Data Warehouse Architectural Design
(Using Parallel Processing on Distributed Environment
For Effective Load Balancing And Data Sharing)

Anshu Saluja
Lecturer-IT GNIM, Affiliated to GGSIP University, Delhi. E-Mail: saluja_anshu@yahoo.co.in

ABSTRACT
Today’s business environment is information reliant. Data warehousing is an efficient and competent way to manage demand for lots of information from wide variety of users. Data warehousing provides the capability to analyze large amounts of historical data for nuggets of wisdom that can provide an organization with competitive advantage. The structure of data warehouses is easier for end users to navigate, understand and query against unlike the relational databases primarily designed to handle lots of day to day transactions. There is no theoretical limit to how much data can be placed in data warehouse. If the volume of data inside the warehouse begins to exceed the limit of single processor, then another processor could be added to the network. As the need for information and the possibility of what can be done with it grows, and businesses expand their information systems beyond organizational and geographic boundaries, it is increasingly important that information be dispersed to various locations in order to have better performance., higher reliability, increased organizational control, smooth incremental growth and greater levels of independence over the centralized system. This paper examines how “architecture of distributed data warehouse” can be integrated with the use of parallel processing. With the increase of volume of data stored in Data Warehouse, the traditional architectures need high performance in terms of processors and input and output systems. We, now focus on environments with multiple data warehouses that are possibly derived from other data warehouses. This kind of problem is well characterized when it is required to perform a high complex query. Using distributed environment, is a good choice to increase the performance and to reduce cost.

KEYWORDS
DATA BASES, DATA WAREHOUSE, PARALLEL PROCESSING, DISTRIBUTED DATA, AGGREGATED DATA, MULTITHREADED

INTRODUCTION
The data warehouse environment holds lot of data. Data volume continues to grow; the database size may rapidly outgrow a single server. Therefore, one of the biggest challenges when designing architecture of data warehouse which includes the placement and distribution strategy. It becomes necessary to know how the data should be divided across multiple servers, and which users should get access to which type of data. Logically there is a single data warehouse, but physically there are many data warehouses that are all tightly related but reside on separate processors i.e. technically distributed data warehouse. A distribution center acts as the hub for all merchandise, sending products to individual retail stores for customer consumption. Similarly, a distributed DW, the nucleus of all enterprise data, sends relevant data to individual data marts from which users can access information for order management, customer billing, sales analysis, and other reporting and analytic functions. The data placement and distribution design should consider several options, including data distribution by various subjects for example by area, by location or by time etc.

PERFORMANCE CONCERN IN DATA WAREHOUSE
A complaint that is heard too often from many data warehouse users from newly implemented data warehouses is that data access performance is slow. In the world of fast client-server applications and ever-increasing CPU speeds, users have come to expect rapid results from virtually any query. A large visible data warehouse effort must properly set user expectations for performance. But achieving proper performance with a large diverse data set is not a negligible matter. Performance of the data warehouse is based on various factors. The rapid query processing is the highly desired feature that should be designed into the data warehouse. Of course, the actual performance levels are business dependent and vary widely from one environment to another. Also, unpredictable usage patterns against the data vary the performance of the entire system. Therefore, it is difficult to predict the performance of a typical data warehouse. Other factors that can improve performance of the data warehouse could be the physical data distribution of the data, aggregating data, and use of parallel processing. This paper broadly talks of architectural aspect of the data warehouse which includes the use of parallel processing in the warehouse on the distributed environment.

REVIEWING PHYSICAL DATA DISTRIBUTION
The physical distribution of the data on the disk can impact performance to a great extent. Decision support applications that access data warehouse data often scan a large number of records. For a particular table these scans are often referred to as full table scans. A full table scan can benefit from contiguous data, data pages (collections of table records) that
reside physically side-by-side on the disk. A table that contains contiguous pages is easier to scan than a table that contains data pages that are fragmented or distributed in various non-contiguous locations on the disk. A table that is fragmented would require more i/o's to read than a table that is contiguous. And more i/o's means that data access will be slower than for the same amount of data collected with fewer i/o's.

AGGREGATING DATA TO IMPROVE PERFORMANCE

Parallel processing can have a significant impact on the performance of a query. Data warehouses generally have large fact tables (contains the business metrics or measurements) with a number of smaller dimension tables (contains business dimensions; hierarchies within each dimension table are used for drilling down to lower levels of data). The fact table is often scanned to perform various binary operations such as joins etc. Aggregates are precalculated summaries derived from the granular fact table. These summaries form a set of separate aggregate fact tables. Aggregate fact tables or summary tables could have a significant impact on performance levels. Aggregation operations performed on the fact table could also benefit from parallel processing, with parallel threads of execution performing aggregations as data is fed from scan operations. Decision support queries must often aggregate data within fact tables. These aggregate operations are usually performed across dimensions, for instance aggregating daily sales for each quarter for each sales region. If these queries are run consistently it is usually possible to aggregate this data before the query is run and store the aggregations in tables within the database. When the query is run, it will recognize these aggregate tables and use them to compute its results, thus avoiding the processing necessary to create the aggregations. A sophisticated aggregation scheme would recognize dimensional hierarchies and build higher-level aggregations from more granular aggregations. For instance, a weekly aggregation of sales for a region could be used to build a monthly or quarterly aggregation of sales by region, and thereby avoid an aggregation of the more granular daily sales totals. Such aggregate-awareness can significantly improve performance but requires meta-data that recognizes dimensional hierarchies.

TAKING ADVANTAGE OF PARALLEL PROCESSING AND MULTI-THREADING

All major relational database vendors support some form of internal parallel processing. Parallel processing divides a large task into many smaller tasks and executes the smaller tasks concurrently on several nodes. In parallel server on a symmetric multiprocessor one can assign more CPU power to the tasks depending on how your CPUs are partitioned. Each independent task executes immediately on its own processor; no wait time is involved. A parallel processing system has the following characteristics:

- Each processor in a system can perform tasks concurrently and tasks may need to be synchronized
- Nodes at different sites usually share resources, such as data, disks, and other devices
- Threads are lightweight processes. They improve performance by weakening the process abstraction. A process is one thread of control executing one program in one address space. A thread may have multiple threads of control running different parts of a program in one address space. Multithreading allows different parts of a single program to run concurrently. Without multi-threading, a task is processed sequentially. In general, when working with large amounts of data, multi-threaded processing is faster than sequential processing because a task is split into partitions that are processed in parallel. Multi-threading allows several internal threads of execution to process at the same time. These capabilities allow a database engine to perform multiple tasks concurrently to get improved performance.

DATA WAREHOUSING ARCHITECTURE ALTERNATIVES

Data warehousing architecture is needed to accommodate dynamic business requirements that cannot be anticipated. The lack of fit between organizational form and architecture has resulted in the subject for which data warehousing has become famous. These are avoidable by aligning a concrete structure. Firms that are highly centralized in geography and governance should pursue centralized data warehouse architecture to reap the greatest operational efficiencies and business benefits. Firms that are highly decentralized will prefer a distributed architecture; those with a mixed organizational pattern should implement a federated one. For over a decade, discussion and even controversy have lingered about which is the best distributed data warehouse architecture. The cost and time required to implement the different architectures would vary. Larger domain and size of these warehouses requires cost and complexity in implementing them. The architecture also requires a considerable commitment to up-front planning, which takes time and money.

A FORMAL APPROACH TO THE DESIGN OF DATA WAREHOUSE ARCHITECTURE USING PARALLEL PROCESSING ON A DISTRIBUTED ENVIRONMENT

The size of data warehouse rapidly approaches the point where the search for better performance and scalability becomes an obvious demand. Operating in limited resource environment under the control of operating system, the DBMS designers follow different strategies to achieve server efficiency in spite of voluminous data in the warehouse. Data warehouse requirements steadily pushing the limits of traditional RDBMS technology, the task of selecting an appropriate DBMS becomes even more critical. A variety of architectures allow multiple computers to share access to data, software, or peripheral devices. A considerably more efficient approach is to launch a separate thread for each separate task. Such lightweight task can be controlled by DBMS server rather than by operating system. Threads performed concurrent tasks; a true multi-threaded
architecture provides a high degree of resource sharing and tends to make system performance more stable. Multithreaded environment would not only suffice; for effective load balancing and data sharing multiprocessor systems are considered necessary. Theses are designed to handle a large number of client requests efficiently. Resource sharing can be achieved more efficiently and naturally between threads of a process than between processes because all threads of a process share the same address space.

A parallel server is designed to take advantage of such architectures by running multiple instances that "share" a single physical database. In client applications, a parallel server allows access to a single database by users on multiple machines with increased performance in terms of speedup and improved scale up to process larger workloads. This includes the data distribution in the data warehouse across multiple servers. Some criteria must be established for dividing it over the servers—by geography, time, function etc. Another factor in determining how data should be divided among multiple data warehouse servers is the pattern of usage, such as what data is accessed, and any binary operations such as joins etc. that take place in the query processing. The binary operations such as joins are the most expensive operations of the database systems, and therefore it is recommended to reduce the size of operands of such binary operations. This criterion is even more critical in case of data being distributed over multiple servers because data exists in database servers are logically correlated. In the multithreaded approach, the inter query parallelism is practicable, in which different threads handle multiple requests from multiple servers at the same time. The parallel processing offers the solution to the traditional problem of poor performance for complex queries and for very large databases. Hence, accessing and processing portions of the database by individual threads in parallel over databases can greatly improve the performance. This is especially crucial for data warehousing supporting complex decision support systems. Therefore, architecture having both multisserver and multithreaded environment leads to the concept of technologically and logically distributed data warehouse across multiple sites over the physical interconnection network.

This technically distributed architecture could be viewed as truly technically distributed memory architecture (see Fig 1) in which each node has its own local memory and disks, all CPUs are connected to each other via network mechanism and communicate by passing messages. An interconnection network can be high speed, “external LAN” or high speed switch. It is the powerful way to the successful and technological very large data warehouses. Each node is itself a powerful system having autonomous capability and the number of processor nodes is limited only by the hardware platform limitation. However, it is difficult to implement because of certain reasons—like it requires new programming paradigm, new operating system, new enhanced translators etc.

Relatively simpler approach based on the same concept could be technically distributed memory shared disk architecture implements the concept of shared ownership of the entire database between RDBMSs servers, each of which is running on a node of a distributed memory system. Multiple processors share secondary storage but each has their own primary memory. [6] This distribution is relevant from application point of view; this is why it is considered as distributed databases.

**Fig1. Technically Distributed Data Warehouse architecture.**

(Based upon Distributed Memory, multi server and multithreaded concept)

The various aspects, which characterize the structure of the computation and of the communication protocol of a distributed transaction, is further the substance of converse as it also have many different architectural schemes.

**COMPARISON OF DIVERSE DISTRIBUTED DATA WAREHOUSE ARCHITECTURE – RELATIVE ADVANTAGES AND DISADVANTAGES**

The basic advantage of technically distributed data warehouse architecture is that the entry cost is not expensive and also when the volume of data inside the data warehouse begins to exceed the limit of a single processor, then another processor can be added to the network and the progression of adding data continues in an unimpeded fashion. Benefits incur in designing above discussed distributed data warehouse architecture:

- Supports multiple, dispersed business divisions over geographical network
- Allocate each business division to have its own extraction, transformation and loading tool thereby reducing the overhead of the one server.
- The central data warehouse stores corporate-wide data
In parallel server environment, each independent task executes immediately on its own processor; no wait time is involved.

More Users: Parallel database technology can make it possible to overcome memory limits, enabling a single system to serve thousands of users.

Parallel processing can benefit certain types of applications by providing: **Enhanced Throughput**: Scale up; **Improved Response Time**: Speed up. With more CPUs available to an application, higher speedup and scale up can be attained.

Graceful degradation of the system[^5]: Multiple databases can provide greater availability than a single instance accessing a single database, because an instance failure in a distributed database system does not prevent access to data in the other databases: only the database owned by the failed instance is inaccessible.

Data distribution solves a number of problems, it may also create a few limitations of its own; for example, if the warehouse servers are distributed across multiple locations, a query that spans several servers across the LAN or WAN may flood the network with a large amount of data. Therefore, any distribution strategy should take into account all possible access needs for the warehouse data. Therefore, considering the different distribution strategies are very important.

Risks implicated in implementing distributed Data warehouse:

- Central coordination and management is required, based on access to a global metadata repository
- Complex environment, higher development cost and risk
- Network issues: transmission of data is one of the major components of the costs and delays associated with query execution.

In terms of information and system quality and individual and organizational impacts, no single architecture is dominant.

**CONCLUSION**

This paper has identified the primary cause of poor performance in data warehouse queries and discussed architectural solutions for managing and improving the performance. Distributed computing systems must focus essentially on the concept of parallel processing in which multiple processors are interconnected by a communication network in which every site has self-sufficient processing capability and is capable of performing its local task and must be participating in at least one of the global request for the global task. Implementation of parallel processing technology in the data warehouse is to break through the bottleneck and thereby improving the performance fundamentally. With parallelization, that delivery rate can also be substantially increased. It's presented a proposal here for structural design of distributed Data Warehouse, incorporated with the use of parallel processing.

**FUTURE SCOPE**

In such a large-scale environment, data updates from base sources may arrive in individual data warehouses in different orders, thus resulting in inconsistent data warehouse extents. This will require to achieve consistency across all data warehouses with an introduction of different architectures, techniques and approaches that should exists to be dealt with a data warehouse that is technically distributed over multiple servers. Further, there could be many other causes for the poor performance in the data warehouse queries which need to be resolved. Several commercial distributed computing systems are already implemented but they require further enhancement for better performance, higher reliability and higher degree of autonomy.

Furthermore, other issues related in implementing distributed data warehouse:

- Most ETL tools do not support a geographically distributed ETL environment with a global metadata repository
- Expensive, complex ETL product

**REFERENCES**