PARALLEL WEB SEARCH: MODERN APPLICATION ARCHITECTURES

Mihai Mocanu[1], Mihai Dorobanțu[1,2] and Mihai Popa[1]

[1] Department of Software Engineering, School of Automation, Computers and Electronics, University of Craiova, Romania, E-mail: mocanu@software.ucv.ro

[2] Health Information Systems Enterprise Architecture Division, Imaging Science and Information Systems, Georgetown University, Washington, D.C., USA

ABSTRACT

In virtual information world, containing billions or more Web pages, search engines is still the main mechanism for accessing information. Although today the main utility of the Internet is for interactive access to documents and applications, and in almost all cases, such access is mediated by human users (typically working through Web browsers or other interactive front-end systems), the situation is constantly changing. This paper is part of a more extensive work, in which we try to find appropriate ways to deploy and use applications as remote services, over the Internet. In particular, we are interested in visualization tools and distributed programs, as front-ends for distributed databases. We would like to treat applications, to a large extent, as "black boxes", and "wrap" them (i.e. into Java Classes and other structures), in order to enable their manipulation using increasingly available Web and Grid services infrastructures.

KEYWORDS: *meta-search, parallel search, Web Service (WS), SOA, SOAP.*

INTRODUCTION

An important issue in the design of global Internet systems is the presence of many models for data exchange, indexing, searching, and retrieval. Rapid advances of computer network technologies made more information sources accessible, and the vast amount of data available on the Web has set out the development of many approaches for heterogeneous data retrieval and information extraction, in which human intervention is minimized and application parallelism and complementarities are inherent. The Web is constantly extended to support communication between applications and there is an increasing interest in establishing services infrastructure for searching existing data and publishing new data in distributed, heterogeneous sources of information across the Internet. This paper describes the evolution of software architectures for searching distributed heterogeneous sources of information, in particular Internet sources, and proposes extensions to traditional approaches based on modern technologies. Our main goal here is to give, an overview of

WSs and to offer solid arguments for their increasing use in applications – in spite of some drawbacks existent today. Focus will be maintained in this paper on WSs – although an implicit relationship with Grids still exists.

PARALLEL WEB SEARCH: THE PROBLEM

Integration of heterogeneous information sources has been one of the most important issues in recent advanced application environments. In particular, with the broad acceptance of the Internet, integration of the Web and other information sources has been strongly required. Users' potential benefits are obvious, in two directions: one is being able to issue a single query and to obtain answers as fast as possible, through its *parallel, asynchronous* execution; the other is using a single interface and obtaining appropriate information from multiple sources instead of sequentially search source after source each with their own interface.

There are, of course, different approaches, that could be denoted as more *physical* or *virtual*. In a *physical* approach, data originating from local and remote sources or databases (DB) are integrated into one single new data source on which all queries can operate. In a *virtual* approach (also referred to as *multi-database system* in literature), data remains in the original local or remote sources; a layer is built on top of them, which takes the query from the user, processes it, sends (parts of) it to the appropriate sources and presents the results. Thus, queries operate directly on them and data integration has to take place during query processing.

There are also two possible choices on *physical* systems. One is to migrate data from the local systems to a *Universal DBMS* able to handle all (or many) types of information; the main drawbacks of this approach is that existing applications for the local systems have to be rewritten for the new database and the process of data migration can be very expensive, since the old data has to be transformed. In the other approach data from the local sources are imported into one DBMS, the *data warehouse* - the difference to the previous case is that the underlying data sources are still operational, so in fact the data is replicated. The warehouse data is typically not imported in the same form and volume as it exists in the local data systems. It may be transformed, cleaned and prepared for certain analysis tasks, like data mining and OLAP *(Online Analytical Processing)*. Data warehouses often do not make the most recent data available, since they are not usually updated immediately after a local data source has changed. With respect to querying, both major approaches have the advantage that real DBMS functionality is available, so only *precise* searching is supported, and the overhead for building such systems is important.

Three major methods can be distinguished. *Meta-search engines* provide for requesting several search engines and composing combined response, and have gained importance regarding querying of unstructured sources mainly due to the popularity of the Web. Their main focus lies with the

combination of results, so they are most suitable for unstructured data and support *imprecise* search. *Federated Database Systems* (FDBS) try to give the user the impression of working with one DBMS, but in fact the data is managed by several individual DBMS. Since a FDBS still provides typical DBMS functionality, queries support only *precise* search. FDBS may also be regarded to follow the physical approach because they may store parts of the underlying data in an internal repository. However, this materialization is only partial and/or temporary (i.e. to enhance performance). In *Mediation Systems*, query processing is very similar to meta-search, the difference is that data in the underlying sources may be heterogeneous, i.e. structured, semi-structured or unstructured.

SOA (SERVICE-ORIENTED ARCHITECTURE): THE MODERN SOLUTION

Problems summarized above make the retrieval of Web information a complex process. An application that obtains Web pages must remove from them irrelevant information such as banners, pictures, etc. Parsing a Web page to get only needed data involves writing particular code (a parser) for each Web site - data format is the biggest problem in the process of extracting information from Web sites (because Web sites are made for humans they suffer continue changes without any notification). Other problems are encountered on the server side - the use of Web sites by applications can cause the overload of the Web server, and to solve this case Web site owners have blocked the access of "simulated humans" to their pages.

WSs and Grid services are a promise to break the inconveniences through their increasing use in applications. The difference from the above solutions consists in that in most of the preceding systems ad-hoc solutions were applied. With these services, we have the promise of standardization, lowering the barrier to application integration.

Even though WSs are new, from an architectural perspective, they are based on established middleware design principles for application-to-application communication. From a historical standpoint, WSs represents the convergence between the service-oriented architecture (SOA) and the Web, through a specific protocol (SOAP). The use of the term refers more correctly to the *architecture, standards, technology and business models that make WSs possible*. In a simple definition, in agreement with many others given by companies that developed and used them (such as IBM, Microsoft or Sun), *WSs are loosely coupled, publicly available components that communicate through standardized XML messages and interfaces*. *Loosely coupled* means that WSs and the programs that invoke them can be changed independently of each other. As well as a Web Browser that communicates with a Web Server without knowing what is on the other side, the WSs Server is not interested in the kind of client (a Web Browser client or non-browser client) that uses it. This agreement also implies that WSs are *platform independent*. The WSs Server can be a Java-

based implementation (Apache Axis, Java version) running on a Linux OS and the client can be a .NET application running on an Windows platform. *Publicly available* means that a WS's behavior, its input and output parameters, and how to bind to it can be obtained by everybody through its description [1].

Grid services are related to *grid computing*, which is *distributed computing over the global network enabled by open standards*. Grid standardization has also been an important issue over the last years, and specifications are covered in bodies such as: Global Grid Forum (GGF), WWW Consortium (W3C), or Internet Engineering Task Force (IETF). GGF, especially, has played a key role in developing and articulating the Open Grid Services Architecture (OGSA), which defines the Grid community's "guiding principles" for WSs/ Grid convergence [2].

Even if implementing a WS can be a complex job for an important company, the revolutionary part of this approach is the separation of the human part (Web pages) from application part (WSs). Applications may use a WS and the humans will continue to use the Web site. Service descriptions (in a WSDL file) can be publicly available and it will make the communication between any application and the WS server very easy to realize.

The illustrative application described in this section is intended to support our allegations on the commodity and robustness in using WSs to implement parallel Web search. We built it as a *demonstrative meta-search engine*, using Apache Axis. Axis is essentially a *SOAP engine*, a framework for constructing SOAP processors such as clients, servers, gateways, etc [3]. Apart from being (only) a SOAP engine, Axis also includes: a simple stand-alone server; a server which plugs into servlet engines such as Tomcat; extensive support for the WS Description Language (WSDL); emitter tooling that generates Java classes from WSDL (**WSDL2Java** tool); and a tool for monitoring TCP/IP packets (**TCPMon** tool). The application introduces a new level (middleware) between humans and Web sites. Generally, the scope is to refine the information provided to humans (e.g.: meta-search engines) or to collect important amounts of data for statistics, predictions, etc. We have chosen to implement a meta-search engine because it can face all problems described above: it must parse the results obtained from Web search engines and nobody can guarantee the format of data will be the same next week or month. Our very basic meta-search engine is based on two WSs provided by two important companies: Google and Amazon. We have chosen these WSs because they are freely available at this moment on the Web. Using only the descriptions of these two WSs we could implement a client for each of them, and a common user interface, showed in Fig. 1.

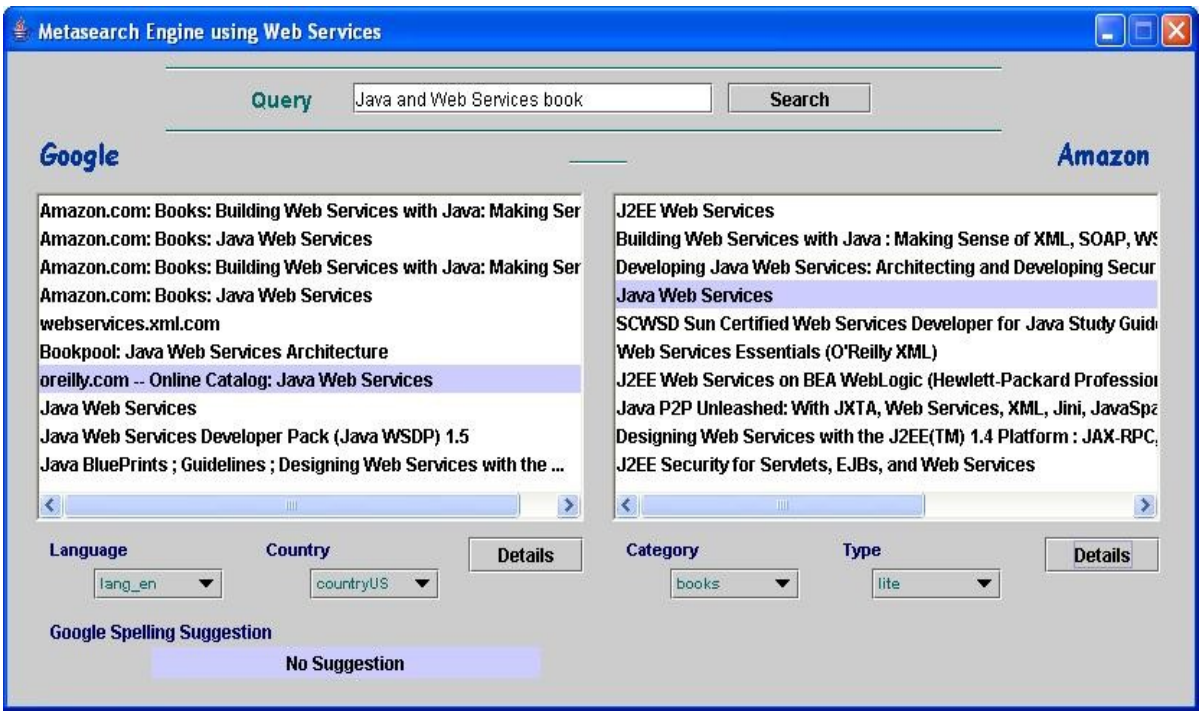

Fig. 1. A Metasearch Engine based on Web services

The descriptions of these WSs are too big to be put here but they can be downloaded and examined from the Amazon or Google web sites [4] [5]. The implementation did not request any code for parsing the data because it did not deal with HTML files. It obtained from Google and Amazon only the useful information in an XML format. The XML messages encoded as SOAP Messages that are returned from these WSs contained detailed information for each result. In fact, all information that is available to users on the Web site can be accessed through the WS. The operations deployed in these WSs are complex, we used only a small part of them.

CONCLUSIONS

An application-centric Web is not a new notion. For years, developers and companies have created programs and Java servlets designed primarily for use by other applications (i.e., search systems or news retrieval systems). The interaction of applications with Web pages for getting useful data is a complex issue and raises multiple problems, due to multiple and various factors: firstly, because Web sites are made for human users not for software applications; then, because precision of search is unavoidably low, as a consequence for the simplicity of home pages “registration” for the whole Web; third, because of the lack of standardization in the field; and, last but not least, because of the unwanted delays in performances, in both the development and execution of applications, either in the communications or deployment phases or during resource classification.

Using WSs made the implementation of a meta-search engine relatively easy to do because all results are obtained in a precise format described by WSDL files and so the data format issue was

completely eliminated. The implementation did not request any code for parsing the data because it did not deal with HTML files. Another big advantage of this approach lies in the automation of the process: once implemented the application can be perfectly functional while there is a WS running. The maintenance of the code may be very low or even null.

REFERENCES

[1] Cerami, E., Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, O'Reilly Publ., 2002

[2] Foster, I., C. Kesselman, S. Tuecke. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* Int. Journal Supercomputer Applications, **15**(3), 2001

[3] Gibbs K., B.D. Goodman, and E. Torres (2003), *Create Web services using Apache Axis and Castor***,** http://www-106.ibm.com/developerworks/ webservices/ library/ws-castor/

[4] Web site: Amazon, http://soap.amazon.com/schemas2/ AmazonWebServices.wsdl

[5] Web site: Google, http://www.google.com/apis/