Query Processing in Connectivity-Challenged Environments*

P. Puri **S. Chakravarthy** **G. Poornima** **M. Kumar**
IT Laboratory and Department of Computer Science & Engineering
The University of Texas at Arlington, Arlington, TX 76019.
Contact Email: sharma@cse.uta.edu

Abstract

This paper presents a discussion of issues that need to be addressed for processing relational queries in distributed environments where the connectivity is unstable and is changing continuously. Query processing over data across nodes that are collecting information is an example of such an environment. Traditional distributed query processing techniques need to be extended to accommodate new requirements.

1. Introduction

Query processing and optimization [1, 2, 3, 4, 5, 6]has been studied in the centralized database management system (DBMS) context and is well established. Similarly, distributed query processing has also been studied using semi-join and other extensions [7, 8, 9, 10, 11]. However, in distributed environments where the data accumulation by each node is dynamic and where connectivity among nodes is changing unpredictably, query processing poses some additional challenges. Replication data item and whether it is complete or not from the viewpoint of query processing becomes very important. Static plan generation does not make much sense. Accurately estimating the result sizes may entail communication with other nodes during plan generation stage.

2. Context

Figure 1 shows a number of aerial vehicles that are acquiring data (over a period of time) and have intermittent connectivity with other aerial vehicles. If a query is sent to one of the nodes that involves join or other computation involving data from other nodes, a query plan needs to be generated to process the query in a distributed data setting to compute the result. As the connectivity changes based on a number of parameters (speed, direction of motion, obstructions, etc.), the challenge is to generate a "good" plan and execute it in this environment with good response time and using complete data available. The dotted lines show change in connectivity along the time dimension.

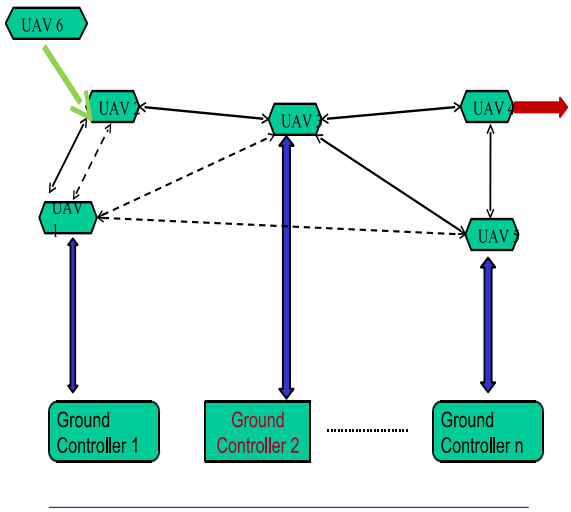

Figure 1. Distributed, Connectivity-challenged Scenario

It is also assumed that the nodes can be of different types based on processing capacity, storage, types of data it can collect, up/down link bandwidths, and latencies of data transfer. Nodes can also play different roles (depending upon the resources available onboard): i) collect data and forward it, ii) collect data, processes it, and forward both collected and processed data, and iii) collect, process, store/hold, and forward data. A single node can play different/multiple roles for different types of data. The roles may change over time.

*This work was supported, in part, by the AFRL Contract FA8750-09-2-0199

3. Architecture

The proposed long-term architecture for handling computations for the scenario shown in Figure 1 is elaborated in Figure 2.

The architecture indicates several capabilities that will make the system complete with respect to query computation over distributed data, Fault tolerance with respect to connectivity and accessibility/reachability of data, and the specific context information as needed by the SOA. However, this position paper mainly addresses the query processing challenges.

4. Query Processing

The overall architecture includes a middleware in each node that has a number of services (based on SOA) for Collecting, Managing, Replicating data and Meta data for the purposes of routing and query processing. Briefly, the following services are used:

1. Message passing and management of meta data
2. Meta data maintained at each node
3. Query plan format and query execution
4. Replica determination and management

Currently, replication is assumed to be single copy and complete for each relation (i.e., data in each node). This needs to be extended to multi-copy and partial replica to accommodate intermittent connectivity. This extension will add complexity to both Meta data management and query processing.

4.1 Query Plan

A query plan for the above scenario is envisioned as numbered sequence of steps that can be easily interpreted and executed at any node. Table 2 gives the plan format for each operator. Each step includes the operation to be applied, the data and the node where it is applied, and the name of the result and the node where it is stored. Table 1 provides all the operators needed for processing a select-project-join (SPJ) query in this environment. A plan counter is initialized to 1 and is incremented until all the plans steps are executed. When the last step of the plan is executed, query execution is over and the result will be in as many nodes as indicated in the query.

Unlike traditional query processing, the plan needs to be sent from node to node for the purposes of query processing. A plan counter indicates the next step of the plan to be executed. The format of the plan is indicated below.

The copy instruction retains a copy of the relation in the node whereas the move instruction does not retain a copy of the relation when moved to another node. This is needed if the result of a query need to made available to multiple nodes. The above format is sufficient to describe any arbitrary relational query plan involving selects, projects, and joins (also known as an SPJ query). The above format can also accommodate SQL aggregate operators, such as a SUM, COUNT, AVG, MIN, and MAX. The query execution proceeds as follows. The plan is sent to the node in which the first operation takes place (if it different from the node where the query plan is generated). The interpreter in that node uses the plan counter to execute as many steps as possible in that node. When a move or copy is encountered, it send the data as well as the plan (can be the remaining plan) to that node. This process continues until the last step of the plan is executed. The result is in the last node where an operation is executed (or moved). Currently, the final query plan is generated as follows. Each node in the architecture has the same query plan generator and uses only the Meta data in that node. The query plan is constructed one join at a time. Cost computations of partial plans are done using the statistics and formulas for computing selectivity of joins and semi-joins. The lowest cost query plan is used as the final plan after all possible plans are explored. The complexity of the current plan generation is k^n where n is the number of joins and k is the number of alternatives for each join. Currently, k being used is about 14 (multiple join alternatives, multiple semi-join alternatives and the same using replica as well) and we are assuming no more than 3 to 5 joins in a query. Even with this assumption, we will explore a large number of alternative query plans and cost computation for each of them. We will incorporate some heuristics to limit the number of plans carried forward after each join. Simulations will be performed to validate the heuristics to make sure they are meaningful. Statistics in the form of cardinality and domain characteristics are used for cost estimation. Join and condition selectivity are inferred from the statistics maintained. Result sizes are also estimated and its accuracy is important as choice of the best query plan is primarily based on the cost of data transfer based on availability of connectivity. These need to be improved using simulation and analysis.

4.2 Challenges

Static plan generation is not the best approach for our situation as the connectivity and hence availability of data can change significantly over short periods. It is entirely possible that the connectivity has changed from the time the plan has been generated to the time a particular step of the query plan is executed. In this case, a new plan from that point of query execution has to be regenerated. Below, we identify a few approaches that are better suited for this scenario.

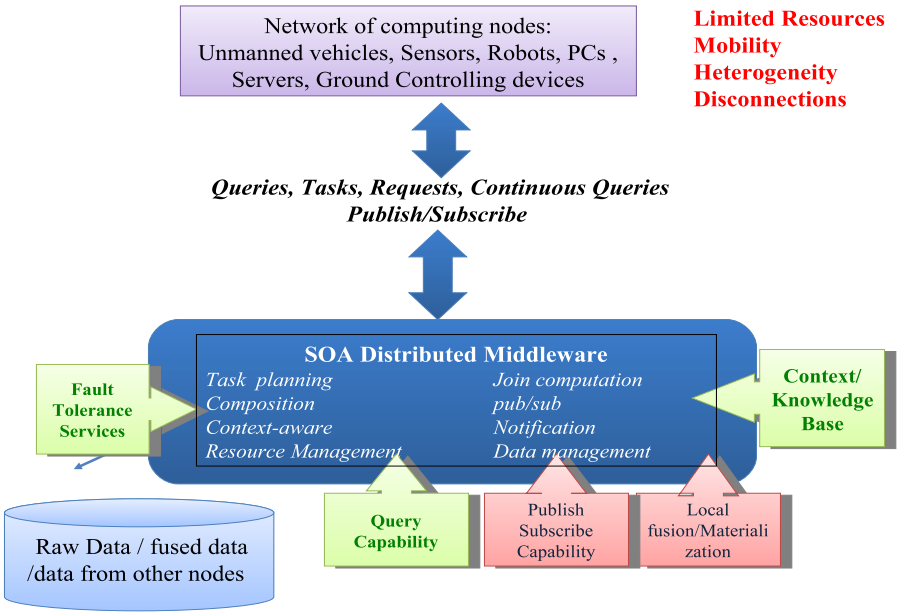

Figure 2. Proposed Long-term Service Oriented Architecture

Operation	Parameter	Opd 1	Opd 1 Loc	Opd 2	Opd 2 Loc	Result Name	Result Loc

Table 1. Plan Format

4.2.1 Dynamic plan generation

One alternative is to generate one join execution plan at a time. At the end of the execution of each join, a plan for executing the next join is generated. This approach, though proposed earlier in Ingres but not used by any commercial system, may be a viable one for our scenario. This also has the advantage of using the cardinality and other statistics that are more accurate than the ones used in a static plan.

4.2.2 Parallel Execution of local operations

Another alternative is to execute *all* local operations in each node in parallel before generating any join plans. At the end of this step, dynamic plan generation approach is used. This approach can also benefit from accurate cardinality and other statistics. This improves over the dynamic plan in that multiple local computations can be done concurrently thereby improving response time. However, the execution of the plan requires more coordination and more message passing as compared to the previous approaches.

4.2.3 Interactive Plan generation

In this approach, each node where a local operation or a join needs to be executed can be queried with respect to the actual cardinality and other statistics information before generating the plan for each join. This approach will provide accurate information due to which a better sequence of join plans can be generated. This combines the advantages of parallel plan execution with that of a dynamic plan generation without making the plan execution and synchronization further complicated.

In addition, all approaches can benefit from sophisticated estimation of selectivity, and cardinalities of intermediate results (such as histograms, distribution, caching of statistics from previous executions, etc.)

4.2.4 Replication

Currently, replication is assumed to be single copy and complete for each relation. Completeness assumption may be difficult to accomplish die to connectivity issues. Single-copy assumption may make the data availability difficult. In order to overcome both of these issues, multi-copy and partial replication can be used. However, extending query processing to include multi-copy and partial replica will add complexity to both Meta data management and query processing.

5. Conclusions

We have highlighted the differences between query processing in a distributed environment and a distributed environment in which connectivity is further subject to

Operation	Parameter	Opd 1	Opd 1 Loc	Opd 2	Opd 2 Loc	Result Name	Result Loc
Select	$A > 100$	R_1	1	Null	Null	R_{11}	1
Project	A_1, A_3, A_6	R_{11}	1	null	null	R_{12}	1
Move	Null	R_{12}	1	null	null	R''	4
Copy	Null	R_{12}	1	null	null	R_14	4
Semi Join	$A = C$	R''	2	R_2	2	SR_1	2
Join	$B = D$	R_{12}	2	R_{22}	2	JR_1	2

Table 2. Plan format of each operation

breakdown. We have analyzed the advantages and disadvantages of various query processing alternatives. Currently, we are developing plan generation and query execution details. Furthermore, replication and is concomitant meta data management issues are also being investigated.

References

[1] P. Selinger *et al.*, "Access path selection in a relational database management system," in *Proceedings 1979 ACM SIGMOD International Conference on Management of Data*, Jun. 1979.

[2] J. D. Ullman, *Principles of Database Systems, 2nd Edition*. Computer Science Press, 1982.

[3] M. Stonebraker, Ed., *Readings in Database Systems*. Morgan Kaufman Inc., 1988.

[4] S. Chakravarthy, "Multiple Query Optimization: Organization of the Strategy Space and the Generation of Shared Multi-strategies, Technical Report, XAIT-89-01," Xerox Advanced Information Technology, Cambridge, Tech. Rep. 177, Mar. 1989.

[5] S. Chakravarthy, "Divide and Conquer: A Basis for Augmenting a Conventional Query Optimizer with Multiple Query Processing Capabilities," in *Proc. of the 7th Int'l Conf. on Data Engineering, Kobe, Japan*, Apr. 1991, pp. 482–490.

[6] R. Ramakrishnan, *Database Management Systems*. WCB/McGraw-Hill, 1998.

[7] Robert Epstein, *Analysis of Distributed Database Processing Strategies*. Electronics Research Laboratory, College of Engg. University of California, 1980.

[8] S. Ceri and G. Pelagatti, *Distributed Databases - Principles and Systems*. McGraw Hill, 1984.

[9] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[10] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys*, vol. 22, no. 3, pp. 184–236, September 1990.

[11] G. M. Sacco and S. B. Yao, "Query optimization in distributed database systems, Tech. Rep. Working Paper MS/S 81 - 029, 1981.