Alternative Technologies™

SUCCESSFUL USES OF DATABASE MACHINES

David McGoveran
Alternative Technologies
150 Felker Street, Suite E
Santa Cruz, California 95060
408/425-1859

During the last few years, most relational database experts have agreed that database machines have a niche market. What is that market? Will it survive and if so, why? Why is it that some relational database applications succcessful and others are not? How do database machines differ from software database servers running on off-the-shelf hardware?

All of these questions revolve around what, if any, applications are likely to be successfully implemented on a database machine. This question does not have a simple answer. Vendors of database machines would like users to believe that database machines are simply better, less expensive solutions. Vendors of software databases would like users to believe that database machines leave the user vulnerable to hardware maintenance problems and unable to take advantage of off-the-shelf hardware advances. Neither of these views responds to the issue of why some applications might or might not be more appropriately implemented on a database machine.

Database machines are database servers implemented on special purpose hardware. Today there are two dominant vendors in this market, Teradata and Sharebase (Britton Lee), both of whom manufacture their own hardware. These vendors have designed hardware architectures optimized to support the database function. In another approach, several software database vendors have projects to implement their database servers on the special purpose hardware of third party vendors. Of course, there are also the efforts of hardware vendors (such as IBM and DEC) to supply native database software. In each case, the database software is partitioned into a frontend portion (host or client software) and a backend portion (the database engine or server). These partitions are then implemented on separate pieces of hardware, requiring some form of communications hardware and software.

The user of a database application must decide whether or not there are systems requirements that preclude the use of a particular solution. For example, transactions rates, load, and throughput requirements will be familiar to most readers. However, database I/O, concurrency control, transaction management and recovery, join mechanisms, indexing mechanisms, search and sort algorithms, performance tuning parameters, administration tools, and many other factors can have a pronounced affect on both the short term and the long term success of the application. If the application is sufficiently simple - it does not involve large numbers of uses, large numbers of tables, particularly deep or wide tables, complex transactions, high availability requirements, flexibility, rapid implementation, or high transaction rates - then the user can make a selection based on whether or not the product meets general functional requirements, ease of use, or price.

As soon as one attempts to implement production applications or integrated database applications, things are rarely so simple. The matter is further complicated by non-technical issues. There may be corporate selection guidelines or even mandates which must be met. These can be either technically or non-technically motivated. In some cases, senior management may have non-technical reasons for mandating that a specific product be selected. It is sometimes the case that the selection is motivated by an attempt to force the application to fail, whether to motivate a non-relational solution or to maintain the status quo or to manipulate control of corporate responsibilities. It is not uncommon for a selection to be made on the basis of perceived popularity of a vendor rather than knowledge of a product's technical capabilities.

Applications written for a database server of any kind must take into account the nature of the communications software and hardware. Not only is there an overhead, but some kinds of transaction management and recovery become more difficult as well. The overhead can typically be modeled in two parts. First, each transaction will have some initialization and termination overhead. Second, there will be a communications cost which is related to the number of bytes to be transferred.

With a database machine, this performance overhead is not likely to disappear. However, because one end of the communications link is under the control of the database machine hardware, the vendor can optimize and perhaps standardize one half of the communications link. This leads to a more homogenous network environment, even when multiple heterogeneous host computers (clients) are served by the machine. If the vendor supports the native communications protocol of the host computer (e.g. DECnet on DEC machines), the advantage of hardware can be significant in that environment. In terms of support, the vendor need support, for each protocol implemented, one kind of link per client platform.

Software database vendors, on the other hand, must make a choice between supporting the running of the server on a single hardware platform or multiple platforms while simultaneously supporting either heterogeneous or homogenous client platforms. This means that the kinds of possible links that the vendor must strive to support are, for each protocol implemented, where ***m*** is the number of platforms, the sum of ***n*** as ***n*** goes from ***1*** to ***m***.

The problems associated with network support do not differ between database servers and machines. Network administration and recovery is difficult in any environment. Database network environment can be divided into two classes, those that support close coupling with the host and those that are loosely coupled. In a close-coupled environment, the link between host and database is a high-speed data highway, finely tuned for communications between host and database server or machine. The loosely coupled environment provides a standard linkage such as Ethernet.

With a distributed database environment, the problems of network administration simply compound the problems of database administration. If a node drops from the network, will the local database administrator be allowed to change the catalog? If not, then how is autonomy to be implemented? And if such changes are allowed, then how are differences in the schemas to be reintegrated in the system catalog? If a network performance problem occurs, how will the nature of the problem be identified? Network tuning is a fine art. Even if a shop has a network guru on board, it is unlikely that this rare breed also understands the nature distributed relational database algorithms, distributed concurrency control, and distributed transaction management and recovery.

Most of these problems can be minimized with a shared, integrated database environment, without any lose of functionality. Most production environments have no need of distributed transactions. Network communications is sufficiently advanced that updating a shared database represents minimal cost. Once the shared database has been updated, the performance problems associated with updates which are qualified by distributed joins or distributed selects do not exist. If the shared database server or machine is a multiprocessor unit, the advantages of multiprocessing the database request are still available.

In addition, the administration of a shared database environment is greatly simplified over that of a distributed database environment.

The ad-hoc environment can make use of distributed data and servers, but not without penalty. The problems of administrating such a system so as to insure data integrity are many.

Database machines such as those provided by Sharebase (formerly Britton Lee) were designed to provide optimum performance in an ad-hoc request environment. Because database machines represent a database vendor controlled environment, certain features can be implemented more readily and more reliably in the database machine. This means that the system is tuned for decision support applications requiring a high degree of concurrency. The system is also designed to provide high availability. This means that applications which require almost continuous on-line database support can advantageously be implemented on such a database machine. The ability to perform on-line database backups while allowing both readers and writers to have access to the database aids in this effort. In addition, on-line disk mirroring (shadowing) and recovery become possible. The successful uses of a Sharebase database machine tend to take advantage of these features and the transactions involved frequently affect numerous tables non-procedurally. Updates are often qualified based upon multi-table joins and selects without sacrificing performance or concurrency, compared to other relational database products.

Database machines such as those provided by Teradata were designed to provide optimum performance in a more traditional transaction environment. This means that extremely high transaction rates are possible, especially if the transactions involve few records. If the tables in the database can be indexed by a hash key, then Teradata's multiple processors and unique bus architecture can distribute the load. Existing non-relational applications needing high transaction rates and using record-at-a-time processing can quickly achieve these results while moving to a relational environment. In this way, the application acquires the benefits of relational database administration and ad-hoc inquiry without sacrificing existing performance.

These two vendors have been able to implement a number of applications that would have been difficult otherwise.