OLAP Query Evaluation in a Database Cluster: A Performance Study on Intra-Query Parallelism

Fuat Akal, Klemens Böhm, and Hans-Jörg Schek
Swiss Federal Institute of Technology Zurich, Database Research Group
Institute of Information Systems, ETH Zentrum, 8092 Zürich, Switzerland
{akal,boehm,schek}@inf.ethz.ch

Abstract. While cluster computing is well established, it is not clear how to coordinate clusters consisting of many database components in order to process high workloads. In this paper, we focus on Online Analytical Processing (OLAP) queries, i.e., relatively complex queries whose evaluation tends to be time-consuming, and we report on some observations and preliminary results of our PowerDB project in this context. We investigate how many cluster nodes should be used to evaluate an OLAP query in parallel. Moreover, we provide a classification of OLAP queries, which is used to decide, whether and how a query should be parallelized. We run extensive experiments to evaluate these query classes in quantitative terms. Our results are an important step towards a two-phase query optimizer. In the first phase, the coordination infrastructure decomposes a query into subqueries and ships them to appropriate cluster nodes. In the second phase, each cluster node optimizes and evaluates its sub-query locally.

1 Introduction

Database technology has become a commodity in recent years: It is cheap and is readily available to everybody. Consequently, database clusters likewise are becoming a reality. A database cluster is a network of workstations (PCs), i.e., commodities as well, and each node runs an off-the-shelf database. In the ideal case, a database cluster allows to scale out, i.e., it allows to add more nodes in order to meet a given performance goal, rather than or in addition to modifying or tuning the nodes.

Even though its advantages seem to be obvious, it is not clear at all how data management with a database cluster should look like. Think of a cluster that consists of a large number of nodes, e.g., more than 50. How to make good use of such a cluster to work off a high database workload? How to deal with queries and updates, together with transactional guarantees? In broad terms, the concern of our PowerDB research area [20] is to address these issues and to develop a cluster coordination infrastructure. The infrastructure envisioned completely hides the size of the cluster and the states of its nodes from the application programmer. For the time being, we assume that there is a distinguished node (coordinator) with ‘global knowledge’ as part of the

* Project partially supported by Microsoft Research.

With respect to data distribution, we apply standard distributed physical design schemes [1]. The design scheme determines the query evaluation. The main design alternatives are full replication, facilitating high inter-query parallelism, and horizontal partitioning, improving intra-query parallelism. Recent investigations [2,3] have examined these design alternatives for OLAP queries. However, given a large number of nodes, it might not be a good idea to have just one distributed physical design scheme that spans over the entire cluster. Instead, several schemes may coexist, e.g., the data might be fully replicated on three cluster nodes while the remaining nodes each hold a partition of a fact table and replicas of all the other tables. We refer to such a scheme as a mixed distributed physical design scheme, as opposed to pure distributed schemes. Note that a design scheme may be pure even though the physical layout of tables of the same database is different.

Mixed physical design in turn motivates the investigation of a two-phased query optimization: The coordination middleware chooses the nodes to evaluate a given query and derives subqueries, to be evaluated by these nodes. In the second phase, each node finds a good plan for its subquery and executes it. Two-phased query optimization is appealing for the following reasons: The coordinator load becomes less, compared to a setup where the coordinator is responsible for the entire query optimization process. This approach does not require extensive centralized statistics gathering, only some essential information is needed in the first phase.

Within this general framework, this paper is a first step to address the following specific questions.
1. How well can we parallelize a given query? How to predict the benefit of parallelization with little effort, i.e., by means of a brief inspection of the query?
2. How many cluster nodes (parallelism degree) should be used to evaluate a given query? What is the limit utility when increasing the number of nodes from n to n+1?

Suppose that a pure distributed physical design scheme is given. Question 1 asks if this scheme allows for faster evaluation of the query, as compared to evaluation on a single database. Question 2 in turn assumes that there is a distributed design scheme that allows to continuously adjust the number of nodes to evaluate the query. To address this question, our study assumes a particular design scheme with this characteristic as given. It is based on the TPC-R benchmark [4]. Answers to this question will allow us to come up with mixed physical design schemes.

The contribution of this paper is as follows: It provides a classification of queries where the different classes are associated with specific parallelization characteristics. In particular, ‘parallelization characteristics’ stands for the number of nodes that should be used to evaluate the query in parallel. We provide simple criteria to decide to which class a query belongs. Our experiments yield a characterization of the various classes in quantitative terms. The results will allow us to build a query optimizer at the coordination level for the first phase of optimization.

While our study clearly goes beyond existing examinations, it is also preliminary in various respects. First, the focus of this paper is on queries executed in isolation. We largely ignore that there typically is a stream of queries, and we leave the interdependencies that might arise by the presence of other queries to future work. Further-