Astronomical Data Analysis Software and Systems VII
ASP Conference Series, Vol. 145, 1998
R. Albrecht, R. N. Hook and H. A. Bushouse, eds.

Search and Retrieval of the AXAF Data Archive on the Web using Java

Sumitra Chary and Panagoula Zografou

Smithsonian Astrophysical Observatory, AXAF Science Center, Boston, MA 02138

Abstract. An important component of the AXAF Data Archive is the interface through the WWW. Internet applications are moving beyond delivering static content to more complex dynamic content and transaction processing. While HTML and CGI based interfaces have been possible for some time, it was not until the advent of JDBC (Java[1] Database Connectivity) that a uniform and powerful interface to the different types of servers in the archive, with secure access to proprietary data, could become a reality.
This paper presents a multi-tier architecture which integrates the data servers, Web server and a HTTP/JDBC gateway and enables data on any of the data servers in the AXAF archive to be available to clients using a Web browser. This Web-based solution to data browsing is currently available for the contents of the AXAF observing catalog and other catalogs in the archive. A Java API for other database applications to access the data is presented. Performance issues and security limitations and workarounds are discussed.

1. Introduction

The AXAF archive architecture involves a number of servers, RDBMS servers for managing relational data and archive servers that manage datafiles. Access to datafiles is provided by existing client applications which send data ingest, search and retrieval requests in a 4gl language to an archive server (Zografou et al. 1997). An architecture was developed to build a powerful and uniform interface to the various servers through the World Wide Web. The advent of JDBC a specification for an API that allows Java applications to access various DBMS using SQL became the obvious choice to build this interface.

2. System Architecture

Figure 1 shows both the client-server architecture developed using the Database vendor API (Zografou et al. 1997) where both the client and the data servers communicate using the same protocol and the 3-tier architecture which allows

[1]Trademark of Sun Microsystems

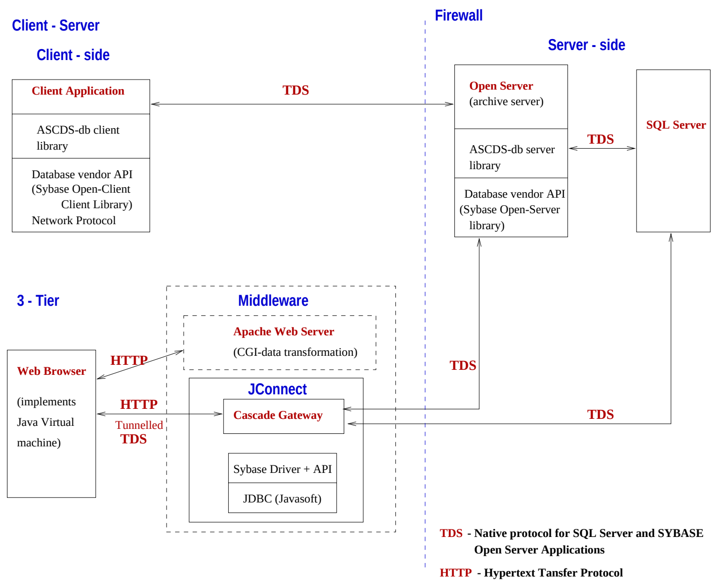
Client - Server
Client - side
Client Application
ASCDS-db client library
Database vendor API (Sybase Open-Client Client Library) Network Protocol
TDS
Firewall
Server - side
Open Server (archive server)
ASCDS-db server library
Database vendor API (Sybase Open-Server library)
TDS
SQL Server
3 - Tier
Middleware
Apache Web Server
(CGI-data transformation)
Web Browser
(implements Java Virtual machine)
HTTP
HTTP
Tunnelled TDS
JConnect
Cascade Gateway
Sybase Driver + API
JDBC (Javasoft)
TDS
TDS
TDS - Native protocol for SQL Server and SYBASE Open Server Applications
HTTP - Hypertext Tansfer Protocol

Figure 1. System Architecture [Client-Server and 3-Tier]

a Java client applet to communicate with the data servers behind a firewall. The middle tier consists of Sybase jConnect[2] for JDBC package. This includes the Cascade HTTP Gateway[3] which is the gateway to the RDBMS and archive servers.

According to Java security restrictions, applets can only connect back to the host they were downloaded from. If the Web server and database server were on the same machine then the Cascade gateway is not required. But in this case the data server hosts are behind a firewall hence the Cascade gateway acts as a proxy and provides the path to the data servers. The HTML documents and Java code are served from the Apache Web Server[4] tree. The Apache Server is also used to execute a cgi-script which performs transformations of input and output data when required. Hence the Apache and the Cascade run on the same host on different ports so that the applet downloaded from the Apache server tree can connect back to the same host at the port of the Cascade gateway.

3. Search Operation

Searching involves querying databases on the SQL Server and viewing the results on the client screen or saving them in a file. Both command-line client applications and Java applets have been written to send browse requests to the archive and the database servers. The primary difference between the above

[2]http://www.sybase.com/

[3]http://www.cascade.org.uk/

[4]http://www.apache.org/

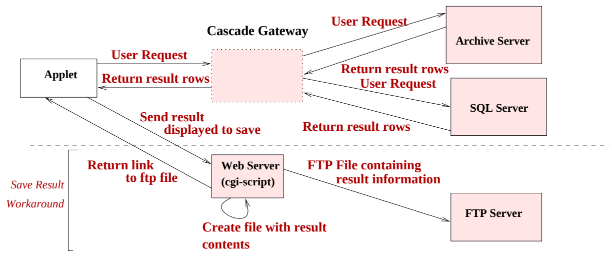

Figure 2. Search Element

two types of client requests is that the latter's request goes through the Cascade HTTP/gateway and then to the data server (Figure 1). The way the browse request is handled on the server-side is the same for either client. The results are received in the same way on the client-side and displayed. One major difference is in saving the results obtained.

Due to Java applet security restrictions, files cannot be created on the client filesystem. Hence the result contents are sent by the applet to a cgi-script on the Web Server host which creates a file on its filesystem from the binary stream. This file is then FTP'ed to the FTP server in a previously created location. The cgi-script informs the client of this path which is displayed to the user.
In the future with JDK1.1 and a compatible Web browser it may be possible to re-create files on the client side depending on authentication and access control schemes. This would eliminate the extra step of passing data to a cgi-script.

4. Retrieve Operation

This involves retrieving files stored in an archive from the archive server via metadata browsing in the SQL server. As shown in Figure 3 the ideal retrieve scenario would function the same way for either client type. The client would receive the filenames and file contents as a binary stream, re-create the files on the client filesystem. Since this is not possible with applets due to security restrictions, two methods are presented to resolve this issue.

1. The file contents are forwarded to the middle-tier and the files are re-created on the Web server's filesystem. These files are then FTP'ed to the FTP Server and the client is notified of the link to access the files. This is similar to the "save results" method outlined above.

2. The second method is much more efficient both from the performance point of view and because it avoids of excessive network activity. The applet in this case only serves as the GUI and passes the user input to the Web server. The Web server now behaves as the client to the archive server. The CGI-script directly executes the "arc4gl" client application which is the command-line client application. Hence it has the ability to send and receive data from the archive server. The files retrieved are re-created on the Web server filesystem. Then they are FTP'ed over to the FTP server. The link to the files is returned and displayed at the client applet end. If the files are of a format supported for viewing in the Web Browser e.g.,

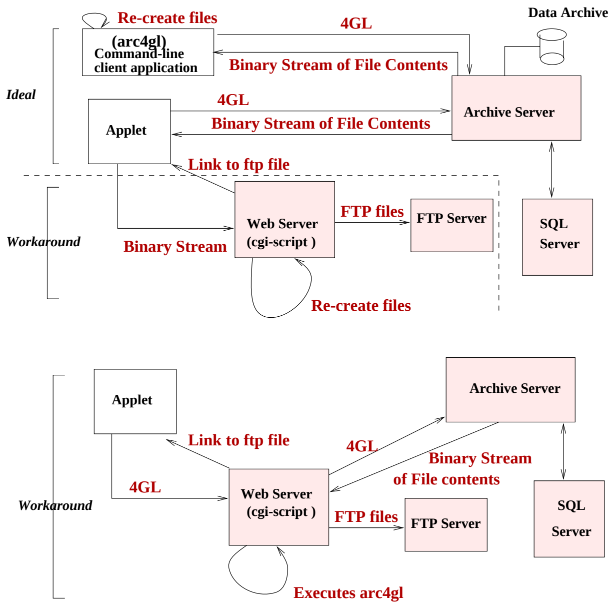

Figure 3. Retrieve Element

HTML, GIF, PS then the files, if placed under the Web Server tree, can be directly viewed by the user.

To port this setup to various environments such as test and production, a cgi-script is used to dynamically create the HTML file presented below with the appropriate information.

```
<HTML>
<applet
       code = "ocat.class"
       codebase = "base URL of applet to be displayed"
       archive = ocat.zip
       width=700 height = 650>
       <param name=proxy value="IP address:port of Cascade Gateway">
       <param name=host value="IP address of DataServer">
       <param name=port value="Port of DataServer">
       <param name=uid value=""> <param name=pass value="">
       <param name=cgiserver value="IP address of Web Server">
       <param name=cgiserverport value="Port of Web Server">
</applet>
</HTML>
```

References

Zografou P., Chary, S., DuPrie, K., Harbo, P., & Pak K. 1998, “The ASC Data Archive for AXAF Ground Calibration”, this volume