2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.83

Research on Outer Join Optimization in Parallel DBMS

Juntao Li [+] and Xiaoling Xia

School of Computer Science and Technology Donghua University,Shanghai,China

Abstract. parallel database management systems (PDBMS) are widely used to process the ever-increasing data volume and complex queries in large-scale enterprise. Business Intelligence (BI) generated a large amount of outer joins to query data warehouses powered by parallel DBMS. With a brief introduction to the development of outer join optimization in parallel DBMS, we firstly describe ROJA, DOJA, and DERA three kinds of outer join algorithms, and then compare them with each other which is the main purpose of this paper. At last, we discuss the challenges of outer join optimization that we are currently facing with.

Keywords: parallel DBMS, outer join, business intelligentce

1. Introduction

Business and governments rely heavily on data warehouses to store and analyze large amount of data. These data helps them to make sound strategic decisions. Data warehousing has recently penetrated the domains of Internet Service, social networks, advertising, and product recommendation. With the development of computer technology and database, parallel processing continues play an important role in modern data base management system.

Business Intelligence (BI) tools based on large data warehouse generate numerous outer joins to find out the useful information. Research has been done on some aspects of optimizing outer joins including outer join elimination [1], [2], [3], [4], [5], outer join reordering [6], and view matching for outer join views [7], [8], [9], [10], [11], [12]. But little research has been done on outer join optimization in parallel DBMS, probably the reason is the assumption that inner join optimization can be largely applied to outer joins as well [13]. However, it turns out that outer joins present some unique optimization opportunities in parallel DBMS [14].

On condition of this, we give the details of three outer join algorithms and compare them with each other, then analyze the advantages and disadvantages.

2. Definitions

2.1 Parallel DBMS

Traditional DBMS runs on symmetric multi processing (SMP) machines. All the data are stored on the disk in traditional DBMS. When a database operation applied, data are loaded from disk to CPU, and be processed by computer, finally, return the operation results.

Differently, parallel DBMS runs on massively parallel processing (MPP) machines. In a shared nothing parallel architecture [14], multiple nodes communicate via high-speed interconnect network and each node has exclusive access to its main memory and disks. In modern systems, to take advantage of the multiple CPUs and disks, there are usually multiple virtual processors which are implements by software running on each node for further parallelism. These virtual processors, responsible for doing the scans, joins, locking,

[+] Tel.: +86 135-2416-9059.

E-mail address: jtli_dhu@163.com

transaction management, and other data management work. In this paper, they are named by Parallel Units (PUs).

2.2 An Outer Join Select Statement

```
Select L.x, L.a, R.y
From L left outer join R
On L.x=R.y
```

This select statement is widely used as an outer join example in this paper. We will describe the three different kinds of algorithms in detail based on the statement.

2.3 Hash function used in parallel DBMS:

In parallel DBMS, data are hashed into different PUs by the hash function as follow:

$$H(I) = I \bmod N + 1$$

H (I) means the *h (i)-th* PU. *I* is the partitioning column value. *N* represents the number of the PUs in total. Fig 1 shows the data of table L and R in database.

In this paper, we assume that the partitioning columns are L.a and R.b and three PUs in total. So the data in table L and R are hashed into three PUs. Fig 2 shows the hash result.

In fact, the partitioning column in parallel DBMS is chosen by the user or assigned by the system automatically. The importance of hash is that the data which perhaps have relation with each other are divided into the same PUs.

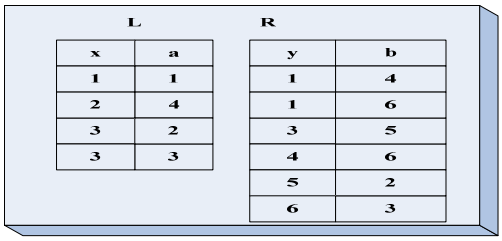

Fig.1: Data of table L and R.

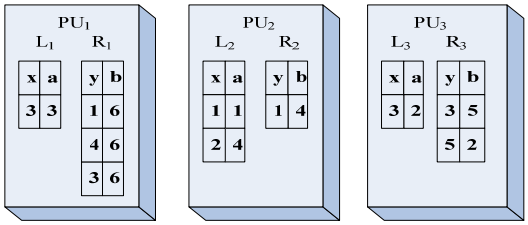

Fig.2: Hash results in parallel DBMS.The partitioning columns are L.a and R.b.

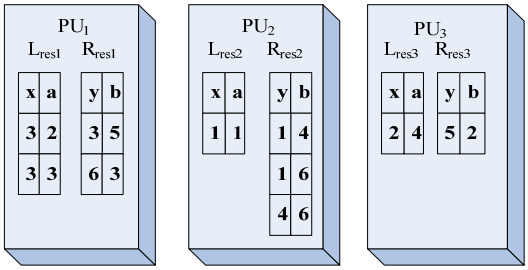

Fig.3: Redistribution results of ROJA. Data are redistributed on the partition columns L.x and R.y.

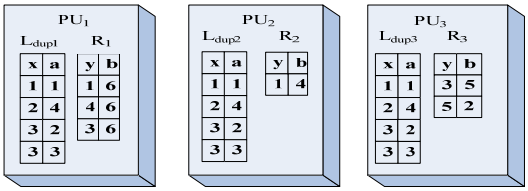

Fig.4:Duplication results of DOJA. Data of table L are duplicated on each PUs.

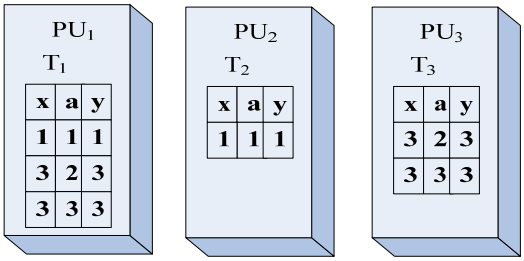

Fig.5: Inner join results of DOJA.

3. Three outer join algorithms

Since the data are already prepared in part 2, now it comes to describe the algorithms.

3.1 ROJA

The first conventional outer join algorithm is called the redistribution outer join algorithm (ROJA).The ROJA algorithm has two steps.

The first step of ROJA is called hash redistribution. In this step, DBMS will check the relational columns, which is column x of table L and column y of table R in our example. Compare the relational columns with the partitioning columns in the hash step, if they are not the same, and then redistribute the data to the PUs use the same hash function as before. Fig 3 shows the result of the first step after hash redistributing both L and R on L.x and R.y.

In the second step, the outer join operation is performed on each PU in parallel. This can be done since the first step has put all matching rows from the join relations on the same PUs [14].

3.2 DOJA

The second outer join algorithm is called the duplication outer join algorithm (DOJA) which is composed of four steps.

In the first step, DBMS choose the small one between the two tables, small here means the size of the data in a table. In our example, table L is the smaller one. Tuples of table L is duplicated on each PU so that each one of them has a complete copy of the table L. Fig 4 shows the result of the duplication of table L in fig 1.

In the second step, table L is inner joined with table R on each PU in parallel. And the results are stored in a temporary table T, as shown in fig 5.

In the third step, use the hash function of H(I), the inner join results of step two are redistribute on column T.a, as shown in fig 6.

In the fourth step, table L in each PU is left outer joined with the results from the third step, After this step, the final results of the select statement are come out, which is shown in fig 7 [2].

3.3 DERA

The third outer join algorithm is called the duplication and efficient redistribution algorithm (DERA). The following steps are executed in the DERA to evaluate the select statement [11].

In the first step, rows of the smaller table L on each PU are duplicated to all PUs and stored in a temporary table T. this step is the same as the first step in the DOJA.

In the second step, tuples of table T are left outer joined and the results are split into two temporary tables M and U. Matching tuples of T and R are stored in table M, and U contains only unique value of the tuples in table L which are not matched. Since the value of column a in table L are all unique. Table U store the value of L.a in our example. The results of table M and U are shown in fig 8.

In the third step, tuples of table U are redistributed based on column L.a, and the results are stored in another temporary table Ures, as shown in fig 9.

In the forth step, count the number of L.a of temporary table Ures. This operation is executed on each PU in parallel. We keep the value of L.a that appears as many times as the number of PUs which is three in our example. Fig 10 shows the results after the forth step.

In the fifth step, table S and Ures are inner joined on L.a with results stored in table U3.

The final results of the select statement are the union of M and U3. The union is only logical and requires no expensive duplicate removal because the two relations have no rows in common [15].

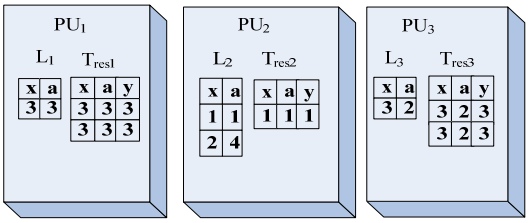

Fig.6: Redistribution inner join results of DOJA. Inner join results (Fig 5) are redistributing to T_{res}.

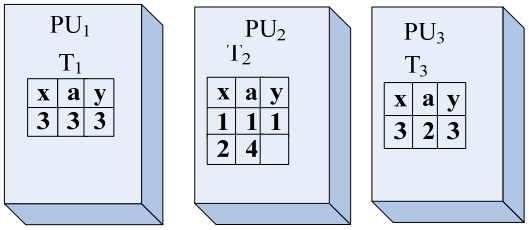

Fig.7: The forth step of DOJA: final results.

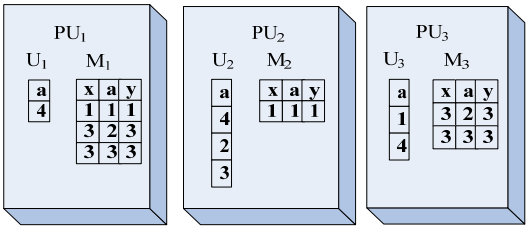

Fig.8: The second step of DERA. Tuples of table T are left outer joined and the results are split into two temporary tables M and U.

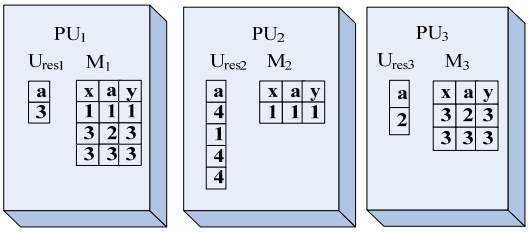

Fig.9: The third step of DERA. Tuples of table U are redistributed based on column L.a.

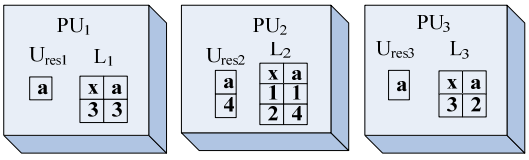

Fig.10: The forth step of DERA. Counting the number of L.a of temporary table Ures in each PUs.

4. Comparison

We have described the three kinds of outer join algorithms with an example in part 3, and in this part we will compare them with each other which is the main purpose of this paper.

ROJA is a conventional outer join algorithm in parallel DBMS. From the first step of ROJA, we can find that it would be a huge time cost if the relational columns are not the partitioning columns in parallel DBMS, for the reason that data in tables should be redistributed to different PUs, which cost a large amount of I/O and other recourses.

The advantage of ROJA is that it can be widely used in any condition without considerate the data size of two table outer join with each other.

In fact, the DOJA is specially designed for the small and large table outer joins, from the second step of DOJA; we noticed that the performance of DOJA can be worse than that of ROJA when the inner join cardinality is high. Because the inner join results also should be redistributed.

When a cost-based optimizer chooses which algorithm from ROJA and DOJA to evaluate an outer join, the optimizer has to consider the inner join cardinality in advance which is usually difficult to be accurately estimated. If the optimizer chooses the DOJA algorithm based on its inner join cardinality estimation, the performance of the DOJA algorithm can be worse than that of the ROJA algorithm.

In this case, The DERA improves the outer join algorithm based on DOJA. Though DERA is more complicated than DOJA, and much more temporary tables are needed, it totally eliminates the redistribution of the inner join results in DOJA. The improvement is remarkable, for the redistribution of data is the highest cost in any algorithms.

So, when a cost-based optimizer chooses which algorithm from DERA and DOJA to evaluate an outer join, the inner join cardinality does not affect the optimizer's decision. This is because the inner join cardinality affects the computing costs of both ROJA and DERA. On the other hand, whether DERA outperforms ROJA mainly depends on the sizes of the small and large tables which usually can be accurately estimated. One important issue is when the optimizer chooses DERA over ROJA. As usual, a cost based optimizer will choose the DERA algorithm over the ROJA algorithm only when it determines the cost of applying DERA is smaller, considering factors such as the number of PUs, networking bandwidth, sizes of the joined tables, characteristics of I/O configurations in the system [14].

5. Conclusions

Business Intelligence tools based on large data warehouse play more and more important roles in large-scale enterprise to answer important business questions and demand high performance. Outer joins in these tools are frequently generated, a great many research has been done on inner join optimization while little has been done on outer join optimization. In this paper, we describe three different kinds of outer join algorithms which are presented by other scientists and engineer and compare them with each other, analyze the advantages and disadvantages for each algorithm, and describe the appropriate situation they are fit for.

Although the ROJA, DOJA, DERA are already widely used in many products, they are faced with challenges:

- ROJA is suitable for the outer join with any table data size, but the redistribution of the original data takes a huge time cost. DERA is designed for the small large table outer joins and avoid the redistribution of inner join results, but a large amount of physical space are needed to store the temporary results. So how can we take the advantages of them both is a challenge.
- Conventional data warehouses employ the query-at-a-time model, because the physical plans compete for access to the underlying I/O and computation resources. But modern systems can efficiently optimize and evaluate a single complex data analysis query, their performance suffers significantly when multiple complex queries run at the same time. So how to make these outer join algorithms adjust to the multi-process DBMS [16] is another great challenge.

6. Acknowledgements

We want to thank the classmates in No.137 lab of Computer Science and Technology College in Donghua University for their significant suggestions.

7. References

[1] G. Hill and A. Ross, "Reducing outer joins," in VLDB,2009.

[2] A.Ghazal, A. Crolotte, and R. Bhashyam, "Outer join elimination in the teradata rdbms," in DEXA, 2004.

[3] G. Bhargava, P. Goel, and B. R. Iyer, "Efficient processing of outer joinsand aggregate functions," in ICDE, 1996.

[4] A.L. P. Chen, “Outer join optimization in multi-database systems,” in DPDS, 1990.

[5] G. Bhargava, P. Goel, and B. R. Iyer, “Simplification of outer joins,” in CASCON, 1995.

[6] G. Bhargava, P. Goel, and B. R. Iyer, “Hyper graph based reordering of outer join queries with complex predicates,” in SIGMOD, 1995: 304–315.

[7] P.-A°. Larson and J. Zhou, “Efficient maintenance of materialized outer join views,” in ICDE, 2007:56–65.

[8] P.-A°. Larson and J. Zhou, “View matching for outer-join views,” in VLDB, 2005: 445–456.

[9] P.-A°. Larson and J. Zhou, “View matching for outer-join views,” in VLDB 2007:29-53.

[10] A.Gupta, H. V. Jagadish, and I. S. Mumick, “Maintenance and self maintenance of outer-join views,” in NGITS, 1997.

[11] Annita N. Wilschut , Jan Flokstra , Peter M. G. Apers, Parallel evaluation of multi-join queries, Proceedings of the 1995 ACM SIGMOD international conference on Management of data, p.115-126, May 22-25, 1995.

[12] Sumit Ganguly,Waqar Hasan,Ravi Krishnamurthy, Query optimization for parallel execution, Proceedings of the 1992 ACM SIGMOD international conference on Management of data, p.9-18, June 02-05, 1992.

[13] Donovan A. Schneider , David J. DeWitt, A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment, Proceedings of the 1989 ACM SIGMOD international conference on Management of data, p.110-121, June 1989.

[14] Y.Xu, P. Kostamaa. “A new algorithm for small-large table outer join in parallel DBMS,” in ICDE, 2010.

[15] Y.Xu, P. Kostamaa. “Efficient outer join data skew handling in parallel,” in VLDB, 2010.

[16] G.Candea, N.Polyzotis, R.Vingralek.”A scalable, predictable join operator for highly concurrent data warehouses,” in VLDB, 2009.