Researchers

C. Naga Pradeep Kumar

Asst.Prof,IT Dept., SRIT, Anantapur, Andhra Pradesh

Prof. A. Ananda Rao

Principal, JNTUACEA, Anantapur, Andhra Pradesh

Email: nagapradeep.srit@gmail.com

MINER: An Improved Adaptive Join Algorithm

Abstract:

Adaptive join algorithms were created to overcome the drawbacks of traditional join algorithms in emerging data integration or online aggregation environments. The input relations to adaptive joins are continuously retrieved from remote sources. The main objective for designing these algorithms is to i) start producing the first output tuples as soon as possible ii) produce the remaining results at a fast rate. One of the early adaptive join algorithm Multiple Index Nested-loop Reactive join (MINER) is a multi-way join operator used for joining an arbitrary number of input sources. Here MINER was limited to chain joins. In this paper, MINER is extended to support snowflake joins, where each relation may participate in joins with more than two join attributes. It will improve producing result tuples at a significantly higher rate, while making better use of the available memory.

Keywords: Query processing, Snowflake joins, Streams.

Introduction:

In a distributed environment, statistical information for the available data sources may be minimal, where the availability or load of physical resources is prone to be changed. Consequently, traditional query optimization can lead to poor performance, especially in long running query evaluations, as the query optimizer may not, at compile time, have the necessary statistics, good selectivity estimates, knowledge of the runtime bindings in the query, or knowledge of the available system resources required to produce an optimal query plan (QP). In addition, traditional optimizers cannot predict the future availability of resources. Adaptive query processing (AQP) addresses this, by adapting the query processing to changing environmental conditions at runtime.

Traditional join algorithms [9, 10] assume that all input data is available beforehand. This assumption is not suitable in emerging data integration or online aggregation environments; a key performance metric is rapid availability of first results & higher join result rate. With the goals of avoiding the blocking behavior of remote data sources & producing join results as quickly as possible a family of adaptive join algorithms are developed [1,3,4,5,6,7]. Adaptive join algorithms have the ability of non-blocking behavior and producing join results even if one or both sources are blocked.

Adaptive joins are designed to deal with some additional challenges over traditional join algorithms: input relations to them are provided by autonomous data sources through heterogeneous networks. Data is transported through

VOLUME 1 NUMBER 4 (July/Aug 2012)
ISSN: 2277–1891

unreliable network environments. The problem is that data arrival rate can't be controlled. Since the data access over wide-area networks involves a large number of remote data sources, intermediate sites & communication links, all of which are vulnerable to congestion and failures. Adaptive join algorithms overcome situation like initial delay, slow data delivery, or bursty arrival [2,11] which can affect the efficiency of 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 [1,3]. In this paper, a similar approach of MINER[1] that supports snowflake joins is proposed, where data are held by multiple remote sources.

The rest of the paper is organized as follows: Section 2 illustrates Snowflake MINER architecture and its operations in detail. In Section 3, MINER for Snowflake joins algorithm is analyzed. Section 4 is carried out through experiments and results their conclusions and future work are proposed in Section 5.

Snowflake MINER Architecture

A snowflake join is a join where a large table is joined with three or more smaller base tables or joins relations, where these smaller base tables may themselves are joined to three or more smaller base tables or joins relations. The following architecture is proposed to extend the MINER to support snowflake joins.

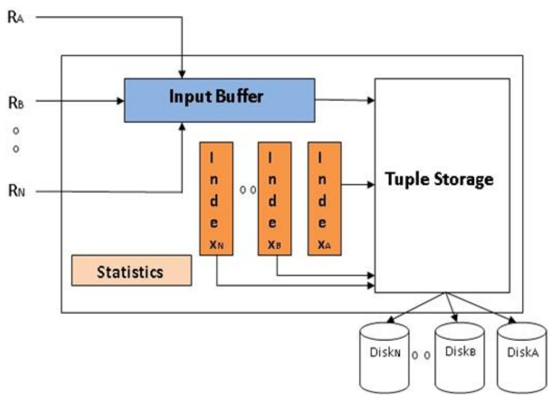

Fig.1 MINER for snowflake joins

Fig. 1 shows MINER for snowflake joins architecture. MINER for snowflake joins proceeds in three stages, each of which is performed by a separate thread. The First Stage joins memory resident tuples. Tuples arrive in the input buffer. Tuples are processed in memory using an index. Some statistics might be kept about relations (e.g., cardinalities). When memory is exhausted, some tuples are flushed on disk by using flushing policy [5]. The Second Stage joins tuples that have been flushed to disk due to memory constraints. It is activated when all streams experience delay. The Third Stage is a clean-up stage, which performs any necessary matching to produce results missed by the first two stages. The first and second stages run in an interleaved fashion, the second stage takes over when the first becomes blocked due to a lack of input. These stages are terminated after all input has been received, at which point the third stage is initiated.

Proposed Algorithm

In this section algorithm for MINER to support snowflake joins of the following form is proposed

$$\begin{array}{c} R_A \bowtie R_B \bowtie R_C. \\ \bowtie \\ R_D \end{array}$$

Fig.2 Snowflake Joins

VOLUME 1 NUMBER 4 (July/Aug 2012)
ISSN: 2277–1891

Fig. 2 illustrates each relation may participate in joins with more than two join attributes.

Step1: While relations R_A, R_B, R_C and R_D still have tuples do following steps.

Step2: If tuple ti € Ri arrived (i € {R_A, R_B, R_C ,R_D }) in input buffer, then move tuples from input buffer to MINER process space.

Step3: Apply join operation for set of matching tuples found using Indexes.

Step4: When Used Memory exceeds Threshold limit then flush those unprocessed tuples that reside in MINER process space for maximum time to disk such that Used Memory must come into below Threshold limit.

Step5: If transmission of R_A, RB, RC and R_D is blocked more than wait threshold then

5a. Bring tuples that have been flushed to disk due to memory constraints to MINER process space.

5b. Apply join operation for set of matching tuples found using Indexes.

5c. If maximum numbers of input tuples are collected in input buffer size is satisfied then repeat steps 2 to 4.

Once all relations have been received in their entirely repeat steps 2 to 4.

There exist multiple join attributes per relation, hence algorithm maintains for each input relation a separate index on each join attribute. In this algorithm hash-table data structure is utilized as indexes. In Step 2 Algorithm is said to be in First Stage, in which tuples will arrive to input buffer. When a new tuple comes, it is stored and indexed in the Miner process space of its relation, based on the relation's join attributes. Then this new tuple is checked for matching with all the in-memory tuples belonging to all other relations participating in the join. This process will continue until memory is exhausted, or else algorithm flushes tuples from the relation with the largest number of in-memory tuples to respective disks, which is mentioned in step 4.

When all data sources experience delays then Second Stage will be activated, in which algorithm performs joins between previously flushed data. Any matches found are output as result tuples. So that algorithm will progress while no input is being delivered. If any of the join inputs have resumed producing tuples, then the algorithm switches back to the First Stage. After all tuples have been received from all relations then Third Stage executes, that makes sure that all the tuples should be in the result set are ultimately produced.

Experimental Results

Table 1 illustrates the percentage of the total result obtained by algorithm during the first stage, in the presence of multiple inputs while we vary the available memory. Result of Table 1 is summarized in Fig. 3 which presents a comparison between MINER for SSTAR joins and MINER for Snowflake joins. There is a modest increase in the number of results obtained as we increase the available memory.

Table.1 Ratio between memory size and results during first stage

Memory size in percentage	% of total results after first stage in MINER for SSTAR join	% of total results after first stage in MINER for Snowflake join
5%	25	30
10%	45	50
15%	63	70
20%	75	80

VOLUME 1 NUMBER 4 (July/Aug 2012)
ISSN: 2277–1891

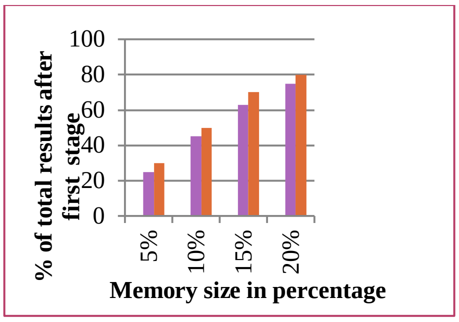

Fig.3 Comparison

Conclusions and Future Work

A novel adaptive join algorithm MINER for Snowflake Joins has been proposed, that produces result tuples at a significantly higher rate, while making better use of the available memory. It is also designed for dealing with cases where data are held by multiple remote sources and relations that may participate in joins with more than two join attributes.
In future work it can explore to the optimization problem of how to best to split a very large multi-way join into a set of smaller multi-way joins.

References

[1] Mihaela A. Bornea, Vasillis Vassalos, Yannis Kotidis, and Antonios Deligiannakis, " Adaptive Join Operators for Result Rate Optimization on Streaming Inputs", IEEE Knowledge & Data Engg., August 2010, Vol. 22, No. 8
[2] S. Babu, and P. Bizarro, "Adaptive Query Processing in the Looking Glass," Proc. Conf. Innovative Data Systems Research (CIDR), 2005.
[3] S.D. Viglas, J.F. Naughton, and J. Burger, " Maximizing the Output Rate of Multi-Way Join Queries over Streaming Information Sources," Proc. 29th Int'l conf. Very Large Data Bases (VLDB'03),pp.285-296,2003
[4] J. Dittrich, B. Seeger, and D. Taylor, "Progressive Merge Join: A Generic and Non-Blocking Sort-Based Join Algorithm," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2002.
[5] M.F. Mokbel, M. Lu, and W.G. Aref, "Hash-Merge Join: A Non-Blocking Join Algorithm for Producing Fast and Early Join Results" Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2004.
[6] Y. Tao, M.L. You, D. Papadias, M. Hadjieleftheriou, and N. Mamoulis, "RPJ: Producing Fast Join Results on Streams through Rate-Based Optimization," Proc. ACM SIGMOD, 2005.
[7] T. Urhan, and M.J. Franklin, "XJoin: A Reactively-Scheduled Pipelined Join Operator," IEEE Data Eng. Bull., Vol. 23, no. 2, pp. 27-33. 2000.
[8] Z.G. Ives et al., "An Adaptive Query Execution System for Data Integration," Proc. ACM SIGMOD, 1999.
[9] J.D. Ullman, H. Garcia-Molina, and J. Widom, Database Systems: The Complete Book. Prentice Hall, 2001.
[10] Mishra and M. H. Eich. Join Processing in Relational Databases. *ACM Computing Surveys*, 24(1): 63–113, 1992.
[11] L. Amsaleg, M. J. Franklin, A. Tomasic, and T. Urhan. Scrambling Query Plans to Cope with Unexpected Delays. *PDIS Conf.*, Miami, USA, 1996

Author Details

C. Naga Pradeep Kumar
Assistant Professor, IT Department, SRIT, Anantapur(D), A.P., India
Prof. A. Ananda Rao
Principal, JNTUACEA, Anantapur(D), A.P., India.

VOLUME 1 NUMBER 4 (July/Aug 2012)
ISSN: 2277–1891