Query Processing and Optimization of Parallel Database System in Multi Processor Environments

Sukheja Deepak
PITM
Indore,INDIA
deepaksukheja@yahoo.com

Singh Umesh Kumar
ICS,
Vikaram University,Ujjain.
umeshsingh@rediffmail.com

Mishra Durgesh
SAIT
Indore INDIA
drdurgeshmishra@gmail.com

Pandya Bhupendra K.
ICS,
Vikaram University,Ujjain
Bhupendra20pandya@yahoo.co.in

***Abstract* - In present scenario parallel database systems are being applicable in a broad range of systems, right from database applications (OLTP) server to decision support systems (OLAP) server. These developments involve database processing and querying over parallel systems. A means to the success of parallel database systems, particularly in decision-support applications (Data warehousing), is *parallel query optimization.* Given a SQL query, parallel query optimization has the goal of finding a parallel plan that delivers the query result in minimal time. Various useful and competent, optimizing solutions to be implemented For the parallel databases. Parallel DBS attempt to develop recently in order to make high-performance and high-availability database servers at a much lower price for multiprocessor computer architectures than mainframe computers. The objective of this paper is define a novel approach on how to achieve parallelism for relational database multithreaded query execution use to maximum resource utilization of CPU and memory. This technique offer a solution to the problem of minimizing the response time of input queries against parallel databases.**

Keywords - parallel database, Query Processing and Optimization, multithreaded query execution

I. INTRODUCTION

Query one of the key reasons for the success of RDBMS technology is the use of declarative languages and query optimization. In any database the user can write a query for what data needs or to be retrieved and the database takes over the task of finding the data through efficient technique. It is the job of the query optimizer, optimization process deals with the efficiency of the query to evaluate other alternative methods of executing a query, and selecting the best execution method. The advance database systems have coincided with noteworthy developments in distributed computing and processing technologies. The inclusion of the following two properties has resulted in the emergence of parallel database systems. Parallel database systems combine data management and parallel processing techniques to provide high performance, high-availability and scalability for data intensive applications [1, 2].

According to [18,19,20], An ideal approach on a parallel machine would scale easily at a low-cost, and would demonstrate linear speedup and scale up for number of processors. Mainly two architectures have emerged from the quest for an ideal parallel machine; they are shared-memory (SM) and shared-nothing (SN).

In the shared memory architecture shown in fig. 1, each processor has access to all disks in the system and to a global shared memory. The main advantage of a shared-memory system is that it has no data transmition cost; processors exchange messages and data through the shared memory. This also makes it easier to synchronize processes. Another advantage of this architecture machine is that load imbalance can be corrected with minimal overhead. The weakness of such a system is the difficult to manage problem of conflicts. If the number of processors is less then Shared memory architecture can be scaled.

Shared-nothing architecture shown in fig. 2 is the just opposite of Shared memory. Shared-nothing architecture machine works on the principle of ownership, that each processor/node in a cluster has individual ownership of the data on that node. The main advantage of shared-nothing is its ability to scale to hundreds and potentially thousands of processors. The drawbacks of such a system are complications created by load-imbalance and its dependence on a high bandwidth interconnection network for effective message exchange. Shared-nothing architecture designed mainly due to its high reliability and ease of scalability to hundreds of processors.

According to Patrick Valduriez in [17], parallel database system designers strive to develop software oriented solutions in order to exploit multiprocessor hardware. The objectives of parallel database systems can be achieved by extending distributed database technology, for example, by partitioning the database across multiple (small) disks so that much inter- and intraquery parallelism can be obtained. This can lead to significant

978-0-7695-4730-5/12 $26.00 © 2012 IEEE
DOI 10.1109/AMS.2012.49

improvements in both response time and throughput (number of transactions per second).

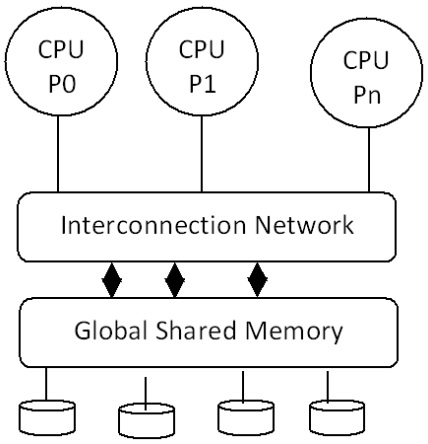

Figure 1 Shared memory machine

II. RELATED WORK

Over the last two decades parallel query processing in database systems was the topic of extensive research. Its outcome is now undoubtedly in daily use as part of major commercial DBMS. Most of the research was concentrated on shared nothing (SN) architectures, e.g. the research prototypes Gamma [4] and Bubba [5]. It is well-designed for modern grid and cluster computing and the focal point of other approaches on symmetric multiprocessing architectures (SMP), such as XPRS and Volcano [5, 6, and 7]. While many other scholars has researched algorithm for database optimization based on CMP. But the mainly of their work were focused on the optimization of join operation considering the L2-cache and the parallel buffers of the shared main memory [8, 9, and 10].

III. PARALLEL QUERY EXECUTION

Commercial database technology has encouraged from the earlier hierarchical and network models to the relational model. The main advantages of relational database systems (RDBMSs) over their predecessors are data independence and high-level query languages (e.g., SQL). These advantages boost programmer efficiency and favor routine optimization. Additionally, the set-oriented nature of the relational model facilitates distributed database management [12]. Today, after a decade of optimization and tuning, RDBMSs can provide a performance level reaching that of earlier systems. Therefore, they are being extensively used in commercial data processing for decision-support or online transaction processing (OLTP) applications. Parallel processing exploits multiprocessor computers to run application programs by using several processors cooperatively, in order to improve performance. Its prominent use is in scientific computing by improving the response time of numerical applications [13].

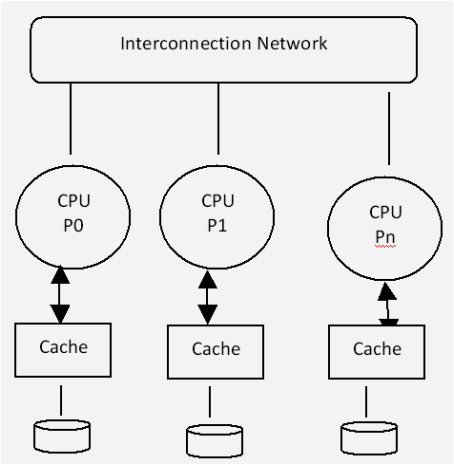

Figure 2 Shared nothing machine

The combination of database management and parallel processing is exemplified by the advances in parallel database systems [3]. In parallel database system PQO allows one to break-up a given SQL statement so that its parts can run simultaneously on different processors in a multi-processor machine. Typical operations that can run in parallel are: full table scans, sorts, sub-queries, data loading etc. Parallel Query allows one to break SELECT or DML statements into multiple smaller chunks and have PQ slaves execute those smaller chunks on separate CPU's in a single box shown in Fig 3 (parallel database optimization) in [21].

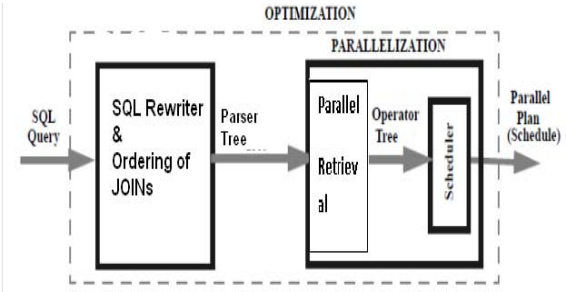

Figure 3

The first part of the Fig 3 is ordering and rewriting, produce a query tree that fixes aspects such as the order of joins and the strategy for computing each join and second part, parallelization convert the query tree into the parallel

plan. Parallelization also breaks this phase into two steps, parallelism extraction followed by scheduling.

IV. QUERY PROCESSING AND OPTIMIZATION APPROACH IN PARALLEL DATABASE

Parallel processing divides a large task into many smaller tasks, and executes the smaller tasks concurrently on several nodes/processor. As a result, the larger task completes more quickly. In sequential processing, independent tasks compete for a single resource. Only task 1 runs without having to wait. Task 2 must wait until task 1 has completed; task 3 must wait until tasks 1 and 2 have completed, and so on. In parallel processing, more CPU power is assigned to the tasks. Each independent task executes immediately on its own processor: no wait time is involved. A parallel database system should maintain the following characteristics: Concurrency Management, Task Synchronization, Resources sharing, Data Placement and Network Scaling. In which synchronization is a critical success factor. To synchronize the database in loosely coupled architecture / shared nothing architecture mostly use locking mechanism. Same mechanism also implement with tightly coupled architecture for data placement. In a parallel database system, proper data placement is essential for load balancing. Ideally, interference between concurrent parallel operations can be avoided by having each operation work on an independent dataset. These independent datasets can be obtained by the declustering (horizontal partitioning) of the relations based on hash function or range index and allocating each partition to a different memory module. The execution strategy that is proposed here as follows:

• First, the total amount of work that has to be done to evaluate the query is minimized.

• Then, try to distribute that minimal amount of work equally over the available processors.

First approach when memory module not shared.

1) Separate the all relations along with it predicates those are use in users query.
2) Assign the separate relation to separate processor / CPU along with its memory module. Processor allocation strategies also well define by researcher Xuemin Lin and Simon Fox in [15].
3) Transfer the intermediates resultset with predicates to other CPU to evaluate result.
4) Repeat step 3 till all predicates not evaluated.

The above four steps are only for retrieve / select statements. If the SQL statements update or insert statements then

1) Partitioning: distributing the tuples of a relation over several disks / memory modules which will help to allow parallel databases to exploit the I/O bandwidth of multiple disks by reading and writing them in parallel or Relations are declustered (partitioned horizontally) based on range index ,hash function methods.
2) Each CPU accesses the separate memory module and insert or update the data in the database in the accessible memory module. Note when the CPU will access the cluster of database, concurrency control protocols (lock mechanism) must be enable.
3) Combine the all memory module and transfer the data into the datafile.

Second approach when memory module shared (fully shared Memory Architecture).

1) Partitioning: distributing the tuples of a relation over several disks / memory modules which will help to allow parallel databases to exploit the I/O bandwidth of multiple memory module (any memory module access by any processor/CPU) by reading and writing them in parallel or Relations are declustered (partitioned horizontally) based on round robin, range index ,hash function methods.

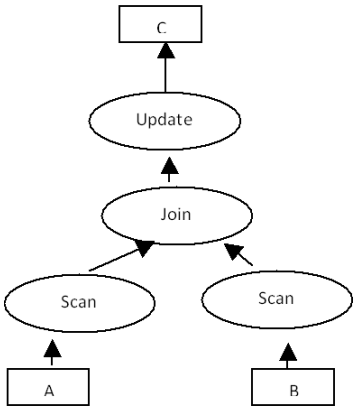

Figure 4

2) In case of select / retrieval statements apply the block level locking mechanism with read lock. So that one CPU access (retrieve) the data it should be lock for other CPU till the complete execution of SQL statement.
3) If the SQL statement belongs in update statement category then apply the record level locking mechanism with read and write lock. Once the record will update by CPU, with the help of cache coherence property update the data in all corresponding memory module.

Consider the following query:

```
Update t into C values
(select t
from A, B
where A.x = B.y);
```

Simple conventional plan is shown in Fig. 4. But when the same query execute in multiprocessor environment then the query execute in parallel manner

shown in Fig. 5 and response time will be minimize and system throughput increase dramatically.

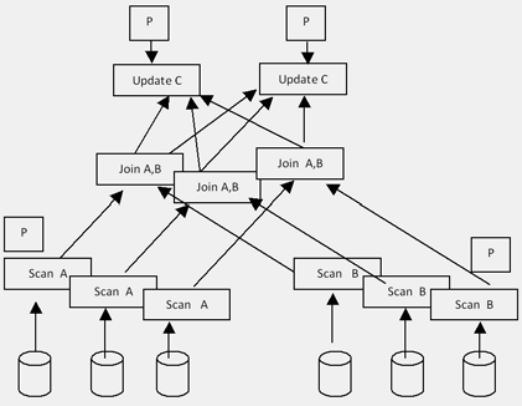

Figure 5

V. CONCLUSION

Parallel DBMSs have become a reality in the last few years. They provide the functionality of centralized DBMSs, but in a multiprocessor system. Parallel DBMSs are perhaps the only realistic approach to meet the performance requirements of a variety of important applications which place significant throughput demands on the DBMS. In order to meet these requirements parallel DBMSs need to be designed with special consideration for the protocols and strategies. This paper is providing an overview of these strategies.

REFERENCES

[1] T. Agerwala, J.L. Martin, J.H. Mirza, D.C. Sadler, D.M. Dias, and M. Snir. SP2 System Architecture. *IBMSystems Journal,* 34(2): 152-184, 1995.

[2] D.J. DeWitt and J. Gray. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM,* 35(6):85-98, June 1992.

[3] D.J. Dewitt and J. Gray, "Parallel database systems: the future of high performance database systems," *Comm. ACM,* vol. 35, no. 6, 1992.

[4] DeWitt, D.J., Ghandeharizadeh, S., Schneider, D.A., Bricker, A., Hsaio, H.I., Rasmussen, R.: The gamma database machine project. IEEE Transactions on Knowledge and Data Engineering 2(1), 44–62, 1990.

[5] Copeland, G., Alexander, W., Boughter, E., Keller, T.: Data Placement in Bubba. In: Proceedings of the 1988 ACM SIGMOD international conference on Management of data, Chicago, Illinois, USA, June 01-03, pp. 99–108, 1988.

[6] Stonebraker, M., Katz, R.H., Patterson, D.A., Ousterhout, J.K.: The Design of XPRS. In: Proceedings of the 14th International Conference on Very Large Data Bases, August 29-September 01, pp. 318–330, 1988.

[7] Graefe, G.: Encapsulation of parallelism in the volcano query processing system. In: SIGMOD 1990: Proceedings of the 1990 ACM SIGMOD international conference on Management of data, pp. 102–111. ACM Press, New York, 1990.

[8] Graefe, G., Cole, R.L., Davison, D.L., McKenna, W.J., Wolniewicz, R.H.: Extensible Query Optimization and Parallel Execution in Volcano. Morgan-Kaufman, San Mateo, 1994.

[9] Cieslewicz, J., Ross, K.A., Giannakakis, I.: Parallel buffer for chip multiprocessors. In: DaMoN, 2007.

[10] Cieslewicz, J., Ross, K.A.: Adaptive aggregation on chip multiprocessors. In: VLDB 2007.

[11] Hardavellas, N., Pandis, I., Johnson, R.: Database servers on chip multi processors limitations and opportunities. In: CIDR 2007.

[12] T. Ozsu and E Valduriez, "Distributed databases: where are we now?," *IEEE Comput.,* vol. 24, no. 8, 1991.

[13] J.A. Sharp, *An Introduction to Distributed and Parallel Processing,* Blackwell Scientific Publications: Oxford, 1987.

[14] D.J. Dewitt and J. Gray, "Parallel database systems: the future of high performance database systems," *Comm. ACM,* vol. 35, no. 6, 1992.

[15] Xuemin Lin and Simon Fox "An Effective Parallelization of Execution of Multijoins in Multiprocessor Systems", *IEEE Comput.,* vol. 24, no. 8, 1996.

[16] Ameet s. Talwadker, "Survey of performance issues in parallel database systems", Journal of Computing Sciences, Volume 18 Issue 6, June 2003.

[17]. Patrick Valduriez, "Parallel database systems: Open problems and new issues", SIGMOD Record, Vol. 25, No. 3, September 1996.

[18] Waqar Hasan, "OPTIMIZATION OF SQL QUERIES FOR PARALLEL MACHINES", A DISSERTATION OF DOCTOR OF PHILOSOPHY,1995.

[19] *Erhard Rahm, "Parallel Query Processing in Shared Disk Database Systems*", *ACM SIGMOD Record 22 (4), Dec. 1993.*

[20] David J. DeWitt Jim Gray, "Parallel Database Systems: The Future of Database Processing or a Passing Fad?", Technical Report 90.9.

[21] Waqar Hasan, Patrick Valduriez, "Open Issues in Parallel Query Optimization", SIGMOD Record, Vol. 25, No. 3, September 1996.

[22] Degn Ya-Dan, "Hash Join Query Optimization Based on Shared-Cache Chip Multi-Processor", Journal of Software 2010 volume 6, PP.1220-1232.

[23] .s. Sumathi, "Distributed and Parallel Database Management Systems Studies in Computational Intelligence, 2007, Volume 47, Fundamentals of Relational Database Management Systems, Pages 559-609.

[24] Sukheja Deepak, singh Umesh K. 'Design of shared-nothing cluster architecture for fast accessing and highly availability of data in heterogeneous database environment ", WICT 2011, PP 112-115.

[25] http:// mazsola.iit.unimiskolc.hu/tempus/discom/doc/db/tema05.pdf