Adaptive Join Operators for Result Rate Optimization on Streaming Inputs

1.Mrs. M. Mary Rexcy Asha,

2.Miss R.Ishwariya,

3. Ms. M. Geetha

1 Associate Professor, Department of M.C.A, Panimalar Engineering College, Chennai
2 PG Scholar, Department of M.C.A, Panimalar Engineering College, Chennai.
3 Associate Professor, Department of M.C.A, Panimalar Engineering College, Chennai

ABSTRACT Adaptive join algorithms have recently attracted a lot of attention in emerging applications where data are provided by autonomous data sources through heterogeneous network environments. 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 delays. In this paper, The first propose Double Index NEsted-loops Reactive join (DINER), a new adaptive two-way join algorithm for result rate maximization. DINER combines two key elements: an intuitive flushing policy that aims to increase the productivity of in-memory tuples in producing results during the online phase of the join, and a novel reentrant join technique that allows the algorithm to rapidly switch between processing in-memory and disk-resident tuples, thus, better exploiting temporary delays when new data are not available. Then extend the applicability of the proposed technique for a more challenging setup: handling more than two inputs. Multiple Index NEsted-loop Reactive join (MINER) is a multiway join operator that inherits its principles from DINER.

I. INTRODUCTION

Adaptive join algorithms were created in order to lift the limitations of traditional join algorithms in such environments. By being able to produce results whenever input tuples become available, adaptive join algorithms overcome situations like initial delay, slow data delivery, or bursty arrival, which can affect the efficiency of the join. All existing algorithms work in three stages. During the Arriving phase, a newly arrived tuple is stored in memory and it is matched against memory-resident tuples belonging to the other relations participating in the join. Since the allocated memory for the join operation is limited and often much smaller than the volume of the incoming data, this results in tuple migration to disk. The decision on what to flush to disk influences significantly the number of results produced during the Arriving phase. The Arriving phase is suspended when all data sources are temporarily blocked and a Reactive phase kicks in and starts joining part of the tuples that have been flushed to disk. An important desideratum of this phase is the prompt handover to the Arriving phase as soon as any of the data sources restarts sending tuples. Each algorithm has a handover delay which depends on the minimum unit of work that needs to be completed before switching phases. This delay has not received attention in the past, but show that it can easily lead to input buffer overflow, lost tuples, and henc incorrect results. When all sources complete the data transmission, a Cleanup phase is activated and the tuple that were not joined in the previous phases (due to flushin of tuples to disk) are brought from disk and joined. Even if the overall strategy has been proposed for a multiway join, most existing algorithms are limited to a two-way join. Devising an effective multiway adaptive join operator is a challenge in which little progress has been made.In this paper, To propose two new adaptive join algorithms for output rate maximization in data processing over autonomous distributed sources. The first algorithm, Double Index NEsted-loop Reactive join (DINER) is applicable for two inputs, while Multiple Index NEsted-loop Reactive join (MINER) can be used for joining an arbitrary number of input sources. DINER follows the same overall pattern of execution discussed earlier but incorporates a series of novel techniques and ideas that make it faster,

leaner (in memory use), and more adaptive than its predecessors.

More specifically, 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. As will be explained, proper management of the three partitions allows us to increase the number of tuples with matching values on their join attribute from the two relations, thus, maximizing the output rate. 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 Cleanup 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.MINER extends DINER to multiway 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.To generalize the reactive phase of DINER for multiple inputs using a novel scheme that dynamically changes the roles between relations.

results could be obtained. Although there were some improvements that attenuate the blocking effect, the first efficient nonblocking sort-based algorithm was PMJ .Hash Merge Join (HMJ) , based on XJoin and PMJ, is a nonblocking algorithm which tries to combine the best parts of its predecessors while avoiding their shortcomings.Finally, Rate-based Progressive Join (RPJ) is an improved version of HMJ that is the first algorithm to make decisions, e.g., about flushing to disk, based on the characteristics of the data.

II RELATED WORK

Existing work on adaptive join techniques can be classified in two groups: hash based and sort based Examples of hash-based algorithms include DPHJ and XJoin , the first of a new generation of adaptive nonblocking join algorithms to be proposed. XJoin was inspired by Symmetric Hash Join (SHJ) which represented the first step toward avoiding the blocking behavior of the traditional hash-based algorithms. SHJ required both relations to fit in memory; however, XJoin removes this restriction. MJoin [algorithm appeared as an enhancement of XJoin in which its applicability is extended to scenarios where more than two inputs are present. The above-mentioned algorithms were proposed for data integration and online aggregation. Pipelined hash join , developed concurrently with SHJ, is also an extension of hash join and was proposed for pipelined query plans in parallel main memory environment.Algorithms based on sorting were generally blocking,since the original sort merge join algorithm required an initial sorting on both relations before the

III. 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 Cleanup 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 multiway 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 nonstationary 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).

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 change.
- 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.

IV. 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.

V.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 attributevalues between the two relations, while flushing to disk tuples that do not contribute to the result and 2) a novel reentrant 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.

VI.REFERENCES

[1] John Sharp R., (2008) ”Microsoft Visual C# Step by Step”, TataMcGraw Hill Publications, Seventh Edition. pp. 144-176

[2] W. Hong and M. Stonebraker, “Optimization of Parallel Query Execution Plans in XPRS,” Proc. Int’l Conf. Parallel and Distributed Information Systems (PDIS), 1991. pp. 219–311

[3] Z.G. Ives et al., “An Adaptive Query Execution System for Data Integration,” Proc. ACM SIGMOD, 1999. pp. 176-287

[4] B. Pang and L. Lee, “Opinion Mining and Sentiment Analysis,” Foundations and Trends in Information Retrieval, vol. 2, nos. 1–2, 2008, pp. 1–135.

[5] M. Hu and B. Liu, “Mining and Summarizing Customer Reviews,” Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD), ACM Press, 2004, pp. 168–177.

[6] N. Jindal and B. Liu, “Opinion Spam and Analysis,” Proc. Conf. Web Search and Web Data Mining (WSDM), ACM Press, 2008, pp. 219–230.

SITES REFERRED

[7] http:// www.microsoft.com

[8] http://www.sourcefordgde.com

[9] http://www.ieee.org

[10] http://www.adobe.com/support/techdocs/321644.html