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

Distributed Searching of Astronomical Databases with Pizazz

K. Gamiel, R. McGrath and R. Plante

National Center for Supercomputing Applications, University of Illinois, Urbana, IL 61801

Abstract. Robust, distributed searching of networked-based astronomical databases requires an investment not only by individual data providers, but also by the astronomical community as a whole. We describe both these investments and introduce a supporting software project called Pizazz. It is our hope that the community will appreciate the social requirement placed on them to work together and to participate in efforts towards a globally integrated, distributed astronomical search and retrieval system.

1. Background

Establishing the infrastructure required for robust, distributed searching of databases is not a simple proposition. In our view, there are four major steps involved; analysis of existing data sources, choosing a model, profiling, and implementation.

In this case, analysis means finding various astronomical database sources around the net of interest to the community and analyzing the interactive capabilities and content. For example, what is the protocol used, what is the query format, what is the response format, etc.

Next, we must choose a model for interacting with the identified data sources in parallel. We narrow these models to two. The first model is the "hands-off" model, that is, we do not impose any changes on the original data sources whatsoever. In such a model, one would build a distributed searching gateway that would offer a single user interface, but would map the user's query to each and every original data source in its native protocol and interaction style. While this is obviously an appealing option for the original data providers, we find that the wide range of protocols and interactive paradigms that must be supported severely limit the overall usage and value of response. We feel this approach is short-sighted and ultimately of little value. The second and preferred model is to agree on a single protocol for information retrieval. By agreeing on a single protocol, we ensure a consistent and highly extensible interface to each and every original data provider. If we agree, for example, on a single protocol and profile, one can easily imagine autonomous user agents interacting with the system on behalf of users.

Profiling is a step that must be taken regardless of which model is chosen. Profiling, in the current context, means agreeing on general features of a distributed searching system. For example, a profile for such a system might state that all original data providers must provide access to title, author, and abstract fields. It may state that a response must include the same information and maybe a URL. In other words, a profile is a document that ensures some level of consistency when interacting across data sources.

The last step is implementation. This involves work depending on which model is chosen. If the "hands-off" model is chosen, one must implement the distributed searching gateway and for each original data source, a driver must be written in order to interoperate with that source, mapping to the user interface. In this model, the programmer must be available at a moment's notice to alter any of the possibly hundreds of drivers when the original data provider alters their interface. Since no open standards are imposed, it is certain that frequent changes will occur, particularl since most of these systems will probably be HTML/CGI-based systems and will change for purely superficial reasons. If the second model is chosen, a communications server must be written and installed at the original data provider's site. A gateway application that need only speak a single protocol in parallel is then written. In this model, the gateway need only point to new data providers as they come online, not requiring any changing of source code.

2. Pizazz

We chose a model based on a single information retrieval protocol, namely ANSI/NISO Z39.50. Z39.50 is a well-defined international standard used in academic, government, and commercial institutions. It defines a standard for interactive search and retrieval of database records from a data source over the Internet. We here announce a software distribution called Pizazz that includes a Z39.50 server toolkit. More information on Pizazz and other project details can be found at http://webstar.ncsa.uiuc.edu/Project30/.

The server toolkit included with the Pizazz distribution builds on an existing application called pizazzd. The typical scenario is that a data provider who wants to participate in the distributed searching system will download Pizazz and build the default pizazzd server. The provider will then alter a single C file that includes callbacks for the four major information retrieval functions, namely initialize, search, present, and close. In those four functions, the provider will make calls to his own native database system, building the response as appropriate with pizazz library calls we provide. Once that interface is written, the provider then notifies NCSA and a pointer to their server is added to the distributed searching gateway.

It would be a disservice to say that creating this interface is simple. However, we are committed to making it as easy as possible with helper tools and, more importantly, feedback from users ("user" in this sense is the programmer who builds the interface between pizazzd and the native database). By far the most difficult part of implementing the interface is in support of the nested RPN query structure. The user has access to function calls for walking the query tree structure and while doing so, must generate a query based on terms and attributes suitable for his native system.

The information retrieval model used by pizazzd is as such. The server receives a single initialize request from the client. The initialize callback is invoked where any native database initialization is performed. Typically, the server then receives a search request which includes the database name to search and a potentially complex, but standard, RPN query, along with other search parameters. The search callback is invoked where the query is translated and passed to the native database system. The search results in the formation of a logical result set. From that result set, the client may then request to have records presented. In that case, the server receives a present request asking, for example, records 1 through 5 from the Default result set in HTML format. The present callback is invoked and the requested records are retrieved from the native database and returned to the client. Finally, the client sends a close request, whereas the close callback is invoked, releasing any resources used by the native database.

More information on the Pizazz software distribution can be found at the project home page, http://webstar.ncsa.uiuc.edu/Project30/. The project team recognizes and appreciates the complexity of interfacing a communications server with a native database system and, accordingly, are happy to help with the effort in any way possible. Contact information may also be found on that Web page.

3. Conclusion

Designing a distributed, networked-based astronomical information system requires a four-step approach including original data source analysis, modeling the system, profiling the system, and implementation of the system. We described each step and proposed our solution. We have a project underway to implement such an information system and are committed to helping the astronomical community realize the dream of a robust, single interface to any and all relevant data sources on the Internet.