SURVEY OF PERFORMANCE ISSUES IN PARALLEL DATABASE SYSTEMS*

Ameet S. Talwadker
Department of Computer and Information Science
University of Mississippi
University, MS
(662) 915 - 7310
ameet@cs.olemiss.edu

ABSTRACT

Parallel database systems are being used nowadays in a wide variety of systems, right from database applications to decision support systems. These implementations involve database processing and querying over parallel systems. For the parallel databases to be effective and efficient, various optimizing solutions need to be implemented. These solutions deal with various issues associated with such database systems : query optimization, data allocation within the database, etc.

This paper will attempt to give the reader an understanding of the various issues involved in optimizing the performance of a parallel database system. The primary areas of focus will be system architecture, query optimization and data reallocation. It will look at related work done by other authors in this field and hopefully provide a thorough understanding of the solutions in this area.

KEYWORDS

Parallel database systems, query optimization, data reallocation, system architecture, survey

---

INTRODUCTION

The development of database management systems has coincided with significant developments in distributed computing and processing technologies. The merging of these two resources has resulted in the emergence of parallel database management systems. These systems have become the dominant data-management tools for highly data-intensive applications.

A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system is then defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users.

A parallel database system can be defined as a database management system implemented on a tightly coupled multiprocessor. An important distinction between a parallel DBMS and a distributed DBMS is that distributed DBMSs assume loose interconnection between processors that have their own operating systems and operate independently. Parallel DBMSs exploit multi-processor architectures in order to build high-performance and high-availability database servers.

In the following sections we look at three main issues concerning parallel database systems performance : architecture, data placement and reallocation, and query optimization. In each section a brief explanation is provided about the various performance improvement possibilities and their summary.

ARCHITECTURE

Parallel database system architectures range between two extremes : the shared- nothing and the shared memory architectures. In the shared nothing architecture, each processor has its own main memory and disk units. In shared memory architectures, every processor has access to any memory module or disk unit through a high speed interconnection [1] .Based on these two system architectures, other modified architectures are available for use. One of them is the shared disk architecture. In this architecture, processors have their own individual main memory space but they share a common global disk storage. The shared disk could however become a bottleneck for limiting scalability, especially during large data retrieval.

As mentioned above, the processors in a shared- nothing architecture have their own memory and disks. Nodes communicate by passing messages to one another through an interconnection network. Shared-nothing architectures can incorporate hundreds of processors within a single system and this is one of its main advantages. However, its dependence on an interconnection network for message passing can be a cumbersome issue. Also, a high number of processors within a single system makes it imperative that a good load-balancing strategy be in place. Also, if a processor fails then access to data owned by that processor is lost. A solution to this is a shared-disk architecture where every processor can read and write to any of the disks in the system but manages its own memory [2] .

The shared memory architecture has zero communication cost : data and messages are exchanged through the shared memory area. Advantages of such an architecture include easier process synchronization and minimal overhead for load balancing correction. But this architecture can be scaled effectively up to 40 processors. Beyond that interference affects the rate at which memory is processed.

The shared-nothing architecture is currently the more popular implementation, mainly due to its high reliability and ease of scalability to hundreds of processors. It is also the most cost effective means of constructing a parallel database system as it can be built from existing sequential machines using a simple interconnection network [3].

DATA PLACEMENT

In parallel database systems, the database can be physically distributed across data sites by fragmenting and replicating the data. Given a relational database schema, fragmentation subdivides each relation into horizontal (by a selection operation) or vertical (by a projection operation) partitions. Fragmentation is desirable because it reduces the size of relations involved in user queries. Based on the user access patterns, each of the fragments may also be replicated [4] .

Data placement strategies act as a significant aid in improving the performance of a parallel database system. Partitioning and distributing the data allows for the use of improved data processing techniques so as to allow each data section to be processed individually. Similarly data replication in a parallel database system can act as a backup strategy as well as provide an opportunity for carrying out separate processing on each individual data set.

These described strategies are static methods i.e. they are carried out at periodic intervals when data is not being processed. Likewise, dynamic data placement techniques are available which rearrange the data when data processing algorithms are also executing on the data. This is turn takes care of data access skew by carrying out periodic data redistribution. Such a dynamic data allocation method identifies data fragments which are accessed the most (so called hot nodes) and moves data from these nodes onto other segments so that there is even data access throughout the system, thereby reducing the chances of system failure through processor overburdening [5] .

The effect of these data placement techniques, either static or dynamic, is positive as throughput is improved with re-allocation. Both reallocation types show a significant improvement in database processing if they are supported with multiprogramming techniques. At lower levels of multiprogramming, the advantage of dynamic reallocation is not substantial due to the fact that the resources outweigh the demands for information from each individual node. At a higher level of multiprogramming reallocation is not very effective as the gain in performance is offset by excessive data transfer and transaction restart. Thus the conclusion is that reallocation is observed to improve throughput but not always significantly as might be naturally assumed.

QUERY OPTIMIZATION

In parallel database systems, query processing and optimization techniques have to address difficulties arising from the fragmentation and distribution of data. To deal with fragmentation, data localization techniques are used where an algebraic query, which is specified on global relations, is transformed into one that operates on fragments rather than global relations. In the process, opportunities for parallel execution are identified and unnecessary work is eliminated. Localization requires the optimization of global operations, which is undertaken as part of global query optimization, This in turn involves permuting the order of operations in a query, determining the execution sites for various distributed operations and identifying the best distributed execution algorithm for distributed operations [6] .

Query execution techniques can be classified into two forms of parallelisms:

i) Intra-operator parallelism - One operation is parallelized over several processors. This can be achieved by partitioning the data among the processors. Each processors then carries out the query on its own data set and the results are then combined for the eventual result.

ii) Inter-operator parallelism - Here, several processes are executes simultaneously, each processor carrying out a process. There are two forms of inter-operator parallelisms : independent and pipelined [7] [8] .

Various implementations exist for achieving intra-operator parallelism. An approach known as de-clustering can be used to partition the query into fragments which can then be allocated to the processors. This approach is most useful when optimizing complex queries. Various sections of the queries can be broken and each part can be allocated to a separate processor. The allocation of the various query subparts can also be done such that those sharing the same data set can be allocated to adjacent or nearby processors. This helps in reducing the costs associated with moving the query results from one stage to another [9].

CONCLUSION

Considering the foregoing sections, it is clear that for a parallel database system to perform at an optimum level, various operational and non-operational aspects need to be given careful consideration. In this regard, architectural design, data placement and query optimization can be considered the main factors. Besides these, other factors also exist which have due weightage in determining the performance of the parallel database system.

BIBLIOGRAPHY

1. M. Tamer Özsu, Patrick Valduriez, "Distributed and Parallel Database Systems", *ACM Computing Surveys*, 1996

2. Abdelsalam Helal, David Yuan, Hesham El-Rewini, "Dynamic Data Reallocation for Skew Management in Shared-Nothing Parallel Databases", *Distributed and Parallel Databases*, Volume 5, Number 3, July 1997

3. Manish Mehta, David J. DeWitt, "Data Placement in Shared-nothing Parallel Database Systems", *The VLDB Journal*, 1997

4. M. Farrukh Khan, Ray Paul, Ishfaq Ahmad, Arif Ghafoor, "Intensive Data Management in Parallel Systems: A Survey", *Distributed and Parallel Databases*, Volume 7, Number 4, November 1997

5. Erhard Rahm and Robert Marek, "Dynamic Multi-Resource Load Balancing in Parallel Database Systems", *Proceedings of the 21st VLDB conference*, 1995

6. Stefan Manegold, Johann K. Obermaier, Florian Waas, "Load Balanced Query Evaluation in Shared-Everything Environments",*Proceedings of European Conference on Parallel Processing*, 1997

7. Anant Jhingran, S. Padmanabhan and A. Shatdal, "Join Query Optimization in Parallel Database Systems", *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, 1993.

8. Jaideep Srivastava and Gary Elsesser, "Optimizing Multi-Join Queries in Parallel Relational Databases", *Proceedings of the Second International Conference of Parallel and Distributed Information Systems*, December 1993

9. Rajeev Raman and Uzi Vishkin, "Parallel Algorithms for Database Operations and Database Operation for Parallel Algorithms", *Proceedings of the 9th International Parallel Processing Symposium*, 1995

10. Luc Bouganim, Daniela Florescu, Patrick Valduriez, "Dynamic Load Balancing in Hierarchical Parallel Database Systems", *The VLDB Journal*, 1996

11. Soumen Chakrabarti S. Muthukrishnan, "Resource scheduling for parallel database and scientific applications", *ACM Symposium on Parallel Algorithms and Architectures*, 1996