International Journal of Computer Science
and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 2, Issue 5, Nov 2013, 61-64

International Academy of Science,
Engineering and Technology
Connecting Researchers; Nurturing Innovations

CALLING COBOL APPLICATIONS FROM DISTRIBUTED CLIENTS USING JAVA

TANUL BHASIN[1] & SAGAR GUPTA[2]

[1]Systems Engineer, Infosys Limited, Jaipur, Rajasthan, India

[2]Systems Engineer, Infosys Limited, Bangalore, Karnataka, India

ABSTRACT

In the fast improving technological world, businesses still use Legacy applications as database management systems working on their mainframes and minicomputers. The reason could be influenced by a variety of factors other than functionality, such as economic reasons – the time of return on investment is too huge, vendor lock-in or the inherent challenges of change management. The need to modernize legacy applications could be achieved with one of the methodologies using JAVA direct connectivity to COBOL.

KEYWORDS: JAVA, COBOL, Distributed Connectivity, WAS (Web-Sphere Application Server)

INTRODUCTION

Clients using legacy applications find it difficult to modify the entire application when distributed access is required for a single module, as a single change to the module would sequentially require changes in the work flow of other modules dependent on this module. The execution of the one module without changing the workflow of dependent modules in the same application requires the direct calling of it. This can be done in two ways:

- Creating JAVA wrapper layer to accept the distributed request which can internally call the COBOL application for processing and sends back the response the calling application.
- Exposing target module as web-service, but to use this; a CICS (Customer Information Control System) layer needs to be designed which requires several changes which also requires changes the work flow of other modules dependent on the this module either for response or for request.

The direct connectivity of between COBOL program and distributed client can be established easily using JAVA application deployed on Web-server (in discussed case it is Web-sphere Application Server) which act as receiving stub for all client calls and internally calls the COBOL program as a sub-module using a **glue-layer** (a C program – exposed as native function to JAVA), which initiates and ensures the data passing/data conversion between two different environments i.e. COBOL and JAVA, the interface glue layer is called **JAVA Interface Navigation Layer**.

Web services / HTTP web application can also be used to connect application with mainframe but to accomplish this; several changes would be needed on legacy application which will further affect the work flow of other applications. Moreover by introducing web-services CICS will come into the picture which will increase the total throughput of the whole application and the performance of the system will depends majorly on time taken by CICS to process the request, moreover CICS can only be used as a synchronous behaviour which will keep the main application in a waiting state till CICS return back the response. To prevent such unwanted changes, and issues with respect to performance Java connectivity can bring robustness and flexibility in the system such that the system can be maintained and can be scaled up or down without disturbing the other tiers involved in the business model.

MATERIALS AND METHOD

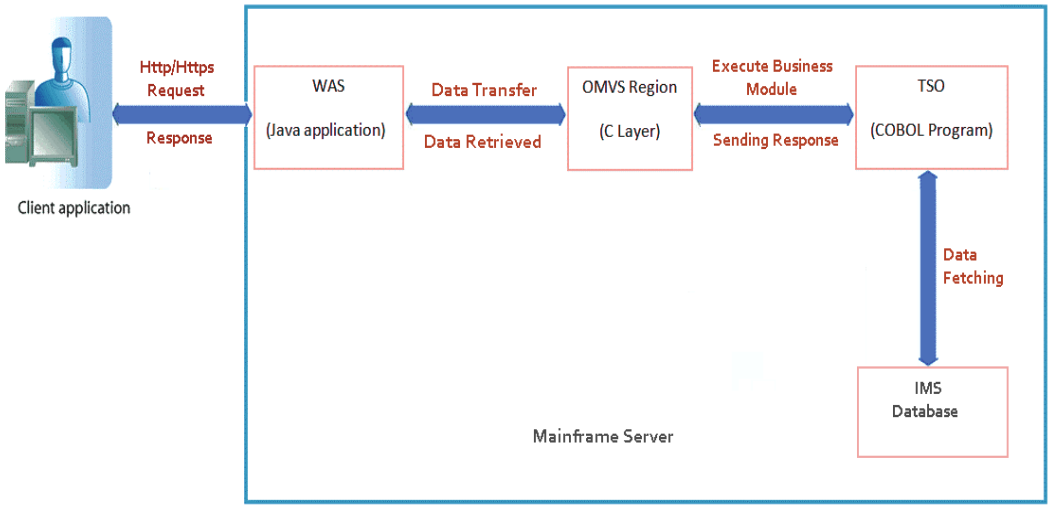

Figure 1: Client-Server Communication

Client Application: The client sends input to JAVA program, which gets its output from the mainframe server after processing the request.

WAS (Web-Sphere Application Server): WAS (Web-Sphere application Server**)** works as a web application server on which Java based web applications are deployed. WAS provides an environment for the installation of server-side applications and is designed to help enterprises deploy and manage Web-based applications of varying degrees of size and complexity.

Java Application: A java application receives the input and calls the C program exposed as native function machine level language C, the JAVA program takes cares of converting the input data to ASCII format so that data can be sent to C application and converts the response to the required format.

C Application: The C layer has a native support to invoke COBOL applications as language environment and inter language communication mechanisms are available for C and COBOL. The C application receives inputs from the JAVA application will invoke COBOL application for further processing.

COBOL Application: As the name suggests, the application is programmed in COBOL and is deployed in the mainframes. This application takes the data from interface layer and processes it. The application calls backend database IMS (Information Management System) v10 which helps in organizing the business data with both device and program independence. It is difficult for a JAVA application to directly call the backend database; hence a COBOL application is used. The COBOL interface to IMS database segments is by issuing DL/I calls. A DL/I call is a request made by the interface to DL/I to access one or more segments of a database. The call contain information about the function specifying the action DL/I is to perform. The PCB (PSB) to be used to perform the function, the IO area information and the segment search arguments (SSAs) which is a set of formatted search.

IMS Database (Any Version Prior to v11): IMS DB is a Hierarchical database which helps business organizations to organize in structured way and provides both program and device independence. IMS database stores data in tree format, where every level except root will have a parent level which represents data at higher level of hierarchy. The advantage of IMS database is its data storage and data retrieval functionalities and in IMS database user cannot create their personal copies of data which helps the organisation in terms of data security.

- **Data Integrity:** The data will remain in the database even when IMS DB is not running and also is guaranteed to be consistent
- Queries can be performed against data in the database
- All database transactions will be performed as a single unit of work
- Multiple database transactions can occur concurrently and the results of each transaction is isolated
- Tuning can be performed on the databases for reorganizing and restructuring them

DISCUSSIONS

To established the connectivity between client application and COBOL application some settings are required on Mainframes, for this discussion below mentioned hardware/software assets were used to create test set-up.

- Mainframe
- JDK 1.5 or above in OMVS region
- **OMVS Region:** The OMVS region should be installed and running and user should have permission on it.
- WAS (Web-sphere Application Server)
- C89 compiler for C/C++

Once the set-up is created the JAVA application can be exposed to the distributed environment as the web-app which will take care of client connectivity and COBOL application can receive inputs from JAVA application and process the request in regular manner.

To call COBOL application from JAVA, the COBOL load module should be present in a library which must have suffix like *.c, .OBJ, .hh, .I .C, .i* etc. and after setting up of environment, client application can call JAVA application running on web-server, to establish connectivity between all the modules following compilation/execution level practices needs to be followed for respective module:

- The COBOL program should be compiled by mentioning the following parameter list in the 'LINKEDIT' step of the JCL with PARAM='LIST, XREF, RMODE (ANY), RENT'
- After compiling the JAVA program using 'javac' command issue 'javah' command with the class name specified in the JAVA program, to generate the header file for the C program
- To create a dynamic link (dll) between C program and COBOL program c89 compiler command needs to be used with -c –o option.

BENEFITS

- The COBOL application in TSO region can be invoked from a JAVA program running in OMVS region a mainframe without disturbing the workflow of the application.
- This calling can be extended to distributed environment also by connecting your OMVS region to distributed environment.
- Legacy applications which needs to be connected to distributed environment can easily be connected.

- Wrapper programs can be created for IMS applications such that the data access can be possible from the distributed environment.
- Less administrative and development effort is required as in case of web-services CICS needs to be involved.
- CICS web-services can support only single operation call but by using direct connectivity approach more than one modules of the same application can be exposed for distributed access.
- This approach creates tier based architecture which can be enhanced or managed on an individual level.

CONCLUSIONS

The establishment of a direct connectivity between COBOL application and distributed client can expose legacy application to distributed clients which can be used to modernize these applications easily. It also gives better efficiency, saves time and its scalability is improved as the client can make changes in the legacy application without making any changes in workflow of the whole applications and gets the desired results. The tier level architecture introduce more flexibility as the dependency on the web-service definitions is no longer required, which give independency to server level application to update the business flow without informing the client applications. Moreover in CICS web-services the data passage is also limited in terms of size and the data-types used to pass data, but in direct connectivity the dependency on data-type for input values is also removed which gives more robustness to the application.

REFERENCES

1. Redbook: Java Stand-alone Applications on z/OS Volume II.
2. Redbook: SMP/E for z/OS V3R5.0 User's Guide z/OS V1R10.0-V1R11.0 SA22-7773-14
3. Redbook: Java Stand-alone Applications on z/OS Volume II (sg247291).