QoS-Aware Distributed Query Processing*

Haiwei Ye
Université de Montréal
Département IRO
ye@iro.umontreal.ca

Brigitte Kerhervé
Université du Québec à Montréal
Département Informatique
Kerherve.Brigitte@uqam.ca

Gregor v. Bochmann
University of Ottawa
SITE
bochmann@site.uottawa.ca

Abstract

In the environment of wide-area networks such as the Internet, distributed query processing becomes problematic due to the changing coming from both underlying networks and user's requirements. In this new context, conventional query processing strategies with the homogeneous assumption will not work well, because they are unable to adapt to unexpected changes in the performance of the communication networks. In this paper, we address the issue of how to make distributed query processing be aware of these changes. We introduce the idea of integration distributed query processing with Quality of Service (QoS) management and accordingly illustrate our view of QoS-aware distributed query processing.

1. Introduction

Quality of Service (QoS) management and database systems (DBS) are two key functional units that contribute to the success of emerging distributed multimedia applications, such as electronic commerce [2] or news-on-demand [14]. However, these applications are not the simple combination of database and network technologies, since they require higher performance and real-time constraints. These new requirements change some of the underlying assumptions and thereby pose new challenges. Among them, the integration of QoS management and DBS appears of prime interest to meet requirements arising in wide-area distributed multimedia systems.

In such environments, we have to answer the fundamental question traditionally addressed by the database community - how to efficiently manage and query large volumes of widely distributed data? The problem is still the same, but the situations become more complex: data access over wide-area networks involves a large number of remote data sources, intermediate sites, and communication links, all of which are vulnerable to congestion and failures. Query processing is now placed in the context of wide-area, distributed, and dynamic environments instead of local, homogeneous and static environments, and thus it becomes problematic due to the changing coming from both underlying system and user's requirements. The changing in underlying system introduces problems such as unpredictable nature of the communication network and lack of knowledge about the load and potential delays at remote end-system. On the other hand, user's requirements and expectations from the service provider are varying in terms of accuracy of the result, the cost to be charged or the response time.

In this new context, conventional query processing strategies[6, 5, 10] with the homogeneous assumption will not work well, because they are unable to adapt to unexpected changes in the performance of the communication networks. The limitation of this assumption becomes manifest in their consideration of communication cost: constant throughput and no delay consideration. In addition, traditional query optimizers aim at achieving one of the two optimization criteria: reduce response time or minimize total resource consumption, in

* This work was supported by a grant from the Canadian Institute for Telecommunication Research (CITR), under the Network of Center for Excellence Program of the Canadian Government.

other words they do not consider different user requirements. However, in most distributed multimedia applications today, this single-criteria optimization is much less plausible and flexible. Since user's concerns vary in different applications and thus lead to other optimization criteria. For example, some user might care more about the service charge he can afford. Therefore, the optimizer should also take different optimization goals into account and the extension and revision of traditional query processing strategies are required.

We believe one feasible way to capture these changes is to resort to QoS management [1, 3, 4, 8, 9, 11, 13], because QoS management aims at deciding and controlling if and how data streams can be delivered to the user within the given delay, cost or quality constraints. The helpfulness of QoS management lies in the fact that it can acquire the user's requirements and gather network performance-related parameters dynamically, including such as end-to-end delay, bandwidth, and maximum packet size supported by the underlying network. We consider in our work the following two quality criteria, as seen by the user: (1) cost of a service request, and (2) total delay for obtaining the response. Depending on the usage context, one or the other of these criteria may be more important to the user.

The rest of the paper is organized as 3 sections. Section 2 presents an overview of distributed query processing and a motivation example. Next, in Section 3, we present our QoS-based query processing. And finally, Section 4 concludes this paper.

2. Review of distributed query processing and a motivation example

Distributed query processing has attracted a lot of research attention in the last two decades. These efforts essentially concentrate on proposing strategies and algorithms to minimize response time while minimizing resource consumption. In new environments imposed by the Internet, such approaches have to be reevaluated to fit for the changing conditions. In this section, we first present a short review of distributed query processing and then give a motivation example.

2.1. Review of distributed query processing

Optimizing a distributed query implies that an optimization algorithm creates a distributed query plan composed of relational operators, data transfer operations and location information for execution. In centralized database systems, the primary criterion for measuring the cost of a particular strategy is the number of disk accesses. In a distributed system, we must take into account several other matters, including the cost of data transmission over the network. The relative cost of data transfer over the network and data transfer to and from disk varies widely depending on the type of network and on the speed of the disks. Thus in general, we must find a good tradeoff between the disk cost and the network cost.

Generally, a query can be executed in many ways, and different database systems may follow different steps. The authors in [7] introduced a *two-phase* view of the query optimization phase; in [12], more general steps were given. In Figure 1, we summarize and synthesis them as *representative* steps, meaning that not all database systems exactly follow these steps.

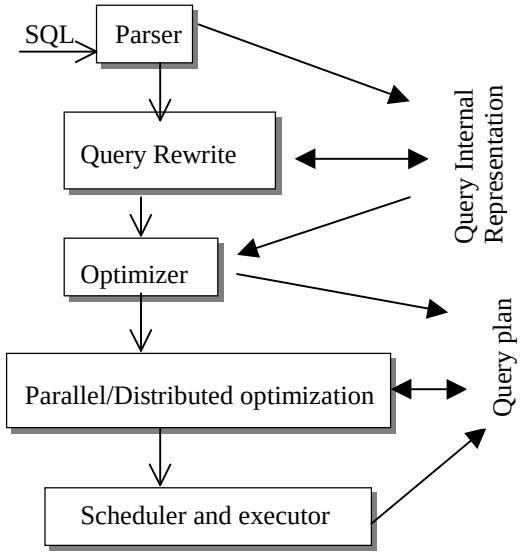

Figure 1. Query Processing Steps

The user's query is first parsed, syntactically analyzed, schema validated against the database schema (checking if tables and attributes really exist), and translated into a chosen internal representation, often in a graph form. Then output of the parser is transformed by a set of rewrite rules. These rewrite rules are usually heuristics aiding in transforming the query into a semantically equivalent form that may be processed more efficiently. Based on the statistic information and the cost model, the optimizer has the responsibility of generating a query evaluation plan that minimizes the most relevant performance measure, which could be the query response time, total resource usage, a combination of the two or some other performance measure. In the case of distributed/parallel query processing, the optimizer should further consider the data distribution information and the available processors for a possible parallel execution. For example, the allocation of processors to each operation and the data transfer direction should also be decided in this stage. And last, the scheduling information is associated with this query plan.

Communication is an important component of distributed database processing since data need to be transferred from one site to another. However, existing

approaches proposed in literatures to evaluate the communication cost are very simple and ideal: it is only a function of the amount of data transmitted. This is obviously not enough when the underlying network has a very dynamic nature. Thus, other important network performance metrics, such as current delay and available throughput, should not be neglected.

2.2. A motivation example

In this section, we give a motivating example to show how the network delay affects the choice of query plan. In the example we assume the optimization objective of the optimizer is *to minimize the response time.* Consider a join between two tables, T_1 and T_2, stored at two different sites, node A and node B, as shown in Figure 4.1. Suppose the table sizes for T_1 and T_2 are 1K bytes and 100K bytes, respectively. Also assume that the available throughput among three links is the same, say 1MB/s, the current delays between CA, CB, AB are 0.1s, 0.1s and 1s respectively.

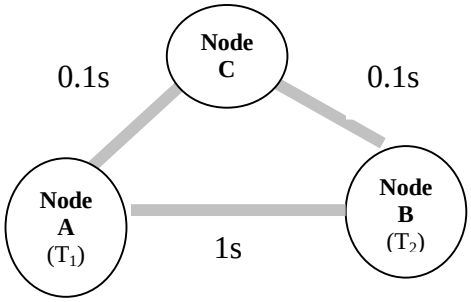

Figure 2. An example

To make a join between T_1 and T_2 , depending on where the join should be executed, three possible strategies can be used:

1. Make join at node A, thus the data from node B should be shipped to A;
2. Make join at node B, thus the data from node A should be shipped to B; or
3. Make join at node C, which means that data from both node A and B should be transferred to node C.

A traditional optimizer (with the assumption of constant throughput and no delay consideration) may choose strategy 2, because when considering the amount of data to be transferred, this is the 'least cost' strategy. However, if we take delay into account (with the help of QoS management), strategy 3 is obviously a better choice, because this may reduce the transfer time and thus the response time.

This example is only the simplest situation in a distributed system, but convincing enough to make the optimizer select a different query execution strategy. Thus, to keep track of the current dynamic performance information about the underlying network, it is important for the optimizer to be customized to various environments and application requirements. In particular, the cost models should provide a more precise evaluation of the query execution strategies in order to adapt to the changing needs in distributed systems. Therefore revised or new cost models, particular communication cost, should be built which add delay and other performance considerations.

3. QoS-aware distributed query processing

As we pointed out previously, to address the changing from both the user's requirement and network's unpredictable performance, we propose the idea of the integration of QoS with distributed query processing. Specifically, we are guided by two main goals when designing the QoS-based cost models, that is, recognition of different user's demands and consideration of dynamic nature of the underlying network. Thus in this section, we first give a global view of the interaction of QoS and query processing. Then the QoS-based cost model is introduced.

3.1. A global view

A logical architecture would be proposed to show the relationships between QoS management and query processing and how to put QoS information into the query processing. Figure 3 shows one possible logical view of query processing with the interference of QoS. This figure is based on the query steps given in section 2.1.

The difference between this graph and Figure 1 lies in the fact that two QoS modules are added. These modules correspond to the two goals of the proposed cost model: one provides different optimization criteria and the other one provides the dynamic information for the underlying network.

First, in order to effectively target the right information to the right user, *user profile* is adopted to store the information of user's preference and requirements. The *user profile*, generated from QoS specification, contains user's QoS requirements and expectations understandable by the underlying systems, thus it helps the optimizer to map the QoS specification into various optimization criteria.

Second, the *QoS profile mapping* module is used for this conversion. Sometimes, user's requirements are mapped onto several criteria, which means multiple cost models can be used. In this case, the optimizer has to analyze these criteria in order to get a compromise so that it can decide which one of the cost models is most suitable in this situation.

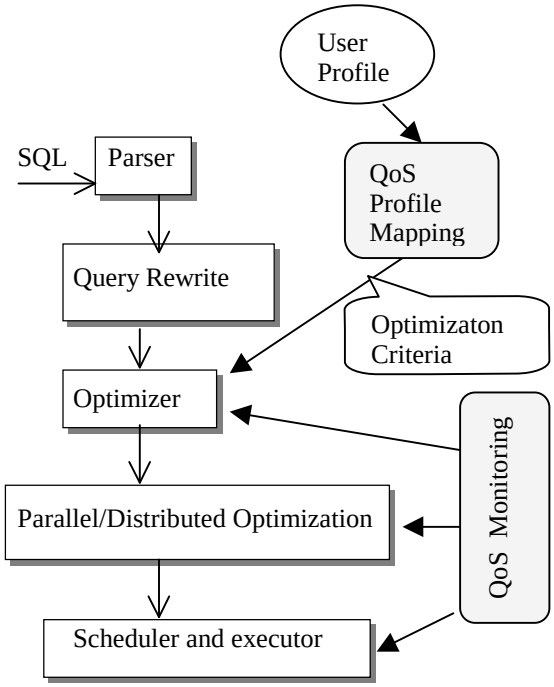

Figure 3. Query Processing with QoS consideration

The last added QoS part shown in the figure is used to provide dynamic information, such as network throughput, delay, jitter etc.. This information helps the optimizer to gain current system state in order to generate a more reasonable and effective parallel plan. The measurement of those dynamic is mainly done by QoS monitoring function, mainly invoked during the optimization phase and execution phase. A more detailed discussion of QoS monitoring can be found in [15]. We listed QoS monitoring in the figure, but in fact, other QoS functions, such as QoS adaptation, mapping and negotiation, also make contributions to this part.

In summary, a proper interaction between QoS management and distributed query processing is important for our QoS-based query processing. Specifically, the specification should be an input to the query processing and help to define the optimization criteria. The QoS monitoring function can help the query processor to gather information on the system state and produce information on the network performance. QoS negotiation, mapping and resource reservation can help the query processing to reduce the search space and give constraints for the execution.

3.2. QoS-based cost models

One of the key problems in the design of a query optimizer for distributed database system is the derivation of efficient and accurate cost models. To make the query processing aware of the QoS, the cost models should also be QoS based. Together with the database statistic information, QoS information is also provided to the cost models so that they can take into account both the user's requirement and capture the current network performance.

Two things should be noted for this QoS-based cost model. First, corresponding to various optimization criteria, several cost models should be built to deal with different criteria. Depending on the user's requirement, the optimizer should pick up the relevant one to do the calculation. Thus the input of the user profile is used to guide the optimizer for the selection of cost models. For our current work [15], we proposed two cost models. The first one is derived from user's point of view, which mainly considers the query response time and user's affordability for that query. This is suitable for the case that user specifies the time and money constraints. But sometimes there is the situation where the user does not specify any optimization constraints: what he cares is just getting the query done (for example to make some update operation). For this case, query optimization should be done from the system performance perspective, that is, how to complete the query with the *lowest* system cost. Accordingly, another cost model is proposed for this situation.

Second, the QoS information needed can be broadly distinguished into two types: **static parameters** and **dynamic parameters.** The static parameters mainly come from the user profile such as media type, coding format, network topology, maximum bandwidth supported, cost that user can afford and so on; the dynamic information, mainly from QoS monitoring and adaptation, includes current server load, available network throughput and current delay.

4. Conclusion

The intention of this paper is to investigate distributed query processing, particularly cost-based query optimization, with the awareness of quality of service. Consequently, QoS-based distributed query processing is discussed. Being aware of various user's requirements is necessary in that they can be further mapped onto different query optimization criteria, which in turn lead to different strategies while constructing a query execution plan. Being aware of current network status is also crucial for the query optimizer to pick up an optimal query plan. Both goals are achieved with the aid of QoS management. First, user requirements can be extracted from user profile, which is generated by QoS specification. Second, other QoS management functions, QoS monitoring in particular, play an important role for collecting parameters of current network performance and server load. Our current work is mainly on the theoretical construction of cost model. Thus, the validation of this cost model through implementation is our immediate future work.

References

[1] ACM Multimedia Systems Journal, Special Issue on QoS Architecture, May 1998.

[2] N.R. Adam and Y. Yesha *Electronic Commerce: An Overview,* in Electronic Commerce, N.R. Adam and Y. Yesha (eds.), Springer-verlag, 1996.

[3] C. Aurrecoechea, A. Campbell., and L. Hauw, *A Survey of QoS Architectures*, ACM Multimedia Systems Journal, No. 6, May 1998, pp. 138-151.

[4] G.v. Bochmann, B. Kerhervé, H. Hafid., P. Dini, and A. Pons. *Architectural Design of Adaptive Multimedia Systems*, In IEEE International Workshop on Multimedia Software Development, Berlin, Germany, 1996.

[5] *IBM DB2 Universal Database Administration Guide*, Version 5.2, IBM Corp, 1998.

[6] G. Graefe, *Query Evaluation Techniques for Large Databases,* ACM Computing Surveys, Vol. 25, No. 2, June 1993.

[7] W. Hasan, D. Florescu and P. Valduriez, *Open Issues in Parallel Query Optimization*, SIGMOD Record, Vol. 25, No. 3, September 1996.

[8] 5th IFIP International Workshop on Quality of Service, Columbia University, New York. URL:http://www.ctr.columbia.edu/iwqos, 1997.

[9] 6th IFIP International Workshop on Quality of Service, Napa Valley, URL: http://www-ece.rice.edu/conf/iwqos98/, May 1998.

[10] Lohman, l.Mohan et al., *Query Processing in R**, Query Process in Database System, Springer-verlag, 1985, pp. 31-47.

[11] K. Nahrstedt, *End-to-End QoS Guarantees in Networked Multimedia Systems*. ACM Computing Surveys, 27 No 4, December 1995, pp. 613-616.

[12] A. Silberschatz, H.F. Korth, S. Sudarshan, *Database System Concepts*, Third Edition, McGraw-Hill, 1997.

[13] A. Vogel, B. Kerhervé, G.V. Bochmann, and J. Gecsei, *Quality of Service Management: a survey*. IEEE Journal of Multimedia Systems, Vol 2 no 2 (Summer 1995) , 10-19.[Nahrstedt, 1995] Nahrstedt, K. (1995). *End-to-End QoS Guarantees in Networked Multimedia Systems*. ACM Computing Surveys, 27 No 4, December 1995, pp. 613-616.

[14] J. Wong, K. Lyons, D. Evans, R. Velthuys, G.V. Bochmann, E. Dubois, N. Georganas, G. Neufeld, T. Özsu, J. Brinskelle, A. Hafid, P. Iglinski, B. Kerhervé, L. Lamont, D. Makaroff, D. Szafron, *Enabling Technology for Distributed Multimedia Applications*, IBM Systems Journal, Fall 1997.

[15] H. Ye, B. Kerhervé and G. V. Bochmann, *An adaptive cost model for distributed query processing*, submitted for publication. May 1999.