Mobile query optimization based on agent-technology for distributed data warehouse and OLAP applications∗

Abdelkader Hameurlain and Franck Morvan
Institut de Recherche en Informatique de Toulouse IRIT, Université Paul Sabatier
118, route de Narbonne, 31062 Toulouse cedex, France
E-mail: {hameur,morvan}@irit.fr

Philipp Tomsich and Robert Bruckner
Institut für Softwaretechnik und Interaktive Systeme, Technische Universität Wien
 Favoritenstrasse 9–11/E188, A-1040 Wien, Austria
E-mail: {tomsich,bruckner}@ifs.tuwien.ac.at

Harald Kosch
Department of Information Technology, Universität Klagenfurt
Universitätsstrasse 65-67, A-9020 Klagenfurt, Austria
E-mail: harald.kosch@itec.uni-klu.ac.at

Peter Brezany
Institut für Softwaretechnik und Parallele Systeme, Universität Wien
Liechtensteinstrasse 22, A-1090 Wien, Austria
E-mail: brezany@par.univie.ac.at

Abstract

With the rapid collection of data in wide variety of fields—ranging from business transactions through medical investigations to scientific research—the demands in data analysis tools are ever growing. Today’s challenges are less related to data storage and information retrieval, but can rather be found in the analysis of data on a global scale in a heterogenous information system: technologies such as On-Line Analytical Processing, Data Mining and Knowledge Discovery in Databases all require the integration of information and efficient query processing. In distributed and heterogenous datasets this can only be achieved by the efficient distribution and scheduling of subtasks in a distributed computing resource. We propose the use of mobile query optimizations based on agent-technology for distributed data warehouse and OLAP applications to adapt the concurrent query execution dynamically to the computing resource it executes on. This is of particular importance in cluster and grid computing.

1 Introduction

Across a wide variety of fields, huge datasets are being collected and accumulated at a dramatic pace. Depending on the application area, these datasets may include data produced by business transactions, medical investigations, scientific simulations, along with data obtained from satellites, telescopes, microscopes, seismic or tomographic techniques, etc. There are major challenges involved in the efficient and reliable storage, fast processing, and extracting descriptive and predictive knowledge from this great mass of data. The volume of these datasets is already measured in terabytes and will soon total petabytes. They are often geographically distributed and their complexity is increasing, meaning that the extraction of meaningful knowledge requires more and more computing resources. The communities of users that need to access and analyze this data are often large and geographically distributed. This combination of large dataset size, geographic distribution of users and resources, and computationally intensive analysis results in complex and stringent performance demands that, until recently, were not satisfied by any existing computational and data management infrastructure.

∗This work is supported in part by the French-Austrian integrated actions program, Amadeus 2002 grant number 03763RE.
Knowledge discovery in databases (KDD) has been defined as the non-trivial process of discovering valid, novel, and potentially useful, and ultimately understandable knowledge from data [7]. KDD has become an important area of research in disciplines such as machine learning, statistics, database systems, and parallel and distributed systems. It is considered as a very promising support for a wide area of modern scientific, engineering, medical, and business applications.

2 Motivation

With the proliferation of e-commerce, federated databases have gained in importance for the collection and analysis of transaction and customer data. For the continuing success of electronic trading, customer relationship management aims to improve the customer’s experience, detect weaknesses in the overall process of fulfilling a contract and help to improve the value chain. This requires the integrated analysis of a multitude of information sources such as customer information, order data, supplier data, product catalogs, delivery tracking data, and statistical material supplementing the user demographics. These various data elements are all stored in different source databases, often with a high update frequency. Due to the size and update frequency, the integration into a data warehouse is infeasible. Rather, queries need to be executed within the entire federated database. This poses new challenges to the implementation of data analysis solutions, as the techniques developed for the parallel loading and prematerialization [9] of the data warehouses become inapplicable.

OLAP is used for interactive analysis of data stored in a data warehouse. Its applications require viewing the data from many perspectives (dimensions). A Data Cube is a data structure, which is used as a multidimensional view of a data warehouse where a critical value, e.g. sales, is organized by several dimensions, for example, sales of automobiles organized by model, color, date of sale and so on. The user can select any subset of those dimensions for processing and perform aggregations along the dimensions. Usually, there are multiple levels of granularity upon them the aggregation may be performed.

Data mining goes far beyond OLAP summarization-style analytical processing by incorporating more advanced techniques for data analysis. Simply stated, data mining refers to extracting or “mining” knowledge from large amount of data. Typically, data mining has two high-level goals of prediction and description. These goals can be achieved using a variety of particular data-mining methods, for example, mining association rules and sequential patterns, classification, prediction, clustering, change and deviation detection, etc.

Parallel and distributed data mining is a young and dynamic scientific field. Several parallel data mining algorithms and systems have already been implemented on parallel computing platforms, e.g. [17].

In many applications, data mining systems must be able to handle and analyze multi-site data repositories. The combination of large dataset sizes, geographic distribution of data, users and resources, and computationally intensive analysis demand for a parallel and distributed data management and analysis infrastructure for parallel and distributed data mining (PDDM). In the literature, several PDDM systems are described (e.g. [6, 5]) that operate on clusters of machines at a local site or clusters of clusters or other machines connected via Internet. Mahinthakumar et al. [14] report about the first data mining algorithm implementation on the Grid. Their parallel clustering algorithm was applied to environmental data taken from the 48 conterminous United States and implemented on various geographically distributed heterogeneous computing systems using the Globus Toolkit. Cannataro and Talia [5] discuss an abstract architecture of a data mining system proposed as an extension of Globus Toolkit services.

On the contrary to data mining, less effort has been devoted to parallel and distributed OLAP [15].

Among many different designs and architectures of data mining systems, On Line Analytical Mining (OLAM) [11], which integrates OLAP with data mining, is a promising direction for data mining on the Grid. Data mining is here performed on OLAP cubes, which are a reduced representation of the mined data sets that is much smaller in volume, yet closely maintains the integrity of the original data. PARSIMONY [9] is the only parallel system implementing a set of selected OLAM techniques.

3 Mobile Query Optimization

The works relative to the optimization of relational queries raise to the 70s, and mainly since the publication of the paper of E. Wong and al [18], and that of P. Selinger and al. [16] (Optimizer of the R-system based on the dynamic programming) who motivated alarge part of the database scientific community to center their efforts on this subject. Hence, because of the importance, and of the complexity of the query optimization problem, the database community spread a considerable effort to develop approaches, methods and techniques of optimization for various relational DBMS, distributed, objects, deductive, and parallel.

The role of an optimizer is to generate, for a given query, an optimal execution plan (or close to the optimal) from the considered search space. The objective of the optimization is to minimize the response time and to maximize the use of the resources of the underlying system while minimizing the optimization cost. The general problem of the query optimization can be expressed as follows: As a query q, a
space of the execution plans \( S \), and a cost function associated to the execution of \( p \in S \), find the execution plan of computing \( q \) such as the \( \text{cost}(q) \) is minimum. Generally, the design of a query optimizer can be decomposed into three elements: A search space defining the syntactic representation of all the relevant aspects of an execution, a search strategy generating an execution plan close to the optimal, and a cost model allowing to annotate operators’ trees in the considered search space.

However, the execution of an execution plan generated by a classic optimizer can present bad performances for three main reasons:

- **Centralization of the decision taken by the optimizer**: Actually, we can notice that all the query optimization methods, static and dynamic, are centralized. They depend on the node or on the site responsible for the optimization. So whatever is the environment (centralized, parallel, or distributed) the methods of dynamic optimization are overseen by a master process that must control all the processes participating in the optimization and in the dynamic re-optimization to make all the necessary adaptations. Although the collection of the informations (current state of the system and the metrics relative to the execution of the relational operators) for the cost evaluator is distributed and organized into a hierarchy, the processes realizing the clones stemming from operations in the course of execution or in wait of liberation of resources are thus completely controlled by the optimizer-allocator of resources presenting a bottleneck.

- **Inaccuracy of the estimations**: There are a lot of reasons for which the estimations computed in the compilation can not be accuracy:
  1. the statistics stored in the system catalog are subject to the errors of estimate or to the obsolescence,
  2. the estimations of the selectivity of the predicates (join and selection operators) but also of the size of the intermediate results, can then be only more inaccuracy, and
  3. the estimations relative to the availability of the resources in terms of the workload of processors, size of the available memory, and the network bandwidth.

- **Inaccessibility or unavailability of the data**: in consideration of the distant access, there is a risk that the necessary data during the execution are not accessible for the following reasons [3]:
  1. the distant site computing the data is overloaded or is not enough powerful,
  2. the network throughput is relatively slow, and
  3. an operand is the result of a complex subquery.

The problems of optimization owed in inaccuracy of the estimations and to the data unavailability were extensively and widely studied in a parallel and distributed environment by considering only the conventional distributed execution models such as the message passing and the distant procedure call. Indeed, the works, most recently, in methods of query optimization, underline the importance not only to balance the workload, but also to correct the sub-optimality of the execution plan [13, 4, 12, 10] foreseen in compilation phase, detectable at the run time. The developed methods became so particularly sophisticated, and constitute more strategies of re-optimization even of re-allocation of resources at the run time. Furthermore, the decisions of resource allocation (i.e. processors, memory, and network bandwidth) are completely controlled by the optimizer.

Recently, the research in distributed systems saw appearing a new programming model for the distributed applications on a large network: mobile agents based programming [8, 19]. A mobile agent is an autonomous and adaptable software entity capable of moving dynamically (code, data and execution state) to reach data or distant resources [1]. The fundamental difference with the classic migration is mainly the initiator of the migration [2]. In fact, in the classic migration, the decision is taken by the operating system, while the mobile agent takes himself the decision to move. This paradigm, that must contribute to improve the performances (e.g. we replace distant interactions by local interactions), is susceptible to substitute itself for the others more conventional such as the message passing, and the distant procedure call. Furthermore, it is important to notice that at present mobile agents based programming platforms supply only migration mechanisms, but they do not offer policy of decision of migration.

For these multiple reasons we study the contribution of the mobile agents in the resolution of the performance problem of mobile distributed queries at two levels:

1. in the decentralization of the control made by the optimizer, and
2. in the dynamic process of optimization (i.e. generation of distributed execution plan) by reacting to the inaccuracies of the estimations (i.e. size of the intermediate relations, and size of the available memory), and to the unavailability of the requested data.

It thus becomes convenient to enable each process realizing an autonomous clone, aware of its environment, and capable of deciding on the moment, and on the place where it migrates to continue its execution on the destination site.

The objective is centered on the query performance improvement which is based on the following elements:
• the transfer of the process workload of a server on the other one,
• the reduction of the data volume between two distant servers (i.e., we replace distant interactions by local interactions),
• the elimination of the bottleneck due to the centralization of the decisions taken by the optimizer.

In this perspective, the handled problems focus, more exactly, on [1]:
1. the definition of a migration policy of the mobile agents stemming from operators of distributed queries,
2. the study of the contribution of the mobile agents in the decentralization of the control done by the optimizer, and
3. to show the validity and the viability of the policy proposed by the development of an experimentation platform.

The use of the mobile agents in the process of dynamic evaluation and optimization of queries presents mainly two advantages:
• on the methodological plan, by the designing and the development of methods and techniques of dynamic optimization freeing itself from the constraint of the heterogeneity (e.g., language, specialized machine, and data formats) and assuring the sharing of workload.
• on the software development plan, the autonomy degree, the capacity of adaptability and the precision of the estimations of costs which possesses the agent will have a considerable impact on the efficiency of the decentralization process of the decisions taken by the optimizer.

4 Conclusion

To study the contribution of the mobile agents in the process of evaluation and dynamic optimization of distributed queries in a grid environment, we design, and develop a system which allows:
• the implementation of a model of evaluation and dynamic optimization of queries, in particular by spreading the relational operators as the join in mobile environment (join, semijoin) [1],
• to the agents, stemming from a query decomposition, to decide between their migration and that of the exploited data,
• the decentralization of the control made by the optimizer so avoiding the bottleneck,
• the specification of statistical operators of system informations and application. The quality of the decision taken by the agent depends strongly on the accuracy of the estimations of costs and on the efficiency of techniques to compute them.

The use of mobile queries in a grid context offers the advantage of enabling dynamic adjustments to the actual query execution plan to be made. This is particularly useful, if the computing resources are not dedicated to the task of data analysis and if parts of the data set may become unavailable intermittently.

References


