Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

Using a Java Web-based Graphical User Interface to access the SOHO Data Archive

I. Scholl, Y. Girard, A. Bykowski

Institut d'Astrophysique Spatiale, CNRS/Université Paris XI, Bâtiment 121, F-91405 ORSAY, France, Email: scholl@medoc-ias.u-psud.fr

Abstract. This paper presents the architecture of a Java web-based graphical interface dedicated to the access of the SOHO Data archive. This application allows local and remote users to search in the SOHO data catalog and retrieve the SOHO data files from the archive. It has been developed at MEDOC (Multi-Experiment Data and Operations Centre), located at the Institut d'Astrophysique Spatiale (Orsay, France), which is one of the European Archives for the SOHO data. This development is part of a joint effort between ESA, NASA and IAS in order to implement long term archive systems for the SOHO data.

The software architecture is built as a client-server application using Java language and SQL above a set of components such as an HTTP server, a JDBC gateway, a RDBMS server, a data server and a Web browser. Since HTML pages and CGI scripts are not powerful enough to allow user interaction during a multi-instrument catalog search, this type of requirement enforces the choice of Java as the main language. We also discuss performance issues, security problems and portability on different Web browsers and operating systems.

1. Introduction

SOHO is a satellite dedicated to the study of the Sun, built as part of an ESA and NASA cooperation. It operates with 12 instruments on-board that produce about 1.5 TB for 2 years of mission. These are operating in the frame of joint observing programs but their data and catalogs are not homogeneous. SOHO scientific and engineering data are currently stored at NASA/GSFC and at MEDOC/IAS (Scholl 1996). The goal of the current joint effort between ESA/NASA and MEDOC teams is to define a unique catalog for all data as well as a unique user interface to access them independently of their physical location. After a common catalog definition (Scholl & Sanchez 1998), the software development was split between the two teams. ESA/NASA team was in charge of the catalog data ingestion programs while, at MEDOC, we designed and developed the software that is responsible for catalog and file access. Because of the international character of this mission, our basic requirement was to develop a software that is independent of the archive architecture and which provides users with web access capabilities. Moreover, the amount of data and their complexity can lead to heavy queries producing a large volume of results.

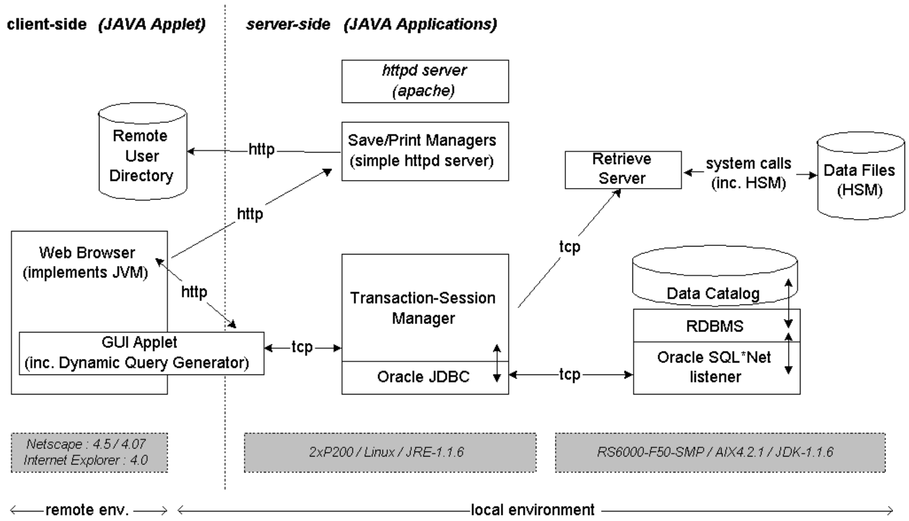

Figure 1. System Architecture.

Then, ergonomy, good response time and easy browsing functions were among our main goals.

2. System Architecture

The catalog access system is pure Java software[1]. It is organized as a 3-tier architecture (see Fig. 1). The client side is a Java applet while the server side is composed of several stand-alone Java applications.

The applet is stored on the HTTP server directory tree to be loadable by the user's Web browser, that must be JDK-1.1 compliant. The client implements the graphical user interface (GUI) functions such as:

1. Query creation and submit,
2. Result display,
3. File retrieval submit.

The main application, the transaction and session manager, is a multithreaded Java server located between the client and the RDBMS. Its roles are:

1. Transaction and session management: the server keeps track of each connected client.
2. Security management: it operates with a dedicated, security enhanced protocol in order to protect the content of the RDBMS from abusive and malicious use.
3. Query management: the server receives the content of the user's query and forwards it to the RDBMS.
4. Result management: in order to speed up result displays, the query result is stored on the server that only sends to the client the number of rows it needs for one display.

---

[1] http://www.medoc-ias.u-psud.fr/new_archive/

The client, the server and the other applications (listed above) communicate using TCP/IP sockets. Due to Java security restriction, the GUI code must be located on the host where the transaction server runs, and therefore both must be located on the HTTP server. Additional components may be run either on this machine or on a separate one. They are the following:

1. The RDBMS: accessed using an ORACLE JDBC driver,
2. The 'retrieve server': dedicated to file retrieval and packaging procedures,
3. The 'save server' is a simple HTTP server used to save retrieved files on the client host,
4. The 'print server' is another simple HTTP server used to allow the user to save/print the result of a query.

The transaction server provides other tiers with a Java API that implements its protocol.

3. Catalog Access

This part of the software involves only the GUI applet, the transaction manager and the RDBMS. The GUI allows the user to compose a query to be sent to the catalog. The first time the user loads the home page of this program, the client establishes a connection with the server. A timeout mechanism keeps this session alive during a pre-defined time of inactivity. Among others, that makes possible to get back the resources the user has been allocated.

The query is built dynamically using a dictionary which is the only part of the software that would be changed if the database structure changes. Each selection enriches the query in order to restrict the set of data (rows) returned.

Once the user has committed his choices, the resulting query is sent to the server which completes it in order to figure out if the data may be accessed by the user. The query is forwarded to the RDBMS via a JDBC driver; the results, received via the same channel, are temporarily stored on the server. Then, the content of the first page can be sent to the client before the end of the transfer. These buffers have several goals: lighten the client, decrease response time, allow the user to browse data by using next/previous (for the same view) and back (for different views) functions without accessing the database each time. Several parameters such as buffer size, number of maximum back operations, etc. can be tuned if necessary regarding the host computer capabilities.

4. File Retrieval

The retrieve operation involves several components at different levels. Figure 2 shows them as well as the sequence of operations necessary for the file retrieval process. The main problem encountered for this part of the software is related to Java security restrictions that don't allow an applet to write on a file system. In order to make the browser to allow this operation, the saving must be done outside the JVM. For user convenience and general security reasons we use an HTTP download instead of collecting login and password from users in order to send files using FTP. The request to the corresponding HTTP server, the save server, is an URL hiding a certificate coming from the transaction server. This

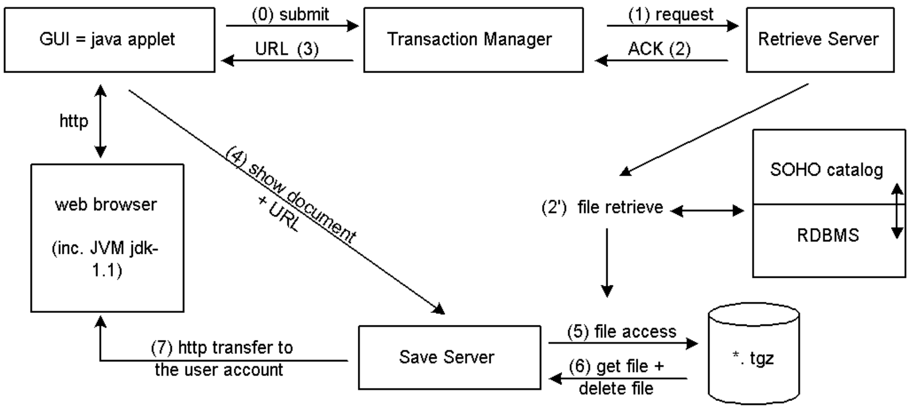

Figure 2. File Retrieval Data Flow.

ensures that the save server will not deliver the precious data to inadequate persons.

The retrieve process is independent of any kind of storage system: for example, at MEDOC we have implemented an HSM architecture with disks and cartridges while the GSFC has on-line disks.

5. Conclusion

In this paper we have presented the first version of a catalog and file web-based access tool. It has been a 10 man-month development. The main problems encountered were due to the Java language instability and the lack of portability of applets on different web browsers and platforms.

Our current work aims to get better performances in tuning the database access and result management. Our future works will be oriented to new facilities enhancing current search criteria as well as adding different parameters (like other ground-based or space solar observatories keywords). We will also follow Java evolutions such as JDK-1.2 with JFC which comes with more powerful components that improve performances and should simplify our code (i.e. RMI communications, direct print capabilities, light weight graphical components).

Acknowledgments. We are grateful to all future users who have tested this software. Special thanks go to Luis Sanchez and George Dimitoglou from ESA-NASA team and Jean-Luc Orcesi from MEDOC team for their technical contributions.

References

Scholl, I. 1996, in SpaceOps'96 (ESA Publ. SP-394) (Munich: ESA), 384

Scholl, I. & Sanchez L. 1998, "User Requirement Document for the SOHO Archive", `ftp://ftp.medoc-ias.u-psud.fr/pub/archive/urd.ps`