WP2000-16

RELATIONAL DATABASE AND JAVA TECHNOLOGIES FOR NUCLEAR DATA – Preliminary Report

R. Arcilla, D. Winchell, Y. Sanborn

National Nuclear Data Center
Brookhaven National Laboratory
Upton, New York 11973-5000

August 4, 2000

Introduction

The National Nuclear Data Center (NNDC), in cooperation with Clark University, has been investigating the use of relational database and Java technologies for the administration and dissemination of nuclear data.

Commercial relational database management systems (RDBMS) using the Structured Query Language (SQL) are widely used in the business world. As a result, the technology is mature and well supported, and a wide variety of auxiliary tools are available. Because of this, an RDBMS approach to nuclear databases might afford several advantages over the software used presently. Furthermore, the automatic replication and synchronization capabilities of these systems promise to improve the efficiency and reliability of data exchange among the cooperating data centers.

This report presents the preliminary results of the evaluation of SQL-compliant relational database management systems and the Java platform. Criteria considered include speed, ease of use, cross-platform capability, support, and cost. The ability to re-use legacy programs was also investigated. Both commonly used database programs such as Microsoft Access and pure Java-based database systems were tested. The report concludes with a recommendation on the best migration strategy for existing nuclear databases. A final, detailed report will be completed in July 2000.

Methods of Evaluation

Database systems

An initial investigation of RDBMS technology was made by porting the entire Nuclear Science References (NSR) database to the Microsoft SQL-Server RDBMS. A web interface to this instance of NSR was developed and is publicly available via the NNDC homepage. Other databases, notably the NNDC Adlist database, were also ported to RDBMS systems for testing purposes.

In order to evaluate several database products using the same underlying data, a subset of NSR was used. In particular, three complete tables from the database were used. Details are given in the "Results" section and the appendices.

The following RDBMS packages were tested:

- Microsoft SQL-Server V7.0
- Sybase Adaptive Server Anywhere V7.0
- Microsoft Access 2000
- PointBase V3.1GA
- Cloudscape V3.0

The last two of these are based on the Java programming language and are platform-independent; the first three were run under the Windows NT operating system. A set of identical queries in SQL format was performed on each of the databases.

Java Optimizing Compilers

Computer industry studies show that a Java application compiled into native code will run 10 to 50 times faster than the interpreted Java byte code. At present, Compaq is developing an optimizing compiler for Java that should run under OpenVMS. This compiler will not be available until after July 2000. In order to test the concept, the compiler TowerJ from Tower Technology Corporation was used to compile parts of the Cloudscape database program on Windows NT.

Legacy Codes

There is a large body of legacy software associated with the databases maintained at the NNDC, and duplicating all of the functionality of these codes by writing new software would make the cost of migration prohibitive. However, it should be possible to re-use most legacy code, replacing only database calls. To test this, software routines were written to replace Fortran database access subroutines with calls to external Java programs. These Java programs can then access remote databases using a Java Database Connectivity (JDBC) package. The concept was tested by converting the NSR retrieval program NSRRET, running on OpenVMS, to access a remote RDBMS version of NSR running on a Windows machine. A figure illustrating this process is shown in Appendix A.

Results

Database systems

Details of the benchmark tests and results are given in Appendix B. In summary, the non-Java databases were found to be comparable in speed and size, while the Java-native software was much slower. In and of itself this result is not surprising, as Java is known to sacrifice some speed and power in order to be platform-independent. However, the degree to which the Java software is slower makes it unusable for our purposes, as it stands.

Java Optimizing Compilers

At the time of this writing, only about two-thirds of the required Java classes had been compiled using TowerJ. Because key classes were still interpreted and dynamically loaded at run time, the results for the “semi-compiled” database were no better than for the pure interpreted Java.

Legacy Codes

The converted program NSRRET was found to run somewhat more slowly than the original version, but not excessively so. Several bugs remain in the converted program, but it has been demonstrated that the underlying technology is a viable method for converting legacy software.

Conclusion and Recommendations (preliminary)

Before the completion of the final report in July, we will continue to test evaluation copies of the database software, in preparation for making a final purchasing recommendation. We will test the replication and synchronization technology in one or more of the database programs. Compilation of the Cloudscape database will be completed, and tested against an interpreted version on the same machine.

RDBMS and Java technologies can help the nuclear data community in both the efficiency of database administration and the flexibility of dissemination. Our preliminary recommendation is to adopt one of the commercial database products as a standard and begin converting the nuclear databases and legacy programs. The databases will reside on Windows NT or Linux machines, and will be accessed from legacy programs on OpenVMS via JDBC. We also recommend continued investigation of Java-native database software used in conjunction with optimizing compilers, in order to take advantage of Java’s platform independence.

Appendix A

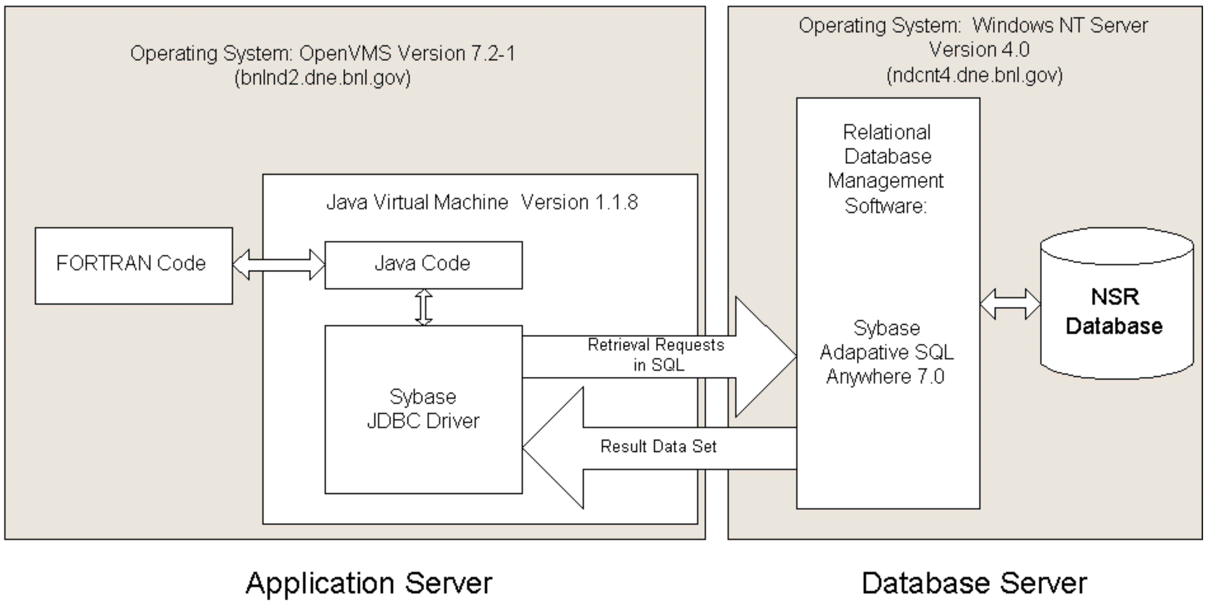

Fig. 1 Linking FORTRAN legacy code with Java code to access a remote database

Appendix B

Software	OS	host	db size (MB)	log size (MB)	load time (approx)	Qry1 (sec)	Qry2(sec)	Qry3(sec)	Qry4(sec)
Pointbase	Linux	ndcnt2	144	550	3 days	32	26	176	N/A (c)
Pointbase	OpenVMS	bnlnd2	144	420	36 hours	21	17	126	N/A (c)
Cloudscape - interpreted	OpenVMS	bnlnd2	180	N/A (a)	4 hours	6	3	5	285
Cloudscape - semicompiled	Win NT	ndcnt4	180	N/A (a)	N/A (b)	7	2	6	356
MS SQL Server	Win NT	ndcnt3	74	354	5 minutes	1	3	1	15
MS Access	Win NT	ndcnt3	90	N/A (a)	10 minutes	1	2	1	8
Sybase ASA	Win NT	ndcnt4	89	84	10 minutes	1	1	1	5

notes:

three tables from NSR: (same indices defined in each database)

ref_tbl	157533 lines
auth_tbl	603235 lines
auth_dic	56652 lines

hosts:		
	ndcnt2	200 MHz pentium
	bnlnd2	533 MHz Alpha
	ndcnt3	450 MHz pentium
	ndcnt4	dual 600 MHz pentium

```
Qry1: SELECT keyno,type,info
                  FROM ref_tbl
                  WHERE pubyear=1998 and coden='JPGPE'; (116 lines out)
Qry2: SELECT auth_tbl.keyno,auth_tbl.ord
                  FROM auth_tbl JOIN auth_dic ON auth_tbl.akey=auth_dic.akey
                  WHERE auth_dic.aname='SMITH' and auth_dic.ainit='A'; (306 lines out)
Qry3: SELECT ref_tbl.keyno
                  FROM ref_tbl JOIN auth_tbl ON ref_tbl.keyno=auth_tbl.keyno
                  WHERE auth_tbl.akey=44916 and ref_tbl.pubyear=1997; (12 lines out)
Qry4: SELECT ref_tbl.keyno,ref_tbl.type,ref_tbl.coden,ref_tbl.info
                  FROM ref_tbl JOIN (auth_dic JOIN auth_tbl ON auth_dic.akey=auth_tbl.akey) ON ref_tbl.keyno=auth_tbl.keyno
                  WHERE auth_dic.aname='SMITH' and ref_tbl.pubyear=1992; (37 lines out)
```

(a) no log file
(b) copied directly from VMS
(c) unable to perform query