OLAP Query Processing in a Database Cluster*

Alexandre A.B. Lima[1], Marta Mattoso[1], and Patrick Valduriez[2]

[1]Computer Science Department, COPPE, Federal University of Rio de Janeiro, Brazil
{assis,marta}@cos.ufrj.br
[2]Atlas Group, INRIA and LINA, University of Nantes, France
Patrick.Valduriez@inria.fr

Abstract. The efficient execution of OLAP queries, which are typically read-only and heavy-weight, is a hard problem which has been traditionally solved using tightly-coupled multiprocessors. Considering a database cluster as a cost-effective alternative, we propose an efficient, yet simple, solution, called fined-grained virtual partitioning to OLAP parallel query processing. We designed this solution for a shared-nothing database cluster architecture that can scale up to very large configurations and support black-box DBMS using non intrusive, simple techniques. To validate our solution, we implemented a Java prototype on a 16 node cluster system and ran experiments with typical queries of the TPC-H benchmark. The results show that our solution yields linear, and sometimes super-linear, speedup. With 16 nodes, it outperforms traditional virtual partitioning by a factor of 6.

1 Introduction

Decision support applications require efficient support for On-Line Analytical Processing (OLAP) on larger and larger databases. OLAP queries are typically read-only and heavy-weight. In the TPC-H benchmark [12], specific to decision support systems, twenty-two database queries are complex, heavy-weight and read-only and only two have updates. Furthermore, OLAP queries have an ad-hoc nature. As users get more experienced about OLAP system features, they demand more efficient ad-hoc query support [5].

The efficient execution of OLAP queries, where "efficiency" means "as fast as possible", is still an open problem. High-performance of database management has been traditionally achieved with parallel database systems [13], implemented on tightly-coupled multiprocessors. Parallel data processing is then obtained by partitioning and replicating the data across the multiprocessor nodes in order to divide processing. Although quite effective, this solution requires the database system to have full control over the data and is expensive in terms of software and hardware. Clusters of PC servers provide a cost-effective alternative to tightly-coupled multiprocessors. Recently, the *database cluster* approach, i.e. clusters with off-the-shelf (black-box) DBMS nodes, has gained much interest for various database applications [1, 4, 9].

In this paper, we propose a solution to efficient OLAP query processing in a database cluster using simple parallel processing techniques. The basic technique we

* Work partially funded by CNPq, CAPES, INRIA and COFECUB.

M. Danelutto, D. Laforenza, M. Vanneschi (Eds.): Euro-Par 2004, LNCS 3149, pp. 355–362, 2004.

employ is virtual partitioning [1] which gives more flexibility than physical (static) data partitioning [7] for parallel query processing. In its simplest form, it consists in fully replicating the database among the cluster nodes. To distribute the workload, predicates are added to queries to force each DBMS to process a different subset, called a *virtual partition*, of data items. Each DBMS processes exactly one sub-query. The problem is these sub-queries can take almost as long as the original query to be executed. Depending on the estimated amount of data to be processed, DBMS optimizers can opt for fully scanning the virtually partitioned table, reducing (or even eliminating) benefits obtained from virtual partitioning. Temporary disk-based structures demanded by sub-queries that deal with huge amounts of data can also limit virtual partitioning performance. In this paper, we propose a major improvement called *fine-grained virtual partitioning* (FGVP) which addresses these problems. As proposed in [1], virtual partitioning assigns each cluster node one sub-query, what can lead to problems aforementioned. Our approach is to work with a larger number of virtual partitions, much greater than the number of nodes. It is an attempt to keep sub-queries as simple as possible, avoid full table scans and expensive temporary disk-based structures. Our experimental results, based on our implementation on a 16-node cluster running PostgreSQL, show that linear, and sometimes super-linear, speedup is obtained for typical OLAP queries. In the worst cases, almost linear speedup is achieved. FGVP outperformed the traditional virtual partitioning for all queries when using more than two nodes. We think FGVP also provides a good basis for dynamic load balancing as it makes it possible to perform sub-query reallocation. This article is organized as follows. Section 2 presents our database cluster architecture. Section 3 describes our fine-grained virtual partitioning technique. Section 4 describes our prototype implementation as well as experimental results. Section 5 concludes.

2 Database Cluster Architecture

A database cluster [1] is a set of PC servers interconnected by a dedicated high-speed network, each one having its own processor(s) and hard disk(s), and running an off-the-shelf DBMS. Similar to multiprocessors, various cluster system architectures are possible: shared-disk, shared-cache and shared-nothing [13]. Shared-disk and shared-cache require a special interconnect that provides a shared space to all nodes with provision for cache coherence using either hardware or software. Shared-nothing (or distributed memory) is the only architecture that does not incur the additional cost of a special interconnect. Furthermore, shared-nothing can scale up to very large configurations. Thus, we strive to exploit a shared-nothing architecture as in PowerDB [11] and Leg@Net [4]. Each cluster node can simply run an inexpensive (non parallel) DBMS. In our case, we use the PostgreSQL [8] DBMS, which is freeware. Furthermore, the DBMS is used as a "black-box" component [4]. In other words, its source code is not available and cannot be changed or extended to be "cluster-aware". Therefore, extra functionality like parallel query processing capabilities must be implemented via middleware.

We use data replication to improve performance. As in [1, 4], we assume full database replication for simplicity: each database is replicated at each node. To maintain copy consistency, we can assume a preventive replication protocol [2] which scales up well in cluster systems. But since database updates are rare in OLAP applications,