An Adaptive Join Algorithm For Efficient Query Processing In Heterogeneous Data Sets

Mr. Pratik Patel
PG-Student, Department of Computer Science,
Parul Institute of Technology
Vadodara (Gujarat), India.

Mr. A. M. Rana
Assistant Professor
Department of Computer Science,
Parul Institute of Technology
Vadodara (Gujarat), India.

Abstract: *This work is focused on how join operator works in heterogeneous environment. Join is a means for combining fields from two tables by using values common to each. Join operation is considered as one of the fundamental operations of relational databases and it is also difficult operation to efficiently implement. The testing of the effectiveness of join algorithms is proposed.*
Adaptive join algorithms have recently attracted a lot of attention in emerging applications where data is provided by autonomous data sources through heterogeneous network environments. In traditional join techniques, they can start producing join results as soon as the first input tuples are available, thus improving pipelining by smoothing join result production and by masking source or network delays. In this work, Evaluation of the performance and comparison of Multiway join (MJoin), Double Index Nested Loop Reactive Join (DINER), and Multiple Index Nested Loop Reactive Join (MINER).My proposed system shows that improved MINER outperforms in comparison with the previous join algorithms in producing result tuples at a significantly higher rate, while making better use of the available memory

KEYWORDS: Query processing, Streams, and Joins

1. Introduction

In Real systems, it is difficult to maintain all the data is stored in one large table. To do so would require maintaining several duplicate copies of the same values and could threaten the integrity of the data .Instead, IT department everywhere almost always divide their data among several different tables. Because of this, a method is needed to simultaneously access two or more tables by using join operation. Join is a means for combining fields from two tables by using values common to each. Join operation is considered as one of the fundamental operations of relational databases and it is also difficult operation to efficiently implement. Joins are one of the basic constructions of SQL and databases such as, they combine records from two or more database tables into one row source, one set of rows with the same columns and these columns can originate from either of the joined tables as well as be formed using an expressions and built-in or user-defined functions.

Joins are used for joining records or fields from two or more tables in a database by using a value common to both the tables and the result set can be stored or saved in a table [1]. There are four types of joins and they are specified by ANSI (American National Standard Institute) and they are INNER, OUTER, LEFT, and RIGHT. Inner join are further classified into equi join, natural join and cross join. Outer join are further classified as left outer join, right outer join and full outer join. Two tables are used as an example of joins; they are Dept ID column of the Emp table and Dept table.

Emp Table

LastName	DeptId
Aa	11
Bb	13
Cc	13
Dd	14
Ee	14

Dept Table

Dept Id	DeptName
11	Sales
13	Engineering
14	Clerical
15	Marketing

Figure. 1 *Example of Join*

Inner join are considered as a common operation of join and they are also a default type of join based on the predicate. They combine the values of two tables and the results are kept in new table. Inner join has both explicit join notation and implicit join notation. Outer join does not expect any matching record and they does not require each record in two tables to be joined to have a matching record. Outer join does not have Implicit join notation.

Explicit join notation and implicit join notation are the ways of expressing join syntax and they are specified by SQL explicit join notation uses the keyword "JOIN" and "On" [1]

Select * from Emp INNER JOIN Dept On Emp.DeptID = Dept.DeptID;

Implicit join notation list the join table and they use select statement:-
Select * from Emp, Dept Where Emp.DeptID = Dept.DeptID;

Adaptive Join: Adaptation schemes for join queries are significantly more complicated to design and analyse compared to those for selection ordering for several reasons. The key performance of adaptive joins is rapid availability of first results and a continuous rate of tuple production. It overcomes the situation like initial delay, slow data delivery or bursty arrival, which can affect the efficiency of join [2] It is used for fast data delivery from one location to another location. When first input tuple is available then starts its joining process compared to traditional joining processes.

2. Motivation:

Some additional challenges in adaptive joins compared to traditional joins [3] are: The input relations are provided by autonomous network sources. The implication is that one has little or no control over the order or rate of arrival of tuples. Data is transported through unreliable network environment. It is often unsuitable or in-efficient because most traditional join algorithms cannot produce results until at least one of the relations is completely available, the complete data might be available after a long time. Sometimes these algorithms are unusable, if data is completely available but they produce partial results. The availability of partial join results is important for wide range of applications. Their main advantage over traditional join techniques is that, they can start producing join results as soon as the first input tuples are available, thus improving pipelining by smoothing join result production and by masking source or network delay.

3. Objectives:

a) Experimental study of DINER (Double Index Nested Loop Reactive Join)
b) Calculate the performance comparison of the DINER, MINER and MJoin.
c) Evaluate the performance of MINER (Multiple Index Nested Loop Reactive Join)

4. Dissertation work: The aim of implementing and optimizing of MINER algorithm and comparing it with the DINER and MJoin. In MINER, It uses homogeneous database for higher quality join results. All the data is stored into the buffer and data is fetched according to their index number and apply a join query. In DINER (Double Index Nested loop Reactive Join), it uses heterogeneous database and it is a novel adaptive join algorithm that supports both equality and range join predicates. The feature of this DINER algorithm is that

they are unblocking and they deal with adaptive. They can produce join result, if the one relation is completely arriving. [4] In MJoin, It uses single database for joining two or more tables. Firstly, consider two tables for joining and this result is stored in the third table. Then, this third table is join with the another table i.e. fourth table.

LITERATURE REVIEW Existing Join Techniques: The main three categories of join algorithms are
a) Nested-loop join algorithm
b) Sort-merge join algorithm
c) Hash-based join algorithm

Nested-Loop Join Algorithm*:* - Nested-loop join is considered as a one of the simplest algorithm of join where, for each record of the first table the entire records of the second table has to be scanned. This process is repeated for each and every record of the first table that is for all the first table records. The loop is of two levels and they are outer loop and the inner loop. First table loop is called as outer loop and the second table loop are called as inner loop. As this, Nested loop join algorithm has a repeated input/output scans of one of the table. They are considered as inefficient.
Let the two tables be A and B, then the algorithm of Nested-loop algorithm are as
For each record of table A Read record from table A
For each record of table B Read record from table B
Compare the join attributes
If matched Then Store the records

Sort-Merge Join Algorithm: - Sort merge algorithm are considered as efficient join algorithm when compared to Nested loop join algorithm. sort merge join algorithm have two operations and they are sorting and merging. In sorting operation the two tables to be joined are sorted in ascending order. In merging operation the two sorted tables are merged.

Sort records of table A based on the join attribute
Sort records of table B based on the join attribute
Let i = 1 and j =1
Repeat Read record A(i)
Read record B(j)
If join attribute A(i) < join attribute B(j)

Then i++
Else If join attribute A(i) > join attribute B(j)
Then j++
Else Put records A(i) and B(j) into the Qr

Hash Based join algorithm: - In hash based join algorithm hashing and probing are the two processes. A hash table is created by hashing all records of the first table using a particular hash function. Records from the second table are also hashed with the same hash function and probed. If any match is found, the two records are concatenated and placed in the query result. A decision must be made about which table is to be hashed and which table is to be probed. Since a hash table has to be created, it would be better to choose the smaller table for hashing and the larger table for probing. The hash join algorithm is given as

Let H be a hash function
For each record in table B
Read a record from table B
Hash the record based on join attribute value using hash function H into hash table
For each record in table A
Read a record from table A
Hash the record based on join attribute value using H Probe into the hash table
If an index entry is found
Then
Compare each record on this index entry with the record of table S
If matched
Then Put the pair into Qr

5. PROPOSED SYSTEM MODULE DESCRIPTION

1. DINER (Double Index Nested-loops Reactive) Module:

MODERN information processing is moving into a realm where often need to process data that are pushed or pulled from autonomous data sources through heterogeneous networks. The key differences between DINER and existing algorithms are 1) an intuitive flushing policy for the Arriving phase that aims at maximizing the amount of overlap of the join attribute values between memory resident tuples of the two relations and 2) a lightweight Reactive phase that allows the algorithm to quickly move into processing tuples that were flushed to disk when both data sources block. The key idea of our flushing policy is to create and adaptively maintain three nonoverlapping value regions that partition the join attribute domain, measure the "join benefit" of each region at every flushing decision point, and flush tuples from the region that doesn't produce many join results in a way that permits easy maintenance of the three-way partition of the values. When tuples are flushed to disk they are organized into sorted blocks using an efficient index structure, maintained separately for each relation (thus, the part "Double Index" in DINER). This optimization results in faster processing of these tuples during the Reactive and Clean-up phases. The Reactive phase of DINER employs a symmetric nested loop join process, combined with novel bookkeeping that allows the algorithm to react to the unpredictability of the data sources. The fusion of the two techniques allows DINER to make much more efficient use of available main memory. To demonstrate in the experiments that DINER has a higher rate of join result production and is much more adaptive to changes in the environment, including changes in the value distributions of the streamed tuples and in their arrival rates.

2. MINER Module:

MINER extends DINER to multiday joins and it maintains all the distinctive and efficiency generating properties of DINER. MINER maximizes the output rate by: 1) adopting an efficient probing sequence for new incoming tuples which aims to reduce the processing overhead by interrupting index lookups early for those tuples that do not participate in the overall result; 2) applying an effective flushing policy that keeps in memory the tuples that produce results, in a manner similar to DINER; and 3) activating a Reactive phase when all inputs are blocked, which joins on-disk tuples while keeping the result correct and being able to promptly hand over in the presence of new input. Compared to DINER, MINER faces additional challenges namely: 1) updating and synchronizing the statistics for each join attribute during the online phase, and 2) more complicated bookkeeping in order to be able to guarantee correctness and prompt handover during reactive phase.

3. Memory Allocated DINER & MINER Module

To investigate the impact that several parameters may have on the performance of the DINER algorithm, through a detailed sensitivity analysis. Moreover To evaluate the performance of MINER when vary the amount of memory allocated to the algorithm and the number of inputs. The main findings of this study include:

- A Faster Algorithm. DINER provides result tuples at a significantly higher rate, up to three times

in some cases, than existing adaptive join algorithms during the online phase. This also leads to a faster computation of the overall join result when there are bursty tuple arrivals.

- A Leaner Algorithm. The DINER algorithm further improves its relative performance to the compared algorithms in terms of produced tuples during the online phase in more constrained memory environments. This is mainly attributed to our novel flushing policy.
- A More Adaptive Algorithm. The DINER algorithm has an even larger performance advantage over existing algorithms, when the values of the join attribute are streamed according to a no stationary process. Moreover,
- It better adjusts its execution when there are unpredictable delays in tuple arrivals, to produce more result tuples during such delays.
- Suitable for Range Queries. The DINER algorithm can also be applied to joins involving range conditions for the join attribute. PMJ also supports range queries but, it is a generally poor choice since its performance is limited by its blocking behavior.

An Efficient Multiway Join Operator. MINER retains the advantages of DINER when multiple inputs are considered. MINER provides tuples at a significantly higher rate compared to MJoin during the online phase. In the presence of four relations, which represents a challenging setup, the percentage of results obtained by MINER during the arriving phase varies from 55 percent (when the allocated memory is 5 percent of the total input size) to more than 80 percent (when the allocated memory size is equal to 20 percent of the total input size).

6. The contributions of this project:-

- To introduce DINER a novel adaptive join algorithm that supports both equality and range join predicates. DINER builds on an intuitive flushing policy that aims at maximizing the productivity of tuples that are kept in memory.
- DINER is the first algorithm to address the need to quickly respond to bursts of arriving data during the Reactive phase. To propose a novel extension to nested loops join for processing disk-resident tuples when both sources block, while being able to swiftly respond to new data arrivals.
- To introduce MINER, a novel adaptive multiway join algorithm that maximizes the output rate, designed for dealing with cases where data are held by multiple remote sources. To provide a thorough discussion of existing algorithms, including identifying some important limitations, such as increased memory consumption because of their inability to quickly switch to the Arriving phase and not being responsive enough when value distributions
- To provide an extensive experimental study of DINER including performance comparisons to existing adaptive join algorithms and a sensitivity analysis. The results demonstrate the superiority of DINER in a variety of realistic scenarios. During the online phase of the algorithm, DINER manages to produce up to three times more results compared to previous techniques. The performance gains of DINER are realized when using both real and synthetic data and are increased when fewer resources (memory) are given to the algorithm. To also evaluate the performance of MINER, and to show that it is still possible to obtain early a large percentage of results even in more elaborated setups where data are provided through multiple inputs. The experimental study shows that the performance of MINER is 60 times higher compared to the existing MJoin algorithm when a four-way star join is executed in a constrained memory environment.

7. PERFORMANCE ANALYSIS

To demonstrate DINER's superior performance over a variety of real and synthetic data sets in an environment without network congestion or unexpected source delays. To plot the cumulative number of tuples produced by the join algorithms over time, during the online phase for the CSCO stock and the Weather data sets.To observe that DINER has a much higher rate of tuples produced that all other competitors. For the stock data,while RPJ is not able to produce a lot of tuples initially, it manages to catch up with XJoin at the end. To compare DINER to RPJ and HMJ on the real data sets when to vary the amount of available memory as a percentage of the total input size. The y axis represents the tuples produced by RPJ and HMJ at the end of their online phase (i.e., until the two relations have arrived in full) as a percentage of the number of tuples produced by DINER over the same time. The DINER algorithm significantly outperforms RPJ and HMJ, producing up to 2.5 times more results than the competitive techniques. The benefits of DINER are more significant when the size of the available memory given to the join algorithms is reduced. In the next set of experiments, to evaluate the performance of the

algorithms when synthetic data are used. In all runs, each relation contains 100,000 tuples.

8. CONCLUSIONS

In this work, to introduce DINER, a new adaptive join algorithm for maximizing the output rate of tuples, when two relations are being streamed to and joined at a local site. The advantages of DINER stem from 1) its intuitive flushing policy that maximizes the overlap among the join attribute values between the two relations, while flushing to disk tuples that do not contribute to the result and 2) a novel re-entrant algorithm for joining disk resident tuples that were previously flushed to disk. Moreover, DINER can efficiently handle join predicates with range conditions, a feature unique to this technique. To also present a significant extension to this framework in order to handle multiple inputs. The resulting algorithm, MINER addresses additional challenges, such as determining the proper order in which to probe the in-memory tuples of the relations, and a more complicated bookkeeping process during the Reactive phase of the join. Through this experimental evaluation, we have demonstrated the advantages of both algorithms on a variety of real and synthetic data sets, their resilience in the presence of varied data and network characteristics and their robustness to parameter changes.

9. REFERENCES

[1]. J.Jayashree and C.Ranichandra "Join Algorithm for Efficient Query Processing For Large Datasets" Asian Journal of Computer Science and Information Technology 2: 3 (2012) 31 –35

[2]. Mihaela A.Bornea, Vasilis Vassalos, Yannis Kotidis, Antonios Deligiannakis: Adaptive Join Operators for Result Rate Optimization on Streaming Inputs. IEEE Trans. Knowl. Data Eng. 22(8): 1110-1125 (2010)

[3]. J. D. Ullman, H. Garcia-Molina, and J. Widom. Database Systems: The Complete Book. Prentice Hall, 2001

[4]. M. A. Bornea, V. Vassalos, Y. Kotidis, and A. Deligiannakis. DoubleIndex Nested-loop Reactive Join for Result Rate Optimization. In ICDEConf., 2009

[5]. David Taniar, Clement H.C. Leung, Wenny Rahayu, Sushant Goel. (2008) "High-Performance Parallel Database Processing and Grid Databases" A John Wiley

[6]. Z. G. Ives, D. Florescu, and et al. An Adaptive Query Execution System for Data Integration.In SIGMOD, 1999.

[7]. W. Hong and M. Stonebraker. Optimization of Parallel Query Execution Plans in XPRS. In PDIS, 1991

[8]. T.Urhan and M.J.Franklin.Xjoin: A Relatively scheduled pipilined join operator.IEEE Data Eng.Bull, 23920,2000

[9]. S.D Viglas,J.F.Naughton and J.Burger.Maximizing the output rate of multiway join queries over streaming information sources.In VLDB 2003: proceeding of the 29th international

[10]. J. Dittrich, B. Seeger, and D. Taylor. Progressive merge join: A generic and non-blocking sort-based join algorithm. In Proceedings of VLDB, 2002.

[11]. M. F. Mokbel, M. Lu, and W. G. Aref. Hash-Merge Join: A Non-blocking Join Algorithm for Producing Fast and Early Join Results. In ICDE Conf., 2004.nal conference on very large databases 2003.

[12]. Y. Tao, M. L. Yiu, D. Papadias, M. Hadjieleftheriou, and N. Mamoulis. RPJ: Producing Fast Join Results on Streams Through Rate-based Optimization. In Proceedings of ACM SIGMOD Conference, 2005