DIRTRIBUTED QUERY OPTIMIZATION USING PERF JOINS

Ramzi Haraty, Roula Fany
Lebanese American University
P.O. Box 13-5053
Beirut, Lebanon
rharaty@beirut.lau.edu.lb, rolafany@sodetel.net.lb

ABSTRACT

The advent of telecommunication era and the constant development of hardware and network structures have encouraged the decentralization of data while increasing the needs to access information from different sites. Query optimization strategies aim to minimize the cost of transferring data across networks. Many techniques and algorithms have been proposed to optimize queries. Perhaps one of the more important algorithms is the AHY algorithm using semi-joins that is implemented by Apers, Hevner and Yao. Nowadays, a new technique called PERF (Partially Encoded Record Filters), presented by Kenneth Ross seems to bring some improvement over semi-joins. PERF joins are two-way semi-joins using a bit vector as their backward phase. Our research encompasses applying PERF joins to the AHY algorithm and producing the AHYPERF algorithm. Programs were designed to implement both the AHY and AHYPERF. Several experiments were conducted and the results showed a very considerable enhancement of AHYPERF over the original AHY.

Keywords: Distributed Query Optimization, Semi Joins, and PERF Joins.

1. Introduction

The recent telecommunication boom has encouraged business expansion resulting in the decentralization of data while increasing the needs for instant information access.

A distributed database system (DDBS) is a collection of sites connected on a common high-bandwidth network [5]. Logically, data belongs to the same system but physically it is spread over the sites of the network, making the distribution invisible to the user [6]. Each site is an autonomous database with its processing capability and data storage capacity. The advantage of this distribution resides in achieving performance, reliability, availability, and modularity.

Distributed query processing is the process of retrieving data from different sites. Accessing data from sites involves transmission via communication links that creates delays. The basic challenge is to design and develop efficient query processing techniques and strategies to minimize the communication cost.

Nowadays, with the explosion of interest in data warehouses and the development of huge applications such as federation and mediation over heterogeneous and object-oriented databases, there is a pressing need for data reduction. This is the main purpose of query optimization which estimates the cost of alternative query plans in order to choose the best plan to answer quickly and efficiently, complex and expensive queries [7].

The query optimization problem was addressed many times, from different perspectives, and a lot of work has been done. Proposed algorithms and techniques can be categorized in two main approaches:

1- Minimize the cost of data transferred across the network by reducing the amount of transmitted information, and

2- Minimize the response time of the query by using parallel processing.

Some might add another category which is the hybrid approach, merging both data reduction and time reduction.

In this paper, we will mainly focus on the first approach. One of the most popular and important algorithms suggested for query optimization with minimum cost was algorithm GENERAL (total cost) presented by Apers, Hevner and Yao [2]. The advent of AHY was a revolution in query optimization domain because it introduced semi-joins as reducers in the query optimization process. It uses the three-phased approach method that consists of the following:

- Local processing to filter unnecessary data,
- Semi-join reduction involving shipment of data from one site to another to be reduced, and
- Final assembly at the destination site.

Using local operations such as projections and selections, data was filtered then shipped to other sites. Reduction was made by gradually applying semi-joins while transferring data from one site to another. The improvement made over the traditional unoptimized method was enormous.

A decade after, and with the continuous research and methods developed, a new technique called PERF (Partially Encoded Record Filter) was presented by Kenneth Ross [4]. This method adds to semi-joins another dimension that is the backward phase that will be used to eliminate unnecessary redundant semi-joins by using bit vectors.

In this paper we present an improvement over AHY using PERF joins applied to AHY algorithm. The paper is organized as follows: Section 2 discusses the original AHY algorithm. Section 3 presents our own improvement over AHY - the AHYPERF algorithm. Section 4 presents the experimental results. And section 5 presents the conclusion.

2. The AHY Algorithm

For a special class of queries called simple queries, Apers, Hevner and Yao have developed a collection of algorithms to minimize the cost of query processing. The algorithms developed in were SERIAL and PARALLEL for total time and minimum response time optimization respectively [2]. Then, later they proposed an extended version suitable for general queries called GENERAL. Using this algorithm, data transmissions used for reducing a relation and the transmission of the reduced relation to the assembly site form a schedule for this relation. Throughout this algorithm, different schedules are built and compared until the most optimal one is chosen. Note also that the transmission cost between two computers is a linear function of the size of the data. The incoming selectivity of a schedule for a relation is the product of selectivity of all the attributes in the schedule. As our study mainly focuses on total cost optimization, we will limit our discussion on AHY to SERIAL and GENERAL:

- Algorithm SERIAL: In this algorithm we are trying to construct serial candidate schedules in order to later integrate them with others to form the final schedule if the results are optimal.
- Algorithm GENERAL: This algorithm creates relation schedules by using the candidate schedules created above for each relation and then extracting the best serial candidate schedule for each join attribute to try to integrate using procedure TOTAL or COLLECTIVE.

In other terms, in a distributed database system, it is better to first perform initial local processing in order to reduce the amount of data to be transmitted. After initial processing, if the relations contain only one attribute that is the joining attribute, queries are called simple [1]. Algorithm GENERAL tries to decompose complex queries into simple ones. Then algorithm SERIAL orders relations by increasing order in terms of size and starts creating schedules for the simple queries. The resulting schedules are examined and integrated to form an optimized schedule [3]. For every candidate schedule to relation R_i containing a transmission of a joining attribute from the same relation R_i, the algorithm adds a new candidate schedule without the transmission of this joining attribute. This step is important because a relation cannot be reduced by the selectivity of its own joining attribute. Then, the schedule that minimizes the total time of transmitting R_i, if only one joining attribute, is considered. The selected schedule is called $BEST_{ij}$. Note that procedure TOTAL does not take into consideration the redundant transmissions due to the fact that schedules are constructed separately. This problem is considered in procedure COLLECTIVE that constructs only one basic strategy for the entire query and then tries to do some variations in order to reach the most optimal schedule [1]. In this paper we will discuss algorithm AHY GENERAL Total time version using procedure TOTAL.

2.1 Complexity Analysis of Algorithm GENERAL (Total Time)

The worst case complexity of algorithm GENERAL was proved to be $O(\sigma m^2)$ where σ is the number of different simple queries (i.e., different joining attribute to which algorithm GENERAL is applied) and m is the number of relations in the query.

3. The AHYPERF Algorithm

Partially Encoded Record Filter, is a new two-way semi-join implementation primitive. The basic idea of PERF is as follows - consider two relations *R* and *S*. Apply the following steps to the two tables:

1. Project *R* on a joining attribute and get P_R.
2. Ship P_R to *S*.
3. Reduce *S* by a semi-join with P_R.
4. Send back to *R*,a bit vector (PERF) that contains one bit for every tuple in P_R and in the same order. If the tuple is matching then send 1 else send 0.

The fourth step is known as the backward phase. The main utility of PERF is that it minimizes this phase and hence makes the forward phase (step 2) cost greater than the backward phase. PERF joins can be better enhanced by sending back to *R* not all the bit vector corresponding the P_R but only the 0s part or 1s part according to which one is less in size and hence has lower transmission cost. Figure 1 illustrates the PERF join technique.

R(A,X)

1	a1	10
2	a2	20
3	a3	30
4	a4	40
5	a5	50
6	a6	66

R(X,B)

100	b1	1
30	b2	2
50	b3	3
90	b4	4
20	b5	5
70	b6	6

PERF(*R*)

1	0
2	1
3	1
4	0
5	1
6	0

PERF(*S*)

0	1
1	2
1	3
0	4
1	5
0	6

Figure 1: PERF for *R* and *S*.

When applying PERF to AHY algorithm the following is performed:

1. Perform all initial local processing.
2. Generate candidate relation schedules by isolating the attributes first and then creating simple queries.
3. For each relation R_i,
   a- Use algorithm SERIAL_PERF and create candidate schedules.
   b- Use procedure TOTAL_PERF to integrate candidate schedules.

3.1 Algorithm SERIAL_PERF

1. Order relations R_i such that
   $$S_1 \leq S_2 \leq \ldots \leq S_m$$
2. If no relations are at the result node, then select strategy:
   $$R_1 \rightarrow R_2 \rightarrow \text{- - -} R_n \rightarrow \text{result node}$$
   Or else if R_r is a result at the result node, then there are two strategies:
   $$R_1 \rightarrow R_2 \rightarrow \text{- - -} \rightarrow R_r \rightarrow \text{- - -} R_n \rightarrow R_r$$
   Or
   $$R_1 \rightarrow R_2 \ldots \rightarrow R_{r-1} \rightarrow R_{r+1} \rightarrow \ldots R_n \rightarrow R_r$$
   Select the one with minimum total time.
3. Build a PERF list where PERF $_{R_i R_{i+1} j}$ is set to 1 when transmission was done from R_i to R_{i+1} on join attribute *j*.
4. When calculating transmission cost,
   If PERF $_{R_i R_{i+1} j} = 1$ then
   Cost = 0
   Else
   Cost = $C_0 + C_1 * b_{ik} + (b_{ik} * \rho_{(i+1)k})/8$

where $C_0 + C_1 * b_{ik}$ is the linear function of transmission cost that is equal to the fixed cost per byte transmitted (C_1) multiplied by the size in bytes of the join attribute projected. This is the usual cost of a semi-join known as the forward cost. $(b_{ik} * \rho_{(i+1)k})/8$ is the backward cost that is the cost of transmitting back to R_i the bit vector consisting of only matching values of the corresponding attribute. For simplicity of this equation, we are considering attribute *k* of width 1 byte.

5. Select the strategy with minimum total time.

3.2 Algorithm TOTAL_PERF

1. Add candidate schedules: For each relation and candidate schedule, if the schedule contains a transmission of a joining attribute of the relation then add another similar schedule without the transmission of a joining attribute of the relation.
2. Calculate the cost of the newly added schedules as shown in step 4 of algorithm SERIAL_PERF.
3. Select the best candidate schedule that minimizes the total time for each joining attribute.
4. Update the PERF list: Set to 1 the values corresponding to all transmissions of the $BEST_{ij}$ selected.
5. Candidate schedule ordering: For each relation R_i, order the candidate schedules $BEST_{ij}$ on joining attribute d_{ij} so that,
   $ART_{i1}+C(s_1*SLT_{i1}) \leq . \leq ART_{i\sigma}+C(s_i*SLT_{i\sigma})$
   where ART is the arrival time of the best schedule, SLT is the accumulated attribute selectivity of the best schedule, and *s* is the selectivity of the corresponding relation.
6. Schedule integration: For each $BEST_{ij}$ construct an integrated schedule to R_i that consists of parallel transmission of candidate schedule $BEST_{ij}$ and all schedules $BEST_{ik}$ where $k < j$.

As it can be seen, the PERF version of AHY algorithm does not eliminate redundant transmissions from the schedules but it makes their cost 0 when they occur. This can be made possible by adding a little overhead on the transmission cost that is the backward cost. Using this fact, if a transmission was done from site *A* to site *B* using a join attribute *j*, then every other transmission from *A* to *B* using *j* will have a zero cost and every transmission from *B* to *A* using *j* will have also a zero cost. From this point, a PERF join can be seen as a non-redundant symmetric function. This fundamental property allowed us to enhance over the traditional AHY GENERAL Total Time.

We note that the reduction effect of PERF is proportional to the width of the attributes used. In section 5, we show results from different width selections to clarify this issue.

3.3 Complexity Analysis of the AHYPERF Algorithm

As far as complexity is concerned, there was not a considerable increase in the complexity of AHY algorithm since data will be still scanned in the same way and for the same number of times. Ordering will also be done in the same fashion. What is added is only the maintenance of the PERF list. According to its implementation, PERF list could be very easily maintained and with minimum complexity time. In our case, PERF list was implemented as a three-dimensional array. So globally, and without loss of generality, we can assume that PERF version of AHY algorithm takes no more than O (σm^2) where σ is the number of different simple queries, and m is the number of relations in the query.

4. Experimental Results

Different scenarios were conceived in order to evaluate the performance of the different algorithms and for each scenario programs were run 1500 times. Different kinds of results are collected:

1. Comparison of all algorithms versus the unoptimized method, and
2. Comparison of algorithms versus AHY GENERAL Total Time algorithm.

4.1 Scenario 1:

In this scenario the attribute width is taken as 1 byte for all attributes.

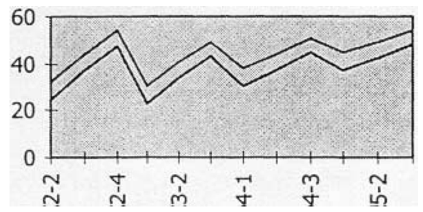
- Comparison versus unoptimized method:

TYPE	AHY	AHYPERF	AHYPERF /AHY
2-2	24.61	31.70	7.09
2-3	37.31	43.86	6.55
2-4	47.13	53.79	6.67
3-2	22.58	30.25	7.68
3-3	34.00	40.36	6.37
3-4	43.10	48.90	5.80
4-2	30.38	37.87	7.50
4-3	37.04	43.70	6.66
4-4	44.61	50.55	5.94
5-2	37.25	44.92	7.67
5-3	42.27	49.01	6.75
5-4	48.18	54.37	6.19
TOT:	37.37	44.11	6.74

- Comparison versus AHY Algorithm:

TYPE	AHYPERF/AHY
2-2	9.59
2-3	10.82
2-4	13.08
3-2	10.15
3-3	10.17
3-4	10.97
4-2	11.33
4-3	11.36
4-4	11.59
5-2	13.31
5-3	12.91
5-4	13.27
TOT:	11.54

Graphically, the results are represented as follows: comparing AHYPERF to AHY: we notice that AHYPERF outperforms AHY in all cases.



4.2 Scenario 2:

In this scenario the attribute width is taken as 50 bytes for all attributes.

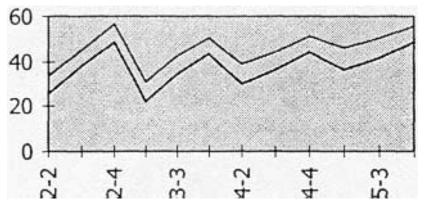
- Comparison versus unoptimized method:

TYPE	AHY	AHYPERF	AHYPERF /AHY
2-2	25.29	33.22	7.93
2-3	37.53	45.27	7.74
2-4	48.39	56.21	7.83
3-2	22.08	30.98	8.90
3-3	34.34	41.98	7.64
3-4	42.81	50.02	7.21
4-2	29.81	38.92	9.11
4-3	36.27	44.01	7.75
4-4	43.99	51.33	7.34
5-2	36.49	46.24	9.75
5-3	41.75	50.01	8.26
5-4	48.35	55.89	7.54
TOT:	37.26	45.34	8.08

- Comparison versus AHY Algorithm:

TYPE	AHYPERF/AHY
2-2	10.86
2-3	12.81
2-4	15.59
3-2	11.71
3-3	12.32
3-4	13.39
4-2	13.74
4-3	13.18
4-4	14.19
5-2	16.63
5-3	15.64
5-4	16.30
TOT:	13.86

Graphically, the results are represented as follows: comparing AHYPERF to AHY: we notice that AHYPERF outperforms AHY in all cases.



We used two different scenarios in order to study the performance of the above mentioned algorithms from different perspectives. For each scenario, we compared the performance of the algorithms with respect to each other and with respect to the unoptimized solution. Using different scenarios we studied better the behavior of all algorithms under a variety of circumstances. We could be able to note that AHYPERF has the best performance for a field width of 50 bytes. This result was expected because of the overhead added by PERF to the backward phase. Remember that PERF concept consists of returning back to the original site a bit vector representing the matching tuples. This overhead is somehow more considerable when the original field width is <= 1 byte because it might be more profitable sometimes not to send back this data. But when having a width of 50 bytes, the backward cost becomes negligible as compared to the forward cost.

Finally, we can conclude that the results of our experiments were up to the expectations and proved the power of PERF joins and their advantage in optimizing the total time of distributed queries.

6. Conclusion

In this paper, we have fully exposed both concepts of semi-joins and PERF joins and then, we have taken an optimization algorithm using semi-joins (AHY) and enhanced it by applying PERF joins (AHYPERF). In addition, we have discussed the advantages of PERF joins over semi-joins which mainly consist of removing the cost associated with redundant transmissions by adding a relatively negligible cost to the backward phase of each PERF join.

Experimental results confirmed our expectations by showing a considerable enhancement over the AHY algorithm. Different series of experiments were conducted, allowing us to study even better the efficiency of PERF joins from different perspectives and to consider the best case for which PERF joins perform at most. We could then, based on our experiments recommend the use of PERF joins for huge textual and graphic distributed databases where the width of some join attributes is quite large, as well as for ordinary data.

References

[1] Alan R. Hevner, O. Qi Wu and S. Bing Yao, *"Query Optimization on Local Area Networks"*, ACM Transactions on Office Information Vol.3, No.1 Pages: 35 - 62, January 1985.

[2] Peter M.G. Apers, Alan R. Hevner and S. Bing Yao , *"Optimization Algorithms For Distributed Queries"*, IEEE Transactions On Software Engineering, Vol. Se-9, No.1, Pages: 57-68, January 1983.

[3] Todd Bealor, *"Semi-join Strategies For Total Cost Minimization In Distributed Query Processing"*, Master Thesis, University of Windsor, Canada, 1995.

[4] Zhe Li, K. A. Ross, *"PERF Join: An Alternative to Two-Way Semi-join and Bloomjoin"*, Columbia University, New York, 1995.

[5] Li-Yun Chen, *"An Algorithm For Distributed Query Processing In a Fiber Distributed Data Interface, High-Bandwidth, Token Ring Local Area Network"*, Master Thesis, North Dakota State University, May 1990.

[6] R. El-Masri , S. Navathe, *"Fundamentals of Database Systems"*, Second Edition, Addison Wesley, 1994.

[7] D. Barbara, W. DuMouchel, C. Faloustos, P. J. Haas, J. M. Hellerstein, Y. Iaonnidies, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross and K. C. Sevcik, *"The New Jersey Data Reduction Report"*, Bulletin Of The Technical Committee On Data Engineering, Pages: 3-45, December 1997.

Biography

Ramzi A. Haraty is an Assistant Professor of Computer Science at the Lebanese American University in Beirut, Lebanon. He received his B.S. and M.S. degrees in Computer Science from Minnesota State University - Mankato, Minnesota, and his Ph.D. in Computer Science from North Dakota State University - Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 35 journal and conference paper publications.

Roula Fany is currently with the Beirut Riyad Bank. She received her Maters of Science degree in Computer Science from the Lebanese American University.