N95-17199

ARACHNID:

A PROTOYPE OBJECT-ORIENTED DATABASE TOOL

FOR DISTRIBUTED SYSTEMS

Herbert Younger
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

John O'Reilly, PhD
Bjorn Frogner, PhD
MIMD Systems, Inc
3145 Porter Drive
Palo Alto, CA 94304

ABSTRACT

This paper discusses the results of a Phase II SBIR project sponsored by NASA and performed by MIMD Systems, Inc. A major objective of this project was to develop specific concepts for improved performance in accessing large databases. An object-oriented and distributed approach was used for the general design, while a geographical decomposition was used as a specific solution. The resulting software framework is called ARACHNID.

The Faint Source Catalog developed by NASA was the initial database testbed. This is a database of many giga-bytes, where an order of magnitude improvement in query speed is being sought. This database contains faint infrared point sources obtained from telescope measurements of the sky. A geographical decomposition of this database is an attractive approach to dividing it into pieces. Each piece can then be searched on individual processors with only a weak data linkage between the processors being required.

As a further demonstration of the concepts implemented in ARACHNID, a tourist information system is discussed. This version of ARACHNID is the commercial result of the project. It is a distributed, networked, database application where speed, maintenance, and reliability are important considerations.

This paper focuses on the design concepts and technologies that form the basis for ARACHNID.

INTRODUCTION

Progress in the field of software for multiple processors is lagging behind the progress made in development of the processors themselves. A key issue is development of effective algorithms that can distribute the load among the processors in a manner that is transparent to the application developer.

PRECEDING PAGE BLANK NOT FILMED

PAGE 176 INTENTIONALLY BLANK

Significant R&D progress has been made throughout the industry during the last few years[1], but the distance between actual versus potential throughput is still very large. For these reasons, it is an exciting field since the technical issues are challenging and the business opportunities are numerous.

The use of multiple processors for searching databases spans the field from massively parallel processors, where each CPU is very low cost, to distributed systems where each CPU essentially is a separate computer (e.g., PC); e.g., see refs[2,3]

PROJECT OBJECTIVES

Phase I of the project started with the goal of using parallel processors to achieve dramatic speed improvements. However, at the start of Phase II, the focus was changed to distributed systems as an arena where more cost effective solutions could be found. The potential gain in this arena was judged to have a larger commercial potential since networked PCs are so widely available.

The specific objectives of Phase II were as follows:

- Develop algorithms and software for distributed access to large databases.
- Demonstrate and test the software.
- Develop a commercialization plan.

Phase II proceeded through a progression of tasks that are typical for a software R&D project: design specifications, implementation, testing, evaluation, and documentation.

When this paper was written, the focus was on the commercialization of the technology.

RELEVANT TECHNOLOGIES

Resource Allocation

Using two or more processors to solve one computational problem (e.g., search through a database looking for specified aggregate results) can be treated as a resource allocation problem[4]. The issue is how to distribute the load onto the available CPUs considering their performance, the speed of communication between them and the coupling between the sub-problems solved by the processors.

Thus, this project did a substantial review of mathematical programming methods (e.g., Dynamic Programming) to determine their possible contribution towards solving this resource allocation problem. It was finally determined that a heuristic approach derived from an understanding of the unique nature of the problem had to be developed.

A geographic decomposition approach was found to be of sufficient general utility for ARACHNID. Although we cannot quantify the extent to which this is a sub-optimal solution, there is a sufficiently large number of problems for which this is an attractive approach.

Object-Oriented Databases

Organizing data in terms of logical units (e.g., records in a database) is well proven. Using an object-oriented approach[5-8] is relatively new and in many ways very appealing. Thus, this project made a review of the available object-oriented databases that were available at the time the project started.

Leading products (e.g., Vbase[9], GemStone[10], and Iris[11]) were evaluated with respect to their relevance to ARACHNID. Although, it was found that these products would give high flexibility and powerful data representations,

they would be cumbersome for a distributed configuration and difficult to optimize for computationally intensive problems.

The final solution was to use a product called C Data Manager (CDM), which is a C-callable library. It has a low-level, object-oriented, database engine with a high degree of flexibility for the developer. However, it does not have the robustness for concurrent access as we all have become accustomed to in relational databases. Thus, CDM was used for storage of local data, mostly in support of the user interface and temporary results.

SYSTEM REQUIREMENTS

Functional Requirements

ARACHNID is based on object-oriented and distributed technologies. Users and application developers are provided with an object-oriented environment, including encapsulation, object identity, persistence, and inheritance.

ARACHNID is designed to integrate with existing databases to provide object persistence. The design allows for interfacing with multiple database engines.

ARACHNID's major technology features are summarized as follows:

- *Object-Oriented.* It uses an object-oriented database for storage of its own local data.

- *Local Autonomy.* Once initiated, it does not rely on external processes for its operation. Furthermore, if a particular server fails, only clients associated with the server objects will be hampered.

- *Object-Oriented Interface.* It provides an object-oriented interface and a set of support utilities for both the database administrator and application developer.

- *Fragmentation Transparency.* It hides the storage fragmentation of an object. Hence, it manages aggregate object types (sometimes multimedia data) and presents the entity as a single object.

- *Distributed Transaction Management.* At the internal level, it relies on the underlying database subsystem for object recovery control.

- *Operating System Independence.* The design assumes that the underlying operating system supports a message-passing paradigm and a client/server architecture.

DESIGN

ARACHNID is based on a distributed client-server model (see Figure 1) in which an interconnected set of servers uses an object-oriented approach to provide a high-level database facility. ARACHNID interfaces to other system-level and application-level software that can access arbitrary data structures across the network.

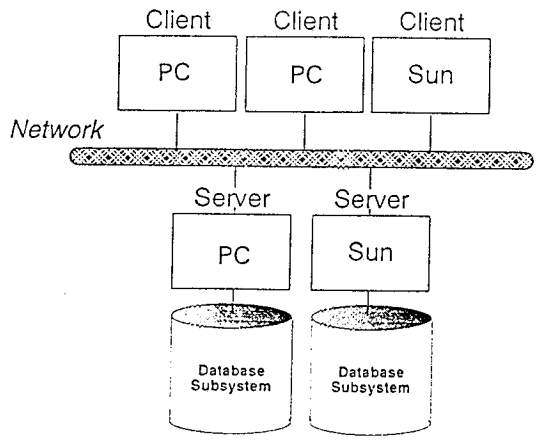

Figure 1: Client/Server Configuration

Although ARACHNID is an object-oriented tool for accessing databases, it is not an object-oriented database management system. Instead, it accesses existing databases that have been organized (and are managed by) other DBMSs, such as Sybase, Paradox, and Oracle. These databases are stored on computers networked with ARACHNID. The system's services are connected in a web-like fashion, thereby logically leading to its name: ARACHNID.

ARACHNID was designed to realize the potential of multiple processors for complex operations on databases located on geographically distributed, heterogeneous computers. ARACHNID accesses data from these databases and transports it to its client computer when needed.

In ARACHNID, a query is divided into small components ("fragments") and each fragment is allocated to a computer on the network. A query module resides on each network computer and controls the execution of fragments on that computer. A control node coordinates the distribution of fragments to network computers, as well as the collection of fragment queries implemented on those computers.

As an example of fragments and objects, consider a form with several fields of which some have methods that are activated when selected by the user. Assume that some of these methods will bring up other forms with data from other databases. In ARACHNID, each field in the form is an object while one or more forms can be classified as a fragment if it represents a natural unit to be executed as one entity.

The user interface for ARACHNID employs the latest in Graphical User Interface (GUI) methods. The ARACHNID user screens has a "point-and-click" environment that makes the creation of user requests quick and easy. ARACHID is available on both MS Windows 3.1 and UNIX. Figure 2 is an example of an input screen for generation of a query to the Faint Source Catalog on a SUN computer.

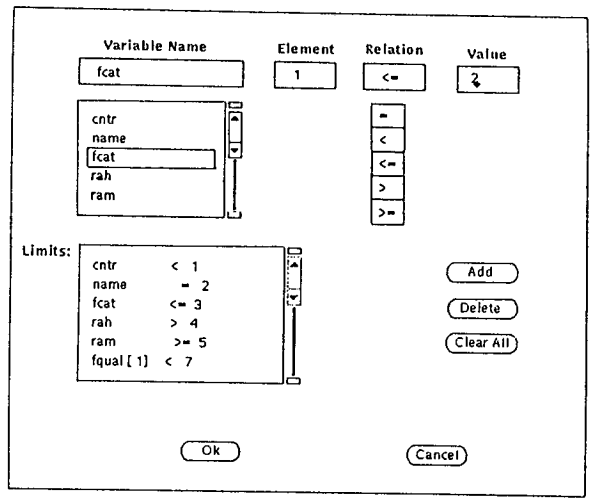

Figure-2: Example of a Query Input Screen

DATABASE DECOMPOSITION

Decomposition can be used to divide a domain into a number of non-overlapping subdomains. This approach has seen many applications in large matrix problems, and it is particularly appropriate for the decomposition of large celestial regions into smaller regions.

Databases often have a high degree of independence between subdomains. When a subdomain is retrieved or updated, other domains are often not involved. This independence is not only a product of the distribution of data -- it is a common occurrence in databases because the designer typically created the database schema that way.

The distribution of astronomical sources can be represented in a spherical coordinate system, with each source described by two arguments: equinoctial α and ecliptic ß, as shown in Figure 3. It is natural and

convenient to search concurrently all stars within a given distance from a fixed position.

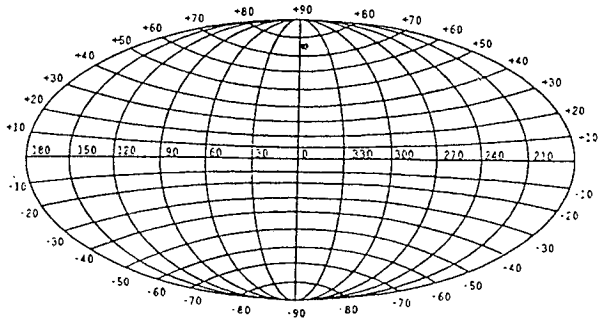

Figure-3: Decomposition for a Sphere

The following two factors were not considered in the above discussion. First, the spherical meshes are not uniform if the decomposition is made according to the method shown in Figure 3. Second, the distribution of the stars in the sky is not uniform. In fact, the distribution is quite irregular, as can be seen in Figure 4.

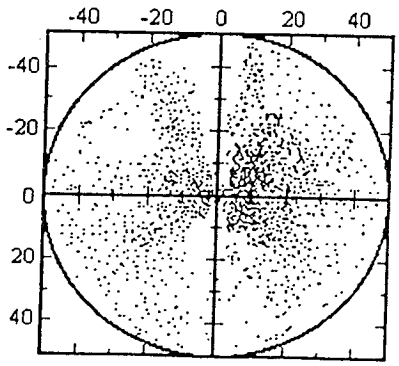

Figure-4: Distribution of Stars

Thus, for sparse star domains, a coarser net is better than a fine net. The allocation problem is then to divide the small group into even smaller groups and to distribute those components in a balanced manner.

Most computer systems provide efficient file managers that allow multiple users simultaneous access to reading a file. Thus, by distributing subsets of the database across multiple processors and each processor only having occasional need to read data residing on another processor, a decomposition method can be effective for finding (for example) objects having a given brightness, a given color, or location within a certain angular separation of a given point.

COMMERCIAL APPLICATION

The objective of SBIR projects is to develop technology that can benefit the sponsor and result in commercial products for the contractor. This project very much followed this path.

The concepts of geographic partitioning of databases developed for ARACHNID has been the basis for a key element of a new tourist information system. Consider such a partitioning in the context of the Miami - Fort Lauderdale area. A system installed in Miami covers that part of Florida that is closest to Miami. The same statement is correspondingly true for Fort Lauderdale. Now consider a user that needs directions from Fort Lauderdale to locations near Miami.

To solve this problem, it is necessary to make a decision with respect to the location of the database which contains the driving directions. There are three simple but unattractive choices:

- The database can be restricted to only cover a local area. This represents the simplest design. However, it is not attractive since there will be a wide

network of fairly closely located ARACHNID systems.

- The entire database could reside in one central location. This is a relatively simple design but it has several performance related problems; e.g., less than timely response to the queries and unacceptable down-time when the network fails.

- The database can have overlap between the various locations. This also represents an easy design; however, from a maintenance point of view it is unattractive to keep duplicate copies of portions of the database.

Thus, ARACHNID is designed to access databases at remote locations for long distance driving directions. Since the directions are stored as a set of connected nodes, a database query can easily result in "hits" from more than one location. It is expected that the users of this system will find it acceptable to wait a little longer for long distance directions than local directions.

RESULTS AND CONCLUSIONS

This paper has described a Phase II SBIR project for NASA performed by MIMD Systems. The objective of this project was to develop algorithms for distributed access to a certain class of large databases.

Much of the ARACHNID software has been implemented on both 486-based PCs and SUN SparcStations. Most of the NASA-specific development and testing was done on SUN computers, while the commercial version is primarily for the PC.

NASA's Faint Source Catalog was used as the initial testbed for a geographical decomposition of the database. Although more work is needed to harness the potential gain, there are good indications that the established goal can be achieved. The limiting factor is in the speed of communication between the processors and the cost involved in implementing the distributed hardware and software configuration.

For the commercial version of ARACHNID, the distributed approach has proven to be very valuable in terms of providing competitive performance, reliability and maintenance. This tourist information implementation of ARACHNID is now being installed on a nationwide basis for a major tourism industry company.

REFERENCES

1. DeWitt, D.J., Gray, J.: "Parallel Database Systems: The Future of High Performance Database Systems." Comm. ACM 35 (6), 85-98, 1992.

2. Sakti, P.; Ho, K.M., "Generalized Parallel Processing Models for Database Systems," Proc. of the 1988 Inter. Conf. on Parallel Processing, 1988.

3. Fox, G. et al., Solving Problems on Concurrent Processors: Volume 1, Prentice Hall, 1988.

4. McEntire, P.L.; and Larson, R.E., "Optimal Resource Allocation in Sparse Networks", IFAC/81, 1981.

5. Blaha, M.R.; Premerlani, W.J.; and Rumbaugh, J.E., "Relational Database Design Using an Object-Oriented Methodology", CACM, Vol 31. No. 4, April 1988.

6. Deppisch, U. P.; and Schek, H.J., "A Storage System for Complex Objects", Proc. of the 1986 International Workshop on Object-Oriented Database Systems, pp. 183-195, 1986.

7. Dittrich, K. R., "Object-Oriented Database Systems: the Notion and the Issues," Proc. of the 1986 International Workshop on Object-Oriented Database Systems, pp. 2-4, 1986.

8. Meyer, B., Object-Oriented Software Construction, Prentice-Hall, 1988.

9. Ontologic Corporation, Vbase Functional Specification for Release 1.0, 1987.

10. Maier, R., et. al., "The GemStone Data Manage-ment System", Object-Oriented Concepts, Database, and Applications, ACM Press, 1989.

11. Fisherman, D.H. et. al., "Overview of the Iris DBMS", Technical Report HPL-SAL-89-15, Hewlett-Packard Laboratories, 1989.

Acknowledgements - The work presented in this paper was partially sponsored by NASA Phase II SBIR Contract # NAS7-1156. While the authors have the sole responsibility for the contents of this paper, we wish to recognize the contributions of: Dr. Joe Mazzarella at NASA/JPL and the following individuals at MIMD Systems: Dr. Robert E. Larson and Dr. Paul L. McEntire.

Data Management

3. Telemetry Processing

* *Presented in Poster Session*

PRECEDING PAGE BLANK NOT FILMED