Chapter 2.14
Efficient and Robust Node-Partitioned Data Warehouses

Pedro Furtado
Universidade de Coimbra, Portugal

ABSTRACT

Running large data warehouses (DWs) efficiently over low cost platforms places special requirements on the design of system architecture. The idea is to have the DW on a set of low-cost nodes in a nondedicated local area network (LAN). Nodes can run any relational database engine, and the system relies on a partitioning strategy and query processing middle layer. These characteristics are in contrast with typical parallel database systems, which rely on fast dedicated interconnects and hardware, as well as a specialized parallel query optimizer for a specific database engine. This chapter describes the architecture of the node-partitioned data warehouse (NPDW), designed to run on the low cost environment, focusing on the design for partitioning, efficient parallel join and query transformations. Given the low reliability of the target environment, we also show how replicas are incorporated in the design of a robust NPDW strategy with availability guarantees and how the replicas are used for always-on, always efficient behavior in the presence of periodic load and maintenance tasks.

INTRODUCTION

Data warehouses (DWs) are specialized databases storing historical data pertaining to an organization. The objective is to allow business analysis on varied perspectives. They have been applied in many contexts, for instance, insurance companies keeping track of individual events on insurance policies, telecom companies with terabytes of data tracking individual phone calls or individual machine events in production factories, generating gigabytes of detailed data per day. The degree of detail over which the data is stored in the data warehouse can vary, but from the examples given, it is easy to see that data warehouses can become extremely large. As such, multiple performance optimization strategies can be sought after, ranging from specialized indexing, materialized views for faster computation over predicted

query patterns, to parallel architectures and parallel processing. Parallel database systems are implemented on one of the alternative parallel architectures: shared-memory, shared-disk, shared nothing, hierarchical, or NUMA (Valduriez & Ozsu, 1999), which have implications on parallel query processing algorithms, data partitioning, and placement. In practice, parallel environments involve several extra overheads related to data and control exchanges between processing units and also concerning storage, so that all components of the system need to be designed to avoid bottlenecks that would compromise the whole processing efficiency. Some parts of the system have to account for the aggregate flow into/from all units. For instance, in shared-disk systems the storage devices and interconnections should be sufficiently fast to handle the aggregate of all accesses without becoming a significant bottleneck. To handle these requirements, a significant initial and continuous investment is necessary in specialized, fast, and fully-dedicated hardware. An attractive alternative is to use a number of low-cost computer nodes in a shared-nothing environment, possibly in a nondedicated local network. The only requirement is that each node has some database engine and connectivity, while a middle layer provides parallel processing. This system must take into consideration partitioning and processing, as the computer nodes and interconnects are not specially designed to that end. The node-partitioned data warehouse (NPDW) is a generic architecture for partitioning and processing over the data warehouse in such an environment. The objective of this chapter is to discuss and analyze partitioning, processing, and availability issues in the design of the NPDW.

BACKGROUND

Typical data warehouse schemas have some distinctive properties: they are mostly read-only, with periodic loads. This characteristic minimizes consistency issues which are a major concern regarding the parallelization of transactional schemas and workloads; data warehouse schemas usually have multidimensional characteristics (Kimball, Reeves, Ross, & Thornthwaite, 1998), with large central fact relations containing several measurements (e.g., the amount of sales) and a size of up to hundreds or thousands of gigabytes, and dimensions (e.g., shop, client, product, supplier). Each measurement is recorded for each individual combination of dimension values (e.g., sales of a product from a supplier, in one shop and for an individual client). While there are specific analysis-oriented data marts stored and analyzed using some nonrelational multidimensional engine (Kimball, Reeves, Ross, & Thornthwaite, 1998), our focus is on the large central repository warehouses stored in a relational engine; warehouses are used for online analytical processing (OLAP), including reporting and ad-hoc analysis patterns. OLAP involves complex query patterns, with joins involving multiple relations and aggregations. These query patterns can pose difficulties to the performance of shared-nothing partitioned environments, especially when nodes need to exchange massive quantities of data. While very small dimensions can be replicated into every node and kept in memory to speed up joins involving them, much more severe performance problems appear when many large relations need to be joined and processed to produce an answer. We use the schema and query set of the decision support performance benchmark TPC-H (TPC) as an example of such a complex schema and query workload and also as our experimental testbed. Performance and availability are relevant issues in data warehouses in general and pose specific challenges in the NPDW context (standard computer nodes and nonspecialized interconnects).

Some research in recent years has focused on ad-hoc star join processing in data warehouses. Specialized structures such as materialized views (Rousopoulos, 1998) and specialized indexes (Chan & Ioannidis, 1998; O'Neil & Graefe, 1995)