Internet Based Large Distributed Geospatial Databases

Yaser A. Bishr
Institute for Geoinformatics
University of Münster
Email: bishr@ifgi.uni-muenster.de

Key Words: Distributed processing, parallel processing, mobile agents, Geospatial data, Internet.

ABSTRACT

The accessibility of geospatial data on the Internet poses several technical challenges. One of the major challenges is the relatively large size of geospatial databases for which they consume network and computer resources when searched and retrieved in a distributed environment.

The allocation of data and operations to nodes in computer networks is a critical issue in distributed system design. An efficient distributed design must trade off performance and cost among retrieval and processing activities at the various nodes. It must consider the concurrency control mechanism used as well as capacity constraints at nodes and on links in the network. It must determine where data will be allocated, where they will be processed, the degree of data replication, and where operations such as image processing tasks will be performed.

The objective of this paper is to propose a system that has architecture and a mechanism to improve the performance of distributed geospatial systems by reducing the data transferred on the network and simulating the parallel processing mechanism. The paper will also show an implementation of this system.

1. Problem and Motivation

It seems that we are overwhelmed by the glare of providing Internet based geospatial services without so much consideration to the potential problems where the accessibility of geospatial data on the Internet poses several technical challenges. One of the major challenges is the relatively large size of geospatial data sets, especially satellite images, for which they consume network and computer resources when searched and retrieved in a distributed environment.

The telecommunications industry is laying down astonishing amounts of fibre. Although Internet traffic is growing exponentially, the bandwidth soon to be available on the Internet backbone, as well as to many offices and neighbourhoods, is immense. Nonetheless, bandwidth to many end users will remain limited by several technical factors. Many users will still connect via modem, or at best, ADSL over the old copper loop. Many other users will connect via low-bandwidth wireless networks. Most users can expect to see no more than 128 Kbps to 1 Mbps available at their desktop or palmtop, although some asymmetric cable modems may reach 10 Mbps (for downloads) .

Perhaps more importantly, the *gap* between the low-bandwidth ``edge" of the network, and the high-bandwidth ``backbone" of the network, will increase dramatically as the backbone benefits from increased quality and availability of fiber, while the edge remains limited by the fundamentals of wireless and copper connections. We expect that this trend will continue even as local connections improve past 1 Mbps in the next few years, since backbone bandwidths are improving much faster than local bandwidths.

Users of current GIS and image processing applications usually retrieve all required data for processing on their local machines. In client server architecture users store the data on the server and perform their operations remotely. There are several problems associated with these two approaches:

1. The Internet bandwidth is consumed during the transfer of the large data set.
2. Processing geospatial data set, e.g., image processing, requires powerful computers. This may use the computer resources whether it is a local machine or a server.

Our approach to solve the above problems is twofold. Firstly to solve the bandwidth problem we send the processing operators (algorithms) to the data sets that are distributed on the Internet and only retrieve the result, instead of retrieving the data themselves. Secondly, to solve the computer resource problem we simulate the parallel processing technology (Oscar 1994), by creating a processing plan that allows the processing operators to be sent to different data sources and cooperatively execute the processing plan.

The objective of this paper is to propose a system that has an architecture and a mechanism to improve the performance of distributed geospatial systems especially bandwidth consumption and distributed processing. The paper will also show an implementation of this system.

The system uses concepts from robotics, where autonomy, pro-activity and intelligence are the main characteristics. These are called mobile agents (Wiederhold, 1989). An agent is a computational entity which acts on behalf of other entities in an autonomous fashion; it performs its action with some level of pro-activity and reactiveness, and exhibits some level of the key attributes of learning, co-operation and mobility. The paper will conclude with a discussion on the different applications that can benefit from mobile agent technology and the profound implications that may result on system performance, innovative services to users who use mobile devices, and future research issues that require the attention of the research community.

Agent based technology is becoming more important as the complexity of computing systems increases. An agent is a mobile computational entity which travels independently between networked computers and acts on behalf of other entities (users or pieces of software) in an autonomous fashion. It performs its action and reacts to requests with a degree of autonomy.

An agent may run when the user is disconnected from the network, even if the user is disconnected involuntarily. This ability greatly reduces network traffic and network consumption. Mobile agents provide specific advantages over traditional software by providing the ability of intelligently query the information directly at data storage location. At the data storage location high bandwidth communications can take place between the client and server. This dramatically reduces network load and query time.

2. Distributed Systems and Component Architecture

Dynamics in open computer environment can be considered in two aspects. Firstly, *dynamic evolution* of service items where services are regularly upgraded during their life cycle. New services are introduced in place of previous ones. In this respect, component-based approaches are appropriate for open system designing to manage system evolutions more easily [Nierstrasz 94]. Secondly, *dynamic cooperation* between services; for example, the travel agency has organized a trip for a customer who wants to go from Bordeaux (France) to New York (USA), e.g., ticketing for rail-road (from Bordeaux to Paris) and flight (from Paris to New York), and hotel reservation (at New York), etc. But it is liable that the flight schedule is suddenly changed (delayed or cancelled). Accordingly the well arranged travel plan must be revised as like finding another route or postponing the departure, which affects also the hotel reservation. This scenario shows that it is essential for open system designers to deal with complex *dynamic cooperation* between services. That is why we used multi-agent based approach to model Open Distributed Cooperative System (ODCS), since it focuses on agents interactions.

In this paper we focus on the performance, behaviour and processes of geospatial distributed systems, i.e., what these systems do. Less attention is given to information and data handled in a distributed system, i.e., what they manipulate. The latter problem is also called semantic heterogeneity and we have extensively demonstrated its complexity (Bishr et al. 1998) and introduced solutions and prototypes (Bishr 1997, Bishr et al. 1998, Bishr et al. 1999).

3. Mobile Agents

Our notion of an agent is simply derived from the more clearly defined notion of a *mobile* agent in which it is a program with the ability to move during execution. Agent based technology is becoming more important as the complexity of computing systems increases. Mobile agents are defined as independent software programs, which run on behalf of a network user. An agent is a mobile computational entity which travels independently between networked computers and acts on behalf of other entities (users or pieces of software) in an autonomous fashion. It performs its action and reacts to requests with a degree of autonomy.

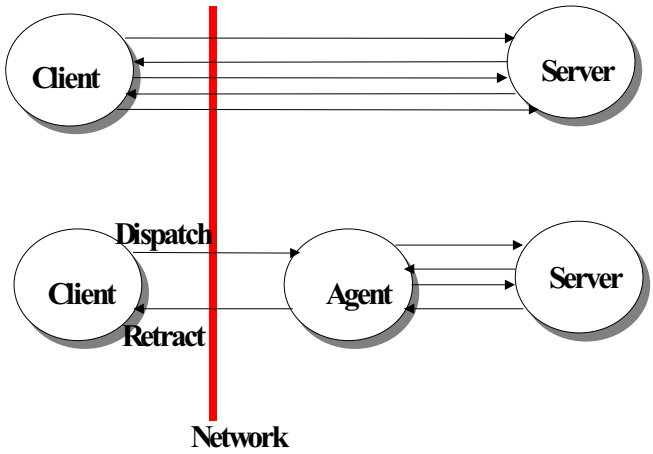

Figure 1 Advantage of mobile agents over client-server

Mobile agents provide specific advantages over traditional software by providing the ability of intelligently query the information directly at data storage location. At the data storage location high bandwidth communications can take place between the client and server. This dramatically reduces network load and query times as shown in figure 1.

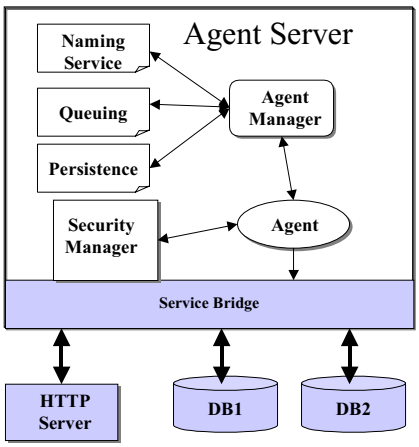

Figure 2 The Agent environment

Agent servers are middleware that connects between agents. Mobile agents servers can be built in conjunction with e.g. a web server (Degano 1999, Genesereth, 1992). Here, users retain complete access to the Internet via the web server, while the mobile agents servers are contacted via the local web server and are only activated upon receiving requests or other agents. Figure 2 shows a general architecture of a mobile agent server. The agent manager that is contacted via the web server provides the communication services that allow agents to be transmitted and received by other mobile agents servers. The agent manager is also responsible for controlling the life cycle and queuing for executions, through the persistence and queuing services, respectively. The naming services are similar to those found in CORBA and other client/service architectures; it provides a mechanism to locate agents and other services in a network. Mobile agents require a special security service in addition to the HTTP security features. This is because mobile agents can access resources and collect data from local machines along their trip. The security manager is responsible for identifying users, authenticating their agents, protect server resources, and ensure that agents are not allowed to violate data access rights. The service bridge provides the interfaces necessary for agents to access resources that are outside the agent services, e.g., HTTP services.

Agents can implement either pull or push models, or both (Martial 1992). The pull model is similar to the standard Java architecture, in which a host sends a request to a server to ask for a java service (which is in fact a java class) the class is then transferred directly to the calling host as byte code where it executes. In the agents pull model, clients receive agents that have a certain state, i.e., its object variables have values. In the *push model*, agents can independently decide

to leave a host and move to another without any external request (via agents itineraries). When agents are instantiated, the itinerary object is created dynamically, in which it provide the agent with the trip information and the tasks that agents have to do in each stop. An important feature of agents is that when they travel between hosts, they maintain their state, e.g., the results of a query.

Java based Agents that travel between hosts, i.e., mobile agents, should implement the Java Interface ' Serializable' (Gmytrasiewicz 1991). It is important to make sure that all object variables of agents are also serializable otherwise; it cannot get transferred to other locations. Agents can perform tasks either at their local host or remotely by sending messages to remote hosts. In this sense, we can either wait for an agent to finish its processing on a server and then package the results and send them to the next host along with the agent, or create a serialized result object for each resulting item and stream to other hosts, which may have other agents waiting for the result (Zlotkin 1994). The agent developer has then the choice of sending the serialized object synchronously or asynchronously. This provides a mechanism to develop highly parallel implementations.

4. Mobile Agent Prototype for Image processing

A prototype was developed to demonstrate how can mobile agents provide a flexible and yet powerful environment for distributed processing on the web. The prototype runs in a java environment and uses the Aglet mobile agent application-programming interface, API, developed by IBM (Lange et al. 1998). The Aglet is a mobile Java agent that supports the concepts of autonomous execution and dynamic routing on its itinerary. Aglets are hosted by an Aglet server in a way similar to the ay applets are hosted by a web browser. The Aglet server provides an environment for aglets to execute in, and the Java virtual machine (JVM) and the aglet security manager make it safe to receive and host aglets. The aglet API is an agent development kit, which has a set of Java classes and interfaces that facilitate you to create mobile Java agents.

The objective of the prototype is to prove the concept developed in this paper of distributed and parallel processing of geospatial data on the Internet. The prototype is designed to allow users to perform image-processing operations on the Internet. Current architectures of image processing allow users to retrieve and process the data in one processing environment. As mentioned before, this architecture consumes the Internet bandwidth and only provides a single processor environment to process all the data.

Fout! Ongeldige koppeling.
Figure 3 High level architecture of the prototype

As shown in figure 3, an agent library hosts a set of image processing operators. These operators are aglet classes, which are instantiated by an agent manager. To simplify the prototype it is assumed that clients know the location of the data sources and the processing sequence they wish to perform. A client who wants to perform a sequence of image processing operations posts a query to the agent manager. The agent manager receives the location of the data sources and the kind of processing to perform on them and constructs what is called " agent itinerary" . Ideally a catalogue of data sources should be available in which the agent manager can automatically identify the location of data and automatically constructs the itinerary.

The agent manager creates instantiate a master agent, which will be responsible for communicating with the other agents. According to the itinerary, the master agent instantiates the operator agents, retrieved from the agent library, and send them to the remote data source with well-defined tasks according to the itinerary. The operator agents know the location of each other and can independently exchange messages and coordinate their operations without intervention from the master agent.

Fout! Ongeldige koppeling.

Figure 4 The prototype flow of events

Figure 4 shows the flow of events and data of the prototype. Initially the user would like to subtract a mask from an image retrieved from database 1. The resulting image should then be convoluted with an image from database 2. An image from database 3 is first subtracted from the mask and then finally composed with the resulting convoluted image with an Alpha Filter. The pseudo equation of the operation is as follows. For simplicity we call the images from databases 1,2 and 3 as images 1, 2, and 3 respectively.

Compose (***Mask*** (image 3) , ***Convolute*** (image 2, ***Mask*** (image 1))

It is clear that the masking operation of image1 and image 3 can be done in parallel. As shown in figure 4, the master agent dispatches the mask to database 1 and 3 with the task to perform the subtracting operation in both sites in parallel. According to the itinerary and after subtracting image 1 at database 1 the operator aglet independently travels to database 2 carrying the resulting image where it convolutes it with an image from database 2. Meanwhile the operator aglet that was dispatched by the client to database 3 has already subtracted the mask from an image there and deactivated itself awaiting the arrival of the convoluted image. Once arrived the deactivated aglet activates itself and start the last operation of composing the final image, which is sent back to the client for rendering.

5. Discussion

Instead of repeatedly sending data to be processed by a remote processor, the processor is sent in one act to the remote data for on-site execution. Data and assorted services can be accessed locally then, and no ongoing connection is needed during the execution. Such a mobile processor, consisting of executable code and some form of execution state, has been termed a *mobile agent.* Mobile agents have raised considerable interest as a new concept for networked computing with potentially very far-reaching implications, and numerous software platforms for various forms of mobile code have recently appeared and are still appearing [CGH95, CMR+96, GRA96, HMD+96, LAN96, LDD95, JRS95, RAS+97, SBH96], with different foci and exploring diverging solutions. Still, the basic idea is quite simple: *Give programs the ability to move.* It seems natural, therefore, to *add* mobility to the large and well-developed world of programming, rather than attempt to build a new realm of " mobile programming" . Mobility should be integrated as comfortably and unintrusively as possible with existing programming concepts — algorithms, languages, programs, and operating systems. This is the basic idea of I-conquest. A platform for agents that carry scientific operators able to move freely and easily at their own choice and without interfering with their execution, and most importantly utilizes the various components of the original Conquest system, to maximize components reuse.

The prototype simulated a parallel processing environment on the web, which can exponentially increase the speed of processing and eventually receiving the results. In order to evaluate the ability of the mobile agent technology to increase the processing speed, the same operations mentioned above were performed on a single machine. The same Java code (without the mobile agent capabilities) was used and was executed on an NT machine with 500 MHZ PIII processor, and 128 MB RAM. To accurately measure the execution time, a batch file was created to automatically download the file from three intranet servers, with the same configuration, and automatically launch the java classes. In our prototype, the same three servers were used and the local NT machine was only used to host the master agent, which dispatched the operator agents. It was found that the same processes in a mobile agent environment were executed 1.32 times faster than a single machine execution.

6. Research Issues

Although the prototype showed the great potential of mobile agents, it was clear to us that a structured methodology to design distributed component architecture and implement agents on them is required. The ability of agents to move between servers and execute tasks autonomously makes the design of any application more complex than just the distribute component architecture. This is due to the fact that distributed components communicate via messages, while mobile agents adds to that by travelling to machines for autonomous execution.

During the conceptual design phase the system architect have the choice to design software components that performs tasks locally and communicate with each other by messages, e.g. the JAVA RMI, to send the component, i.e., agent, to the remote hosts and perform the task there. It is not always advantageous to send mobile agents to the remote hosts, which could be busy in other processes. Instead it is sometimes better to ask the host to send the data and perform the task locally.

The allocation of data and operations to nodes in computer networks is a critical issue in distributed system design. An efficient distributed design must trade off performance and cost among retrieval and processing activities at the various nodes. It must consider the concurrency control mechanism used as well as capacity constraints at nodes and on links in the network. It must determine where data will be allocated, where they will be processed, the degree of data replication, and where operations such as image processing tasks will be performed. Therefore more research is required to develop methodologies to design mobile agent applications. We believe that workflow simulation case tools (e.g., simple++) can provide a good environment to test the design concept prior to implementation.

7. Conclusion

Software agents are programs that assist people and act on their behalf. Agents function by allowing people to delegate work to them. Mobile agents have the unique ability to transport themselves from one system in a network to another. The proliferation of the Web triggered several geospatial data providers to put their data on the web. In remote sensing, there are numerous web servers that provide extensive catalogues of satellite images. Usually users download the data from different catalogues for processing on their machines. This activity may pose a heavy load on the Internet. This paper showed how mobile agents technology can improve and speed the image processing tasks by providing a virtual parallel processing environment were Internet servers cooperate together to perform tasks without necessarily knowing each other.

REFERENCES

Bishr Y., Pundt H., Ruether C., 1999. Proceeding on the Road of Semantic Interoperability – Design of a Semantic Mapper Based on a Case Study from Transportation. In the proceedings of Interoperating Geographic Information Systems, Second International Conference, Interop' 99, Zurich, Switzerland, March 10-12, pp. 203-216, Andrej Vckovski and Kurt Brassel, (eds).

Bishr, Y. 1998. Overcoming the semantic and other barriers to GIS interoperability. In Vckovski, A. (ed.) Special Issue on Interoperability in GIS. International Geographical Information Science, Vol 12 No 4, Jun 1998, pp299-314.

Bishr, Y., 1997. Semantic Aspects of Interoperable GIS. Publisher, The International Institute for Aerospace Survey and Earth Sciences, ITC. ISBN 90 6164 141 1.

Martial F. 1992. Coordinating plans of autonomous agents, LNCS 610, Springer-Verlag.

Harvey F., Kuhn W., Pundt H., Bishr Y., and Riedemann C., 1999: Semantic Interoperability: A central issue for sharing geographic information. The Annals of Regional Science, vol. 33, number 2.

Oscar Nierstrasz, Theo Dirts Meijler, Requirements for a Composition Language, ECOOP 94 Workshop on Models and Languages for Coordination of Parallelism and Distribution, Italy, LNCS 924, Springer-Verlag, 1994.

Lange D. and Oshima M., 1998, Programming And Deploying Java Mobile Agents with Aglets. Addison Wesley Longman, Inc., One Jacob Way, Reading, Massachusetts 01867,. ISBN 0 201 325829.

Degano P., Priami C. Non interleaving semantics for mobile processe, 1999. Theor Comput Sci 216(1– 2): 237– 270.
Genesereth, M. R. 1992. An Agent-Based Approach to Software Interoperability, In Proceedings of the DARPA Software Technology Conference.

Wiederhold, G. 1989. The Architecture of Future Information Systems, Stanford University Computer Science Department, 1989.

Zlotkin, G. 1994. Mechanisms for Automated Negotiation among Autonomous Agents. Ph.D. dissertation. Hebrew University.

Gmytrasiewicz, P. J., Durfee, E. H. and Wehe, D. K. A 1991. Decision Theoretic Approach to Coordinating Multiagent Interactions. In Proceedings of the Twelfth International Joint Conference On Arti¯cial Intelligence (Sydney, Australia 1991). International Joint Conferences on Artificial Intelligence, Inc. pp. 62-68.