Optimism analysis of parallel queries in databases through multicores

P.Mohankumar1, P.Kumaresan2 and Dr.J.Vaideeswaran3
SITE, VIT University, Vellore-14., TN, India
[1]Pmohankumar@Vit.ac.in [2]pkumaresan@vit.ac.in
[3]j_vaideeswaran@vit.ac.in

Abstract: *In the recent worlds computing applications it's the revolutionary challenging task for database people to maintain and process information in cyclic manner. Thus it turned the research people interest how the peculiar queries for particular application can be resolved in an elapsed time slice optimistically. Ultimately it's parallelizing the process. An optimistic analysis is made in this paper how to cover parallelism for database queries without suffering any loss and obtained desired response time.*

Keywords: *parallelism, multicores, monitor, optimism, partitioning.*

1. Introduction.

Apart form data management techniques such as data -ware housing and mining this paper focus mainly on parallelism optimization for peculiar queries for particular applications. Queries involved in day to day database application may of different types such as queries with multiple relations, queries with single relations and multiple operations, multiple queries with multiple relations. These queries consists usually many join (single and multiple) and aggregate operators. The existing Parallel processing methodologies that treat these kinds of queries are very sensitive to data distribution and involve more IO and Communication cost in processing. In order to overcome the issues an analysis is made how to cover parallelism on behalf on architectural basis. The paper organized as basic issues in parallelism query processing and multicores and how inter and intra operator parallelism is deployed (how the threads based parallel programming is used in query processing), how the parallelism is achieved with the desired time.

2. Related work.

Though various approaches has been made by the database experts each has some sort of lack in optimism as specified in the below table how these can be covered and how parallelism is achieved in an efficient manner is analyzed .ie how parallelism guarantees the multi relational operators and how scheduling can be generalized during execution without suffering any performance penalties. These tasks include IO-CPU bounded ness, tuple size and tuples per page and estimated execution time.

Table 1. Related works.

S.No	Concept	Architecture	Algorithm	Performance Analysis	Paper
1.	Join performed before Group by operations.	Shared disk	Aggregation partition and join partition method	Delay in response time	[2]
2.	Join performed after Group-by Operations	Shared nothing	Bulk synchronous parallel model. Histogram data structure.	Minimizes communication cost and avoids data skew for immediate results.	[3]
3.	Multiway join query executed using left deep pipeline(joins executed by separate processor)	Shared nothing	Polynomial time (static) (interleaving large class of joins)	Increased throughput.	[4]
4.	Inter and intra operator execution of a query	Multi cores	2-Phase algorithm 1.query processing 2.query optimization during execution	No conflicts in concurrency. Maximum throughput	[5]

3.Existing parallel database query processing and multicore issues.

It's mandatory to identify the basic issues in query processing and multicore.

3.1. Issues related to query processing.

a) How to partition data b)How to partition operations c)Each operation is partitioned across all nodes and get best sequential plan then parallelize scheduling and pipelining issues. d) Inter query parallelism.(query transaction execute in parallel with one another.) e)Intra query parallelism 1) Intra operator parallelism (parallelize each individual operation in query) 2) Inter operator parallelism (execute different operation in a query expression in parallel).Query Processing: activities involved in retrieving from the database typical steps when processing a high-level query (e.g SQL query*).Query tree: internal representation of the query .Execution strategy: how to retrieve results of query*

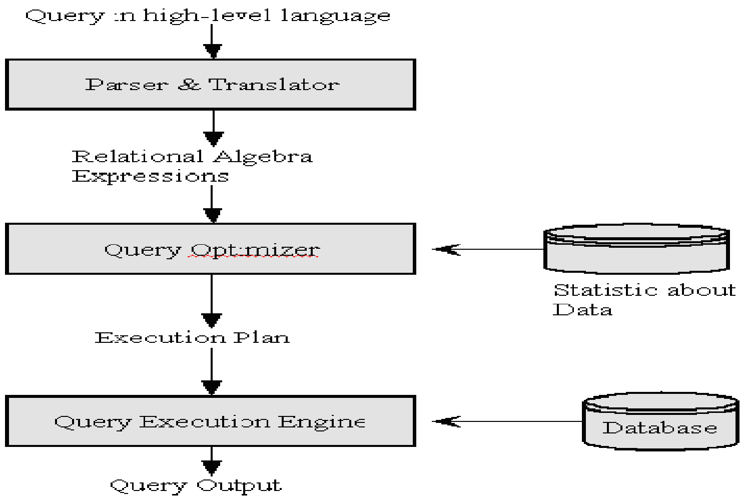

Figure1: Query processing model.

3.2 Issues related to multi-cores.

a) Firstly as integration of multi-core in whole query processing. b) Next finding the parallelism in query optimization based on evaluation plan selection and cost estimation.

4. Proposed work

The overall process can be viewed in two phase one optimism by DBMS optimizer and second dynamical refinement of query plan during runtime execution which guarantees thread rebalancing and resource utilization based on the query.DBMS optimizer drafts a plan on how a query should be carried out in parallel. This done by setting the database admin based on this server will perform the operation. If we consider the server as SQL, it can perform a query or index operation in parallel by using several operating system threads, the operation can be completed quickly and efficiently .During query optimization, SQL Server looks for queries or index operations that might benefit from parallel execution. For these queries, SQL Server inserts exchange operators into the query execution plan to prepare the query for parallel execution. An exchange operator is an operator in a query execution plan that provides process management, data redistribution, and flow control .After exchange operators are inserted, the result is a parallel-query execution plan. A parallel-query execution plan can use more than one thread. A serial execution plan, used by a nonparallel query, uses only one thread for its execution. The actual number of threads used by a parallel query is determined at query plan execution initialization and is determined by the complexity of the plan and the degree of parallelism. Degree of parallelism determines the maximum number of CPUs that are being used. The degree of parallelism value is set at the server level and can be modified by using the sp_configure( ) system stored procedure.

4.1 Base work.

Parallelism is deployed through multi-core processor an abstract programming model which contain the physical information about the data is created in order maintain the number caches and cores that's utilized in query execution.
A logical no existent component (Active_group) is created in order to group different cores and caches. During execution an operator must be, these can be explained in two phases. Distributed over one or more Active_groups.

Phase one. A data structure is created which is responsible for resource management like allocation, deallocation and data transfer between Active work groups and cores as wells for scheduling process .These can be implemented as new Asynchronous operator model (Active_group) is created which uses pipes and buffers responsible for data transfer. 1. Allocation of process 2.Assigning task to process 3.Transfer required data to the process. ***Phase two.*** A monitor program routine is created which is Responsible for managing the execution of query processing by query evaluation plan.

Proposed model for query processing.

Assigning task to Active_group.

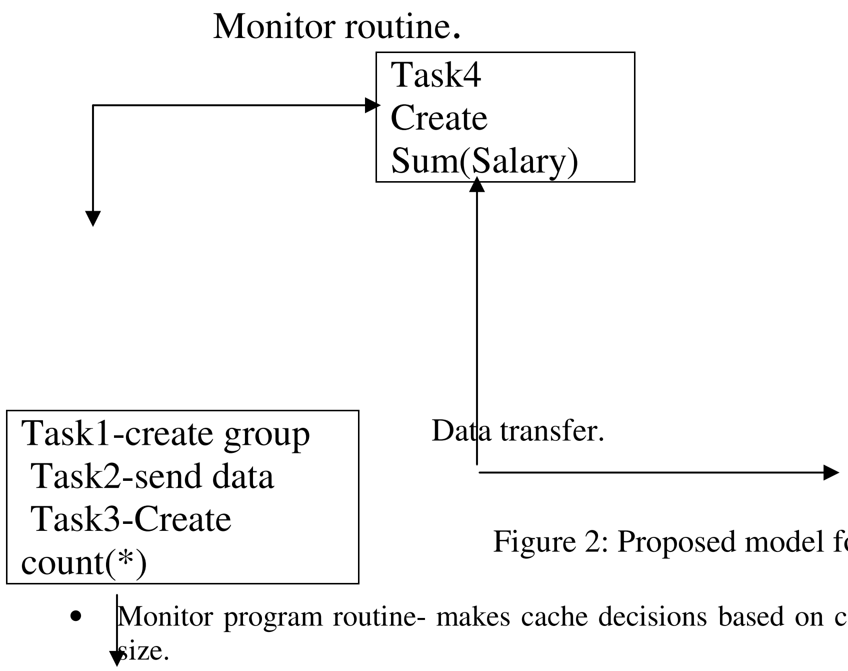

Figure 2: Proposed model for query process.

- Monitor program routine- makes cache decisions based on cost frequency of usage and size.
- Active group – nonexistent logical component to group different cores and caches.

Consider the query as example.
[Select e.dno, sum (salary)
from employee e group by e.dno] .

Let us assume a concern consists various departments. Each department located at different locations. In our example if management need to get the total salary to be issued first the employee table must be retrieved and then the department tables. Department wise the salary must be aggregated the result must be displayed. i.e.: aggregate operation is performed on required tuples stored in disk(results in tuples with partial sums at each processor) and result of local aggregation is portioned on group by attributes and aggregation is performed again at each processor to get the final result. In our model the given query execution will be as follows the active group will take care of creating task ,assigning task and transfer data between different locations and the monitor routine will take care of how the query gets executed and how the results are temporarily cached and how the final result is displayed without suffering any delay.ie how to minimize the total execution time of a workload by Caching intermediate/final results of individual queries, and using these cached results to answer later queries as shown in Figure:5.

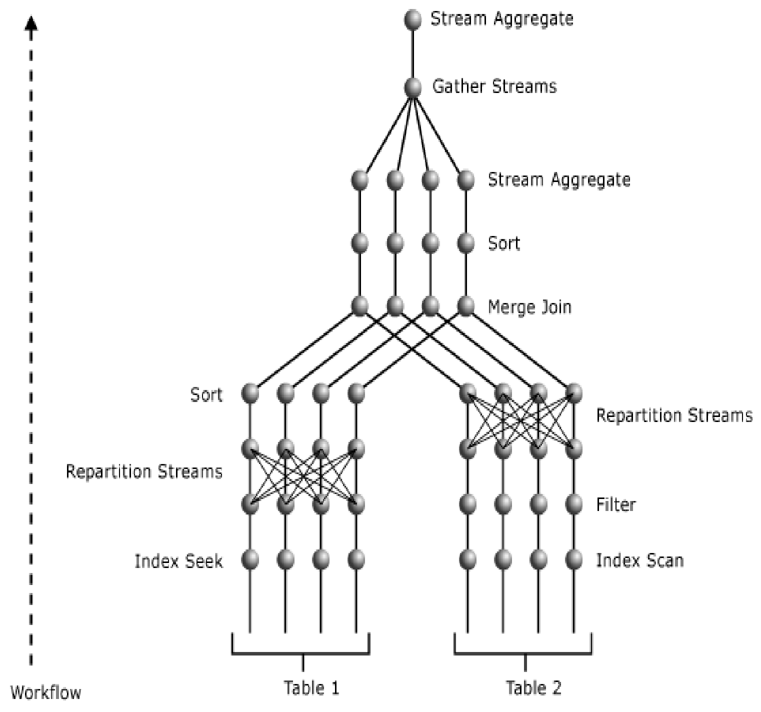

Figure3: Data scan representation during query processing

The illustration shows a query optimizer plan executed with a degree of parallelism equal to 4 and involving a two-table join.

a) A coordinating thread quickly and randomly scans the table to estimate the distribution of the index keys. The coordinating thread establishes the key boundaries that will create a number of key ranges equal to the degree of parallel operations, where each key range is estimated to cover similar numbers of rows. b) The coordinating thread dispatches a number of threads equal to the degree of parallel operations and waits for these threads to complete their work. Each thread scans the base table using a filter that retrieves only rows with key values within the range assigned to the thread. Each thread builds an index structure for the rows in its key range. In the case of a partitioned index, each thread builds a specified number of partitions. Partitions are not shared among threads c) After all the parallel threads have completed, the coordinating thread connects the index subunits into a single index. This phase applies only to offline index operations.

4.1.1 Inter-Operation parallelism and dynamic partitioning.(Parallelism between operations)

The execution plan implements a full scan of the table followed by a grouping the retrieved rows based on the value of the eno column each of the two operations (scan and sort) performed concurrently is given its own set of parallel execution servers. Therefore, both operations have parallelism. Parallelization of an individual operation where the same operation is performed on smaller sets of rows by parallel execution servers achieves what is termed intra-operation parallelism. When two operations run concurrently on different sets of parallel execution servers with data flowing from one operation into the other, we achieve what is termed inter-operation parallelism. Due to the nature of the server's operations, only two operations in a given tree need to be performed simultaneously to minimize execution time. This how is shown in Figure-4.

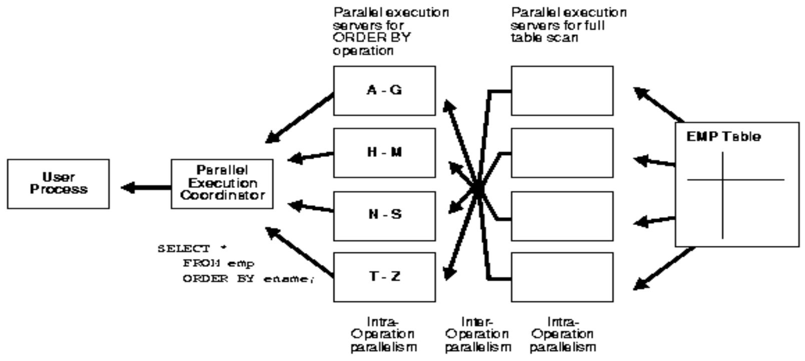
Parallel execution servers for ORDER BY operation
Parallel execution servers for full table scan
A - G
H - M
N - S
T - Z
EMP Table
User Process
Parallel Execution Coordinator
SELECT *
FROM emp
ORDER BY ename;
Intra-Operation parallelism
Inter-Operation parallelism
Intra-Operation parallelism

Figure 4: Inter and Intra operation parallelism.

The number of parallel execution servers associated with a single operation is known as the degree of parallelism. No more than two sets of parallel execution servers can run simultaneously. Each set of parallel execution servers may process multiple operations. Only two sets of parallel execution servers need to be active to guarantee optimal inter-operation parallelism. A monitor routine set up is made at server admin level to manage resource utilization in conjunction with parallel execution environments.

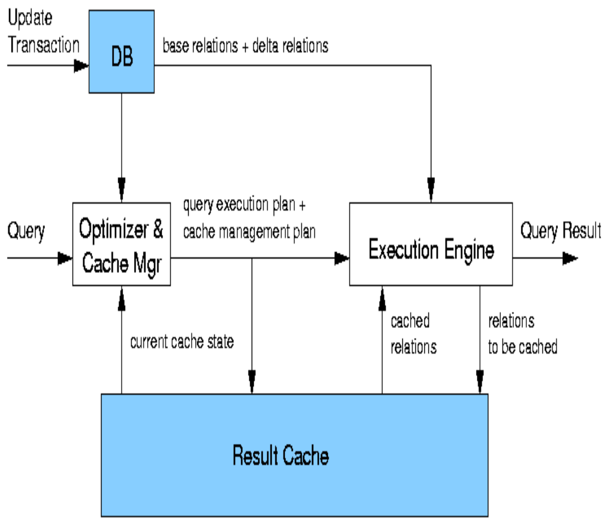

Figure:5-Query result caching representation.

A new Microsoft's language integrated query (LINQ) technology is analyzed in lab test for option instead of SQL which provides an expressive syntax and suite of APIs which facilitate classic data parallel operations: filtering, mapping, reductions, loop tiling, sorts, nested loops parallelism, prefix scans, and more. A new set of extensions to the LINQ technology, called parallel LINQ (PLINQ), which automatically optimizes and parallelizes query operations based on dynamic runtime information, is deployed. Using a familiar SQL-like syntax broadens the reach of this PLINQ, and gives programmers a more declarative and flexible way of expressing data-intensive computations**.** Thus parallelism is analyzed in this paper for query execution through multicores.

Conclusion

In this paper, we present a concept for how to cover parallelism for queries involving multiple joins and aggregate operators processing that strives for maximum resource utilization is discussed. A model is designed for query execution which supports intra and inter operator parallelism .A methodology for load sharing is discussed i.e. how to distribute data as well how to cache the intermittent results dynamically during runtime in order to reduce the delay and lastly as a frame work with examples is explained in order to understand how queries are executed optimistically and how resource can be utilized in maximum and get the output in desired time. This can be extended for xml databases with multiple aggregate operators for parallelism with fast query processing algorithms further.

Acknowledgement

We would like to thank Dr. R.Saravanan, Director, School of Information Technology and Engineering as well as M.N.Narayanamoorthi Assistant professor and N.Siva Kumar Assistant professor of the VIT University, India for motivating research and providing facilities to test our design in labs.

References:

[1.] Takesh Fukda IBM TOKYO Research Laboratory .*Parallel processing of multiple aggregate queries on nothing shared processors EDBT-1998* www.trl.ibm.com/projects

[2.] M.Al.Hajj Hassian and M.Bamba LIFO University of Orleans France. Springer Very Large Database Berlin – [2007] *Parallel processing of Group by join queries on nothing shared machines.*

[3]. David Tanair and Rebecca Boon chapter 7&8 *Performance analysis of group by after join in parallel databases.*www.Science direct.com
Jan – [2003].

[4.] Amol Dasphande Lisa Hellester *Flow algorithm for parallel query optimization* IEEE Transaction on parallel computing 2007 August 22.5.R.Acker,C.Roth R.Bayer .

[5] .J. HoKim Yun Ho Kim Soo Hook -*Parallel query processing in databases on multicore architecture* Springer-Verlag Berlin Heidelberg 20086

[6] *An efficient processing of queries with joins and aggregate functions in data ware house environment.* IEEE 13th Workshop on database and expert applications 2002.

[7]. Lei Chen #1, Christopher Olston ∗2, Raghu Ramakrishnan ∗3 *Parallel Evaluation of Composite Aggregate Queries* ICDE TALK April 2008.

[8]. Patrick valduriez, Hewlett Packard labs Waquar Hasan INRIA USA.
Open issues in parallel query optimization. Sigmoid record vol 25 September 1996.

[9]. Performance and tuning: Optimiser and Abstract plans. Sybase Manual Adaptive server @enterprise12.5.1.

[10]. D.Taniar, Royal Melbourne Institute of Technology. Australia. Y.jiang, School of IT Victoria University. *Aggregate join query processing in parallel database systems. IEEE-2000 Journal.*

[11]. Intel@ Multi-core technology, http://www.intel.com/multicore/

[12] . Blair Guy,"An Analysis of Multicore Microprocessor capabilities and their suitability for current day application Requirements" , Bowie University , Maryland in Europe, Nov 2007.

[13]. Muti-Core Processors—The Next Evolution in Computing http://multicore.amd.com/Resources/33211A_Multi-Core_WP_en.pdf

[14]. Kenneth .Joy-A ,Lawrence Berkely laboratory ,Luke J Gosinic, Institute of data analysis and visualization university of cannada- Parallel method for fast query processing.- workshop on extremely largedatabase-2007.

[15]. Wolf-Tilo Balke and Ulrich Guntenz University of California-Multi objective query processing for database. International conference on extending database technology-2004 Greece.

Authors Details

[1] Mr.P.Mohankumar is working as Assistant Professor.,(senior) in School of Information Technology and Engineering(SITE), VIT University, Vellore. His area of Research includes Advanced Database Management Systems and Neural networks. He is having more than ten years teaching and academic activities. He is currently working in optimization of parallel query processing in distributed databases as a research.

[2] Mr.P.Kumaresan is working as Assistant Professor in School of Information Technology and Engineering (SITE) at VIT University, Vellore .He has vast teaching experience. His research interests include Data Mining – Semantic Web Mining. He guided various UG and PG projects.

[3] Dr.J.Vaideeswaran. is working as Senior Professor in Architecture and Embedded systems division, in School of Computing Sciences and Engineering, VIT University, Vellore. His research includes High Performance Computing and Computer Architecture. Having more the 2 decades teaching experience.