Complex Query JOIN Optimization in Parallel Distributed Environment

Dr. Sunita M. Mahajan\textsuperscript{1}, Ms. Vaishali P. Jadhav \textsuperscript{2}
\textsuperscript{1}Principal, Computer Science Dept., Mumbai Education Trust, Bandra, Maharashtra, India
\textsuperscript{2}Research Scholar, NMIMS University, Mumbai, Maharashtra, India

\textbf{Abstract} - The research work covers the query optimization concept in parallel distributed environment. The queries considered are select-project-join (SPJ) queries with large databases. The main query operation considered for research is JOIN operation of the query. For fast execution of a complex query, JOIN operation time needs to be minimized. Different JOIN operation algorithms such as Network Byte Order (NBO) parallel two way semi JOIN with bit array, NBO parallel collision free intelligent bloom JOIN filter, NBO parallel positional encoded reduction filter (PERF) JOIN and NBO parallel distinct encoded reduction filter (DERF) JOIN are implemented and evaluated with existing algorithms.

\textbf{Keywords:} Query Optimization, JOIN Operation, Network Byte Order, Parallelization, and Distributed Environment.

1 Introduction

A query is defined as content retrieval from database on demand. It can be as easy as “Retrieving name of the person with PAN card number AAQPW2130D” or more complex like “Finding the amount of EMI of all bank customers whose age is between 30-39 years, having more than 5 years of work experience, having loan amount between 20 lacs to 50 lacs and want to make up loan within 5 years having loan period of 20 years with floating interest”. After carrying out a JOIN operation between query tables, the query results are produced. The JOIN order of the tables decides the performance of the query. Query optimizer determines JOIN order via different JOIN algorithms in diverse environments. Depending upon the algorithmic change in each JOIN algorithm, query JOIN optimization time may vary. The research focus is to reduce the optimization time and network cost (in terms of amount of data to be transferred on network) of JOIN operation of a complex query in large databases.

The following Fig.1 gives an example of a complex query consists of many relations R1-R15. To produce the final JOIN operation result in minimum time, we need to reduce the time required by intermediate JOIN operations. To get the fast result, we parallelize a set of query optimization procedures for improving the performance of JOIN operation in a complex query on large databases in parallel distributed environment [1-17] [18-23] [29-31] [48].

For implementing parallel JOIN algorithms, intra-operator parallelism and inter-query parallelism with shared nothing architecture is used. In intra-operator parallelism, single query JOIN operation is executed and in inter-query parallelism, multiple queries are executed on multiple nodes in distributed environment [18-23] [29-32] [48].

In parallelization, various load balancing schemes are used. The load balancing schemes that we used are round-robin, total-sum, equi-depth and stratified-allocation. The cost of parallel execution of JOIN operation includes the cost of data partitioning, data assembling and maximum execution cost of JOIN operation on multiple nodes. For achieving better results of parallelism, the combination of independent and pipelined parallelism is used. In independent parallelism, each node works independently with the data allotted to it. The pipelined parallelism gathers the results in pipeline and gives the final result [9-11] [42-45].

Our research work gives extra facilities during optimization phase. Query JOIN optimizer converts actual JOIN attribute values into a compressed binary form. But the problem is different CPU platforms may store compressed binary data types differently \textsuperscript{17}, i.e. Little Endian or Big Endian representation. To solve this problem, this compressed binary data is again converted into \textit{network byte order} representation.
which will be an intermediate storage of binary data to be transmitted across network. This NBO data again is encrypted and provides the security during transmission of data on network.

The research work focused on various challenges occurred during optimization of query in different environment such as centralized environment, distributed environment and parallel environment. We focused on parallel environment. Producing the query result in less execution time, reducing communication cost when data is distributed among different sites, preventing data loss during JOIN operation, eliminating duplicate data during data transfer, reducing the amount of data to be transferred using data compression techniques, securing the data during transmission are some of the challenges in our research work.

Based on certain parameters such as execution time, memory utilization, amount of data to be transferred on network, speed and efficiency, the query JOIN optimization algorithms are evaluated.

2 Related Work

The existing JOIN algorithms that we considered for our research are as follows:

- Uncompressed JOIN Optimization
- Compressed JOIN Optimization
- Semi JOIN
- Two Way Semi JOIN
- Bloom JOIN
- PERF JOIN
- DERF JOIN

For research in optimization of complex queries, we studied the existing JOIN optimization algorithms. Existing JOIN optimization algorithms are classified as Uncompressed and Compressed JOIN optimization. We studied semi JOIN and two way semi JOIN optimization techniques. In Uncompressed JOIN optimization, the actual JOIN attribute values are transmitted so it increases the transmission time as well as network data. In two way semi JOIN optimization, the common and uncommon JOIN attribute values are compared but data is still in uncompressed form. We then studied some compressed techniques such as Bloom JOIN, Position Encoded Reduction Filter (PERF) JOIN and Distinct Encoded Reduction Filter (DERF) JOIN. Bloom JOIN faces the collision problem i.e. more than one elements point to the same address. PERF JOIN uses position encoding which is not suitable for large databases and DERF JOIN requires more time to remove duplicates. So our research fills these gaps by using parallelization to improve the performance of JOIN operation.

3 Research Work

We develop four JOIN optimizers in parallel distributed databases which focus on network data reduction, time-memory reduction and security during data transmission on network.
The AES (Advanced Encryption Standard) algorithm. The encrypted data is transferred on network and provides security to original data [40] [42-47] [50]. Thus the problems of data reduction, binary data representation and security issues are solved with above solutions (Data Compression, NBO and AES). The time and memory reduction is achieved by using parallelization concept in query optimization [9-11] [42-45].

The detailed block diagram of our system is given below in Fig. 4.

![Block Diagram and Contribution](image)

The basic block diagram of a system is consists of 3 phases: pre-processing, optimization and query execution as shown in Fig. 3. Pre-processing consists of server status checker module which checks the connection between server and client in distributed environment. If the connection is OK then query generator module generates the query. According to the number of JOIN attributes of query tables, adjacency matrix is created in query pre-processing module. It gives the pre-processing time and query type as its output. Query type defines the number of relations in a query with number of JOIN attributes. For example query type 4-#3 means number of relations is 4 and number of JOIN attributes is 3 in a query.

The optimization phase consists of intra-operator and inter-query parallelism. Intra-operator parallelism consists of parallelization of single operator i.e. JOIN operator on multiple machines. Inter-query parallelism consists of multiple queries executed simultaneously on multiple machines [9-11] [45]. Our four parallel JOIN optimizers are available in both environment i.e. intra-operator and inter-query parallelism. After the data compression and data reduction is over in all parallel JOIN optimizers, compressed data is converted into network byte order (NBO). NBO data is then encrypted with the help of AES algorithm and provides the security during transmission of data on network [43-47] [50].

In execution phase, the query is executed with calculation of time and memory requirement. The row count of reduced relation is also calculated in execution phase [1-5] [19-23]. The queries considered for research are star, chain, cycle and clique. The five databases used for evaluation are TPC-H, Northwind, Pub, AdventureWorks and Query Optimization (customized database used for testing) [13-15]. Databases can be scaled with proper scale factors.

The four JOIN optimizers developed are
- NBO parallel two way semi JOIN with byte array
- NBO parallel collision free intelligent bloom JOIN
- NBO parallel positional encoded reduction filter JOIN
- NBO parallel distinct encoded reduction filter JOIN

3.1 NBO Parallel Two Way Semi JOIN with Byte Array

Original two way semi JOIN calculates the common and uncommon JOIN attribute values and compares the count of both the values. The less count of common or uncommon values are transmitted on network and used further for reduction at resultant site [1-5] [19-23] [42-44] [50] [30-35].

The detailed algorithmic contribution of our first NBO parallel algorithm is given below in Fig 5.

![Algorithmic Contribution of NBO Parallel Two Way Semi Join with Byte Array](image)
During data transmission [12] [42-47] [50]. Parallelization of above optimizer improves the performance of original two way semi JOIN optimizer [9-12] [17] [30-35][38]. We have used intra-operator and inter-query parallelization in our research.

### 3.2 NBO Parallel Collision Free Intelligent Bloom JOIN Filter

Original bloom JOIN filter uses hash function to map JOIN attribute values of relations. The main problem of hash function is collision problem. More than one JOIN attribute values can be mapped to the same address in collision problem [6] [24-28] [36-39].

Instead of using hash functions in bloom JOIN, our algorithm used C# data structure i.e. Hash Table. The problem of collision due to hash functions is avoided by using hash table. Also hash table saves only the distinct values so parallel collision free intelligent bloom JOIN with hash table is useful in large databases and improves the performance of original bloom JOIN optimizer[9-12] [17].

The main contribution of this algorithm is conversion of JOIN attribute values into hash table in forward reduction of JOIN operation. The hash table values are converted into network byte order for solving the problem of binary representation. Security to transmitted data is provided by using AES algorithm. In backward reduction, the less count of common and uncommon values are converted into hash table, then network byte order conversion and then AES algorithm is applied for security purpose.

### 3.3 NBO Parallel Position Encoded Reduction Filter JOIN

Original positional encoded reduction filter (PERF) encodes the position of JOIN attribute value into byte array. Considering JOIN operation between two relations, PERF JOIN sets byte array value equal to 1 for the common JOIN attribute value and sets 0 for the uncommon JOIN attribute value [7-8] [41].

The problem with original PERF JOIN is it encodes position in byte array, as the JOIN attribute values increases, the size of byte array increases. This problem is solved by NBO parallel PERF JOIN which compares common and uncommon JOIN attribute values and instead of encoding all JOIN attribute values, less count of common and uncommon values are used for encoding.

The conversion process of compressed data to NBO data and then NBO data to encrypted data remains same for all optimizers. The PERF JOIN encodes all the values including duplicates.

### 3.4 NBO Parallel Distinct Encoded Reduction Filter JOIN

In PERF join, duplicate JOIN attribute values are also encoded. Duplication increases the cost in terms of response time, transmission time as well as it increases the amount of data to be transferred on network. To solve this problem, original distinct encoded reduction filter (DERF) JOIN selects distinct JOIN attribute values while transmitting data on network. It solves the problem of duplication [7-8] [41].

Our NBO parallel DERF JOIN further reduces the cost in backward reduction phase of JOIN operation. NBO parallel DERF provides security during transmission of data as well as reduces transmission overhead. In backward reduction phase, the distinct common and uncommon JOIN attribute values are compared and less count of these values is represented into byte array for further reduction.

### 4 Experimental Set Up

For experimentation, we consider 3 servers (Configuration: Intel® Core™ 2 Duo Processor E7300, 2.66 GHZ, 3 MB Cache) and 20 Clients (Configuration: 2.9GHz Intel Core i5 processor,4GB DDR3 RAM, 500GB hard drive) in one data center. During experimentation, database of size 500GB is considered and 1500 queries are evaluated. The queries considered for research are star, chain, cycle and clique. The five databases used for evaluation are TPC-H, Northwind, Pub, AdventureWorks and Query Optimization (customized database used for testing) [13-15] [16-17]. Databases can be scaled with proper scale factors.

Different load balancing schemes are used during experimentation such as round-robin, total-sum, equi-depth and stratified-allocation [12]. We considered all the test cases (Best, Average and Worst) during our experimentation. This research paper includes the results of ‘equi-depth’ load balancing scheme with all types of queries in intra-operator and inter-query parallelism.

We have used following parameters for evaluation of our optimizers.

- Execution time
- Memory utilization
- Amount of data to be transferred
- Speed –Up
- Efficiency

With the help of these evaluation parameters, optimizer’s performance is evaluated. All NBO parallel optimizers with all types of queries in intra-operator and inter-query parallelism are shown in results.
5 Results and Discussion

This section includes the results of our first NBO Parallel JOIN optimizer in worst case intra-operator parallel environment. Our algorithms work well with best case and average case for all types of queries.

Our first optimizer i.e. NBO Parallel Two Way Semi JOIN with Byte Array is compared with two existing JOIN optimizers i.e. Parallel JOIN and Parallel Two Way Semi JOIN. This optimizer is evaluated on basis of all evaluation parameters such as execution time, memory utilization, amount of data to be transferred, speed up and efficiency. Table 1 shows the evaluation of our optimizer using execution time.

Table 1 Execution Time

<table>
<thead>
<tr>
<th>Query Type</th>
<th>STAR</th>
<th>CHAIN</th>
<th>CYCLE</th>
<th>CLIQUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time (Sec)</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
</tr>
</tbody>
</table>

NBO Parallel Two way Semi JOIN Optimizer works well with all queries in best and average case. The time required to execute Chain, Cycle and Clique queries is more than the time required to execute existing optimizers in worst case. So our optimizer works well for star queries in worst case scenario. The time required to count common and uncommon JOIN attribute values may take extra time for more complex queries in worst case scenario.

Table 2 Memory Utilization

<table>
<thead>
<tr>
<th>Query Type</th>
<th>STAR</th>
<th>CHAIN</th>
<th>CYCLE</th>
<th>CLIQUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Utilization (MB)</td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
</tbody>
</table>

The memory utilization is also less for Star queries in worst case scenario. Memory required by Chain, Cycle and Clique is somewhat more than the existing optimizers.

Table 3 Amount of Data to be Transferred

<table>
<thead>
<tr>
<th>Query Type</th>
<th>STAR</th>
<th>CHAIN</th>
<th>CYCLE</th>
<th>CLIQUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of Data To Be Transferred (MB)</td>
<td>500</td>
<td>600</td>
<td>700</td>
<td>800</td>
</tr>
</tbody>
</table>

Our NBO parallel optimizers used compressed data during transmission on network. So in all test cases, the amount of data to be transferred is less than existing JOIN optimizers. In worst case also, the amount of data to be transferred on network is less for all types of queries.

We compared all our NBO parallel optimizers with above evaluation parameters. For other two evaluation parameters such as Speed-Up and Efficiency, we compared all four NBO parallel optimizers with existing parallel optimizers.

Table 4 Speed Up

<table>
<thead>
<tr>
<th>No. of Processors</th>
<th>Processor 1</th>
<th>Processor 2</th>
<th>Processor 3</th>
<th>Processor 4</th>
<th>Processor 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed Up (8-36 Query)</td>
<td>2.5</td>
<td>3</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Table 5 Efficiency

<table>
<thead>
<tr>
<th>Processors</th>
<th>Processor 1</th>
<th>Processor 2</th>
<th>Processor 3</th>
<th>Processor 4</th>
<th>Processor 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-Operator Efficiency (%)</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
</tr>
</tbody>
</table>
Our four NBO parallel optimizers are compared with existing two optimizers. Speed Up and efficiency of all optimizers is measured. Among all four optimizers, NBO Parallel Distinct Encoded Reduction Filter gives more speed and is more efficient than other 3 NBO parallel optimizers.

6 Conclusion

NBO parallel optimizers significantly improve the performance of JOIN operation in parallel distributed environment along with reduction in the execution time and amount of data to be transferred on network. The compression techniques such as Byte Array conversion, Hash Table conversion, PERF(Relation) conversion and DERF(Relation) conversion solves the existing issues in JOIN operation. We found ‘Equi-Depth’ is better load balancing scheme among other load balancing schemes. Our evaluation parameter, ‘amount of data to be transferred’ is always acceptable for all queries in all cases.

NBO Parallel Two Way Semi JOIN with Byte Array and NBO parallel PERF JOIN is suggested for star queries. NBO Parallel Collision Free Intelligent Bloom JOIN Filter is suggested for star and chain queries. NBO Parallel DERF JOIN is suggested for all types of queries i.e. star, chain, cycle and clique. NBO Parallel DERF is the best optimizer among all other NBO parallel optimizers.

In future these NBO parallel optimizers can be tested on unstructured databases or column oriented databases.

7 References

[21] G.M. Lohman, C. Mohan, L.M. Haas, D. Daniels, B.G. Lindsay, P.G. Selinger, and P.F. Wilms. “Query...


