Proceedings of the ISCA 15th International Conference on Computers and Their Applications (CATA-2000). New Orleans, Louisiana, USA March 29 – 31, 2000.

A PERF Solution For Distributed Query Optimization

Ramzi A. Haraty and Roula Fany
Lebanese American University
P.O. Box 13-5053
Beirut, Lebanon

Abstract
Query optimization techniques aim to minimize the cost of transferring data across networks. Many techniques and algorithms have been proposed to optimize queries. One of the algorithms is the W algorithm using semi-joins. Nowadays, a new technique called PERF seems to bring some improvement over semi-joins [2]. PERF joins are two-way semi-joins using a bit vector as their backward phase. Our research encompasses applying PERF joins to the W algorithm. Programs were designed to implement both the original and the enhanced algorithms. Several experiments were conducted and the results showed a very considerable enhancement obtained by applying the PERF concept.

Keywords
Query Optimization, Semi-Joins, and PERF Joins.

1 – Introduction

Distributed query processing is the process of retrieving data from different sites. Accessing data from different 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. 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 [3].

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.

In this paper, we will mainly focus on the first approach. One of the most important algorithms suggested for query optimization with minimum cost was algorithm GENERAL (total cost) presented by Apers, Hevner and Yao in 1983 [4]. The advent of AHY was a revolution in query optimization domain because it introduced semi-joins as reducers in the query optimization process.

In 1995, Todd Bealor from Windsor University, Canada presented a new algorithm called W algorithm as an enhancement over AHY. At the same time, a new technique called PERF (Partially Encoded Record Filter) was presented by Kenneth Ross [2]. This method adds to semi-joins another dimension, which 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 W algorithm using PERF joins applied to W. This paper is organized as follows: Section 2 presents the W algorithm. Section 3 discusses our contribution in the PERFW algorithm. Section 4 presents the experimental results. And section 5 concludes the paper.

2 - The W Algorithm

The main aim of this algorithm is to minimize total time by using reducers in order to eliminate unnecessary data. This algorithm is characterized by two distinct phases:

Phase 1. Semi-join schedules for constructing each reducer are formed using a cost/benefit analysis based on estimated attribute selectivity and sizes of partial results.

Phase 2. Schedule is executed.

Algorithm W works as follows:

1. Establish schedules for the construction of reducers. For each join attribute *j* construct schedule for the reducer d^*_{mj}. It should be noted that at this level, each schedule is considered independently. Hence, no semi-joins are executed yet. This is achieved in two phases:

Phase 1. Sort attributes by increasing size such that: $S(d_{aj}) \leq S(d_{bj}) \leq - - - \leq S(d_{mj})$.

Proceedings of the ISCA 15th International Conference on Computers and Their Applications (CATA-2000). New Orleans, Louisiana, USA March 29 – 31, 2000.

Phase 2. Evaluate semi-joins in order beginning with $d_{aj} \ltimes d_{bj}$. Append semi-join to schedule if:

a. It is profitable and marginally profitable. $P(d_{aj} \ltimes d_{bj}) > 0$ and $MP\ (d_{aj} \ltimes d_{bj}) > 0$ or,

b. It is gainful but not profitable. Hence, $P(d_{aj} \ltimes d_{bj}) < 0$ but $G\ (d_{aj} \ltimes d_{bj}) > 0$.

If semi-join is appended then $d^*_{bj} \ltimes d_{cj}$ is evaluated next, else $d^*_{aj} \ltimes d_{cj}$ is considered.

Repeat this process until all semi-joins in the sequence are evaluated. The last attribute in the sequence will be called the reducer.

2. Examine the effects of reducers. Consider the reduction effects of the reducers' all-applicable relations by:

a. Sorting reducers from smallest to largest.

b. Estimating the cost and benefit of a semi-join with each admissible relation and for each reducer. Profitable semi-joins are appended to the schedule.

3. Review of unused semi-joins. For non-profitable reducers, reexamine the possibility of having profitable semi-joins for that particular join attribute. This phase is done using the following sub-steps:

a. Sort attributes by increasing size.

b. Evaluate each semi-join and append profitable semi-joins to the final schedule. Note that marginal profit is not considered in this step.

4. Execute the schedule. During this phase, reducers are constructed and shipped to designated sites to reduce the corresponding relations. Then, reduced relations are shipped to the assembly site.

This heuristic is simple and efficient. It aims to construct in the cheapest possible way, reducers who are highly selective. Those reducers will be then used to eliminate tuples from participating relations prior to shipment to the query site (assembly site).

It should be noted that algorithm W ameliorates the choice of join attributes and their order but does not eliminate redundant transmissions because schedules are also treated separately.

3 - The PERFW Algorithm

When applying PERF to the W algorithm, the same concept is preserved but semi-joins are replaced by PERF joins. Our enhancement consisted of the following two phases that were added to the schedule construction:

a. 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*.

b. When calculating transmission cost,

```
If $PERF_{R_i R_{i+1} j} = 1$ then
  $Cost = 0$
Else
  $Cost = C_0 + C_1 * b_{ik} + (b_{ik} * ?_{(i+1)k})/8$
```

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, and $(b_{ik} * ?_{(i+1)k})/8$ 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.

As it can be seen, the PERF version of W algorithm does not eliminate redundant transmissions from the schedules but it makes their cost zero when they occur. This can be made possible by adding a little overhead on the transmission cost, which 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 W algorithm.

4 - Experimental Results

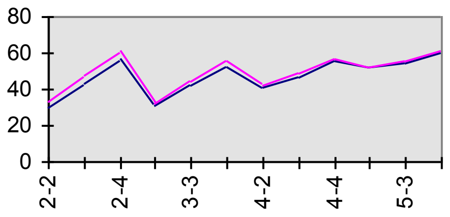
Different scenarios were conceived in order to evaluate the performance of the different algorithms and for each scenario programs were run 1500 times. Different kinds of results are collected including the comparison of all algorithms versus the unoptimized method.

Note that all programs were developed using Visual C++ 4.0 under Windows 95. Experiments were conducted on a Pentium V PC with 64 MB RAM.

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

TYPE	W	PERFW	PERFW /W
2-2	29.79	33.24	3.45
2-3	43.88	47.98	4.11
2-4	56.18	60.63	4.45
3-2	30.63	32.64	2.04
3-3	41.67	44.35	2.69
3-4	52.36	55.32	2.96
4-2	41.45	42.31	0.86
4-3	47.14	48.64	1.50
4-4	55.35	57.12	1.77
5-2	51.74	51.99	0.25
5-3	54.63	55.37	0.74
5-4	60.08	61.14	1.05
TOT:	47.07	49.23	2.15

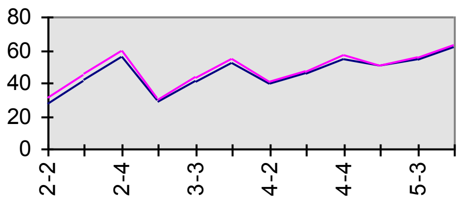
Graphically, the results are represented as follows: comparing PERFW to W: we notice that PERFW outperforms W in all cases.



In the second test scenario the attribute width is taken as 5 bytes for all attributes.

TYPE	W	PERFW	PERFW /W
2-2	27.56	31.12	3.56
2-3	42.31	46.41	4.10
2-4	55.25	59.74	4.49
3-2	28.62	30.74	2.12
3-3	40.63	43.27	2.64
3-4	52.15	55.10	2.94
4-2	40.35	41.17	0.81
4-3	45.54	47.13	1.59
4-4	54.95	56.80	1.385
5-2	50.85	51.16	0.31
5-3	55.11	55.87	0.76
5-4	61.46	62.48	1.02
TOT:	46.23	48.41	2.18

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

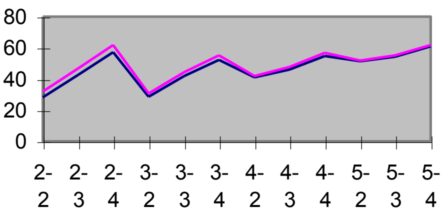


In the third test scenario the attribute width is taken as 50 bytes for all attributes.

TYPE	W	PERFW	PERFW /W
2-2	28.24	31.81	3.57
2-3	42.73	46.75	4.02
2-4	57.23	61.60	4.37

3-2	28.78	30.85	2.07
3-3	41.67	44.42	2.75
3-4	52.03	54.94	2.92
4-2	40.87	41.68	0.82
4-3	46.10	47.56	1.45
4-4	54.76	56.61	1.85
5-2	51.48	51.76	0.28
5-3	54.42	55.23	0.81
5-4	60.96	61.96	1.00
TOT:	46.60	48.76	2.16

Graphically, the results are represented as follows: comparing PERFW to W: we notice also that PERFW outperforms W in all cases.



We used many different scenarios in order to study the performance of the mentioned algorithms from different perspectives. For each scenario, we compared the performance of the algorithms with respect to each other. Using different scenarios we studied better the behavior of all algorithms under a variety of circumstances. We could be able to note that PERFW 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 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.

5 – Conclusion

In this paper, a PERF join algorithm has been presented as our contribution to the query optimization problem using semi-joins. We have fully exposed both concepts of semi-joins and PERF joins and then, we have taken an optimization algorithm using semi-joins (W) and enhanced it by applying PERF joins (PERFW).

6 – References

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

[2] Zhe Li, K.A. Ross, “PERF Join: An Alternative to Two-Way Semi-Join and Bloomjoin”, Columbia University, New York, 1995.

[3] 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.

[4] 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.

[5] Roula Fany, “PERF Solutions for Distributed Query Optimization”, Masters Thesis, Lebanese American University, September 1999.