORBacus (INS, AS)	4.0.5
Oracle Workgroup Server	8.1.6
MySQL	3.23.33

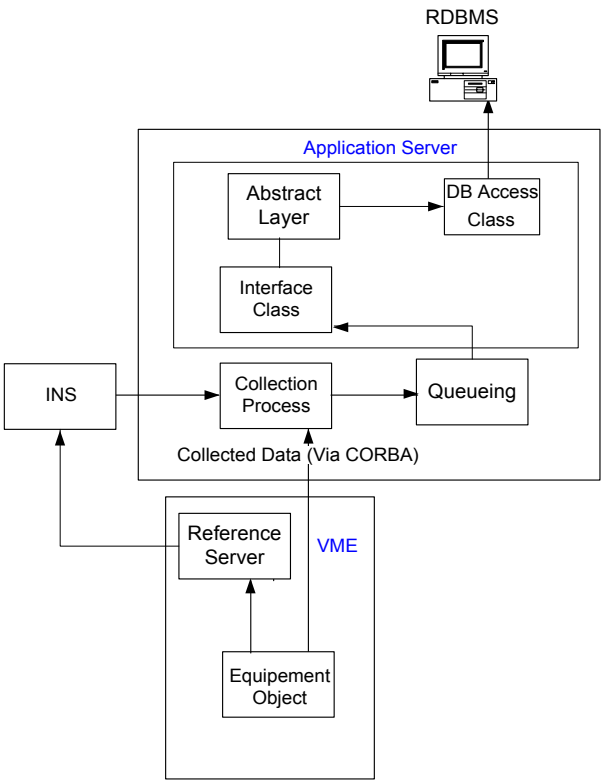

Fig. 1: Schematic diagram of the data collection scheme.

There are two types of RDBMS which can be used in this system. One is Oracle that has become a de facto standard for RDBMS, but highly expensive. The other is MySQL which is a freeware.

Orbacus [2] naming service (NS) in Java is running in the host WS and the object references for the equipment are registered by the reference server. The collection process CORBA client obtains the interoperable object reference (IOR) from the NS using the method defined in the interoperable naming service (INS) specification. The collected data are passed to the queueing process which regulates storing them to the DB via DB writing process.

Table 3: The power-supply table for RDBMS. PK stands for primary key, IN for index and EX for external key.

Column Name	SQL Column Name	SQL Definition	PK	IN	EX
Date	InDate	DATE	×	×	×
PS Name	Object Name	VARCHAR (32)	○	×	×
Current	Cur	NUMBER	×	×	×
Voltage	Vol	NUMBER	×	×	×
Status	Status	NUMBER	×	×	×

3 EQUIPMENT SETUP FROM RDBMS THROUGH XML

The former power supply (PS) control system of ours employed the codes which contains some of information for an equipment setups. Setup values were stored in a csv file and the program would know how to find the right set of data from it. Some of properties were also hardcoded so that the program itself should be modified to accommodate different type of PSs. XML files can hold both the data value and the associated characteristics. Therefore one can separate the logic part of the program and the rest which includes data value, configurations, properties and attributes. Instead of using a specially created parser for a particular code, a standard XML parser such as libxml [3] and Xerces [4] can be used. Many of them are freely available for non-commercial use.

Figure 2 shows the schematic diagram of our data DB based setup system using XML as an intermediary.

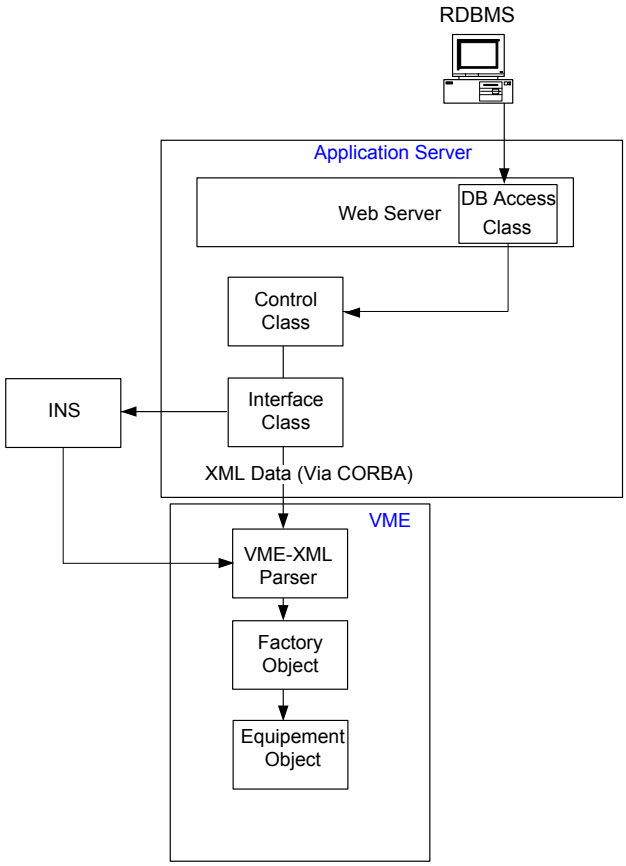

Fig2: Schematic diagram of the data setups using XML.

In this scheme, three different configurations have been

Table4: List of two tables used in the comparison.

Table Name	Field	Data Type	Size	Field Size in Byte	Comments
WorkManageList	**WorkNumber**	**VARCHAR(20)**	Variable	10～15	External Key
	RegistDate	DATE	Fixed	7	3 byte for Mysql
	StartDate	DATE	Fixed	7	3 byte for Mysql
	EndDate	DATE	Fixed	7	3 byte for Mysql
	Title	VARCHAR(80)	Variable	11～15	
	Contents	VARCHAR(1024)	Variable	11～44	
	Rate	INTEGER	Fixed	4	
WorkComments	DateNumber	INTEGER	Fixed	4	
	WorkNumber	**VARCHAR(20)**	Variable	10～15	Primary Key
	EntryDate	DATE	Fixed	7	3 byte for Mysql
	Comments	VARCHAR(1024)	Variable	10～122	
	WorkingHours	INTEGER	Fixed	4	

initially tested. The first two types are the ones using JDBC for both Oracle and MySQL. The last one is using Oracle's XDK for Java [5] tools. Table4 shows the list of tables used in the comparison test. In this comparison, a standalone java application is used to have an access to the DB. The measurement was done using two files with 10000 rows and the average was calculated for ten measurements for each configuration. The results are summarized in Table 5.

Although the Oracle's XDK tools are fastest, it is a proprietary ones and it is not certain that the same class files can be used in the future version of the software. Therefore, the JDBC solution maybe preferred in the working system unless the speed becomes the impediment. For the time being we have constructed all three types of interfaces for PS controls.

Table 5: Database access time using the test files.

	Search Time(ms)	DOM Trans. (ms)	File output (ms)	Total (ms)	Access Time (rows/s)
Oracle XDK	49	2368.3	261.5	2678.8	3733
Oracle JDBC	154.2	3063.8	484.3	3702.3	2701
MySQL JDBC	19	5339.6	467.7	5826.3	1716

For PS data, a web server with Java server page (JSP) capability is utilized to create a unified interface for data access. A JSP file uses either necessary Java files for DBs or XSQL files for Oracle to query to the DB and produce a XML file to be processed in the controller. The created XML file is passed to the VME via CORBA string. In VME C++ program using libxml can parse the XML file with SAX analysis in order to set the necessary information to the control programs.

One of the problems we encounter when different database tools are used is that the XML expression produced by a tool can be different from another from a different tool. Extensible Style Language Transformation (XSLT) must be used to bridge this gap whenever occurs. It is fairly complicated and it greatly reduces the usefulness of XML based solutions. A standard document type definition (DTD) or recently defined XML schema for accelerator components objects are to be defined to make it reasonably portable program.

4 CONCLUSIONS

XML files are used as an intermediary of sending setup data from a RDBMS to VME controllers. Using a JSP file, the difference of system types and tools can be concealed. However, XSLT must be used to create XMLs of a unified format from the output from different tools. It could become more useful if standard XML data formats are defined in accelerator community.

5 REFERENCES

[1] http://ribfweb1.riken.go.jp/
[2] http://www.orbacus.com/
[3] http://xmlsoft.org/
[4] http://www.apache.org/
[5] http://www.oracle.com/