An Overview of DB2 Parallel Edition

Chaitanya Baru, Gilles Fecteau
IBM SWSD, Toronto
Ambuj Goyal, Hui-i Hsiao, Anant Jhingran, Sriram Padmanabhan, Walter Wilson
IBM TJ Watson Research Labs

1 Introduction

In this paper, we describe the architecture and features of DB2 Parallel Edition (PE). DB2 PE belongs to the IBM family of open DB2 client/server database products including DB2/6000, DB2/2, DB2 for HP-UX, and DB2 for the Solaris Operating Environment. DB2 PE employs a *shared nothing* architecture in which the database system consists of a set of independent *logical database nodes*. Each logical node represents a collection of system resources including, processes, main memory, disk storage, and communications, managed by an independent database manager. The logical nodes use message passing to exchange data with each other. Tables are partitioned across nodes using a hash partitioning strategy. The cost-based parallel query optimizer takes table partitioning information into account when generating parallel plans for execution by the runtime system. A DB2 PE system can be configured to contain one or more logical nodes per physical processor. For example, the system can be configured to implement one node per processor in a shared-nothing, MPP system or multiple nodes in a symmetric multiprocessor (SMP) system. This paper provides an overview of the storage model, query optimization, runtime system, utilities, and performance of DB2 Parallel Edition.

2 Storage Model

DB2 PE supports a *partitioned* storage model in which tables in a database are partitioned across a set of database nodes. Indexes are partitioned along with their corresponding tables and stored locally at each node. The system supports *hash partitioning* and *partial declustering* of database tables. Extensions to the CREATE TABLE statement allow users to specify a *partitioning key* and *nodegroup* associated with a table. The partitioning key is a set of columns in the table whose values are used by the hash partitioning strategy to determine the node in which a given row is stored. Nodegroups are named subsets of database nodes. Each nodegroup is associated with a data structure called the *partitioning map (PM)* which consists of 4096 entries each containing a node number. If x represents the partitioning key value of a row in a table, then the node at which this row is stored is given by, PM[H(x)], where H(.) represents the system hash function and PM represents the partitioning map associated with the nodegroup in which the table is created. Data definition language (DDL) extensions are provided to allow users to create nodegroups in a database. For example, users can choose to partition smaller tables across fewer nodes. Nodegroups also provide a convenient way of administering different, independent databases implemented in a single, shared-nothing system.

SIGMOD ' 95, San Jose , CA USA

3 Query Optimization

DB2 PE employs a fully integrated, cost-based parallel query optimizer which directly generates plans for execution in the parallel database system. Some of the important features of the parallel query optimizer include:

- *Transparent parallelism.* Applications that use data manipulation SQL statements do not have to change when moving from DB2/6000 to DB2 PE. Thus, the investment that users and customers have made in existing applications is fully protected and the task of migrating applications is very simple. Applications programs written for DB2/6000 do not even require recompilation. They only need to be *rebound* so that the parallel optimizer can generate the best cost parallel plans for existing SQL queries.
- *Comprehensive usage of data partitioning information.* When choosing a parallel execution strategy, the optimizer makes full use of the data partitioning information for base tables as well as for intermediate tables produced during query execution.
- *Full-fledged cost-based optimization.* The optimization phase of the parallel compiler generates different parallel execution plans and chooses the execution plan with the best cost. The optimizer accounts for the inherent parallelism of different operations and the additional costs introduced by messages while comparing different strategies.
- *Parallelization of all relational operations.* All operations including, index and table scans, aggregation, set operations, joins, inserts, deletes, updates, are able to employ intra-operator parallelism. In generating parallel plans, an extensive range of parallel join methods are considered including *collocated, directed, repartitioned, and broadcast joins.*

4 Runtime System

The runtime support includes the *process model*, *table queue services*, and *control services.* Typically, a query is associated with multiple processes at each database node. Each application is assigned a distinct *coordinator agent* process which coordinates query execution and result collection. A query plan is divided into multiple *subsections* which correspond to different subtrees in the query plan. Each subsection is associated with a *parallel agent* process. Parallel agents execute concurrently, thereby supporting intra-query parallelism. In addition, data is pipelined between consecutive operators in a query plan, thereby supporting pipelined parallelism. The abstraction used for data transfer between consecutive parallel operators in a query plan is called a *table queue.* Many varieties of table queues are supported internally to account for the various types of inter-operator communications required in parallel query plans. The runtime system also includes a fast communications manager which provides support for inter-node message as well as data transfers.

The concurrency control and logging features provided include support for a two-phase commit, presumed abort protocol to coordinate transactions across database nodes, and local logs at each node which allow database recovery either to a point-in-time or to end of logs. A virtual timestamp scheme is used to synchronize transactions across nodes. Each node maintains its own local lock list structures. A global deadlock detection scheme is used to resolve lock contention.

5 Utilities

DB2 PE supports parallel execution of existing (i.e. DB2/6000) utilities such as Load, Import, Reorganize Table, Create Index, Backup, and Restore. All of these utilities can be executed in parallel across database nodes. In addition, several new utilities are provided specifically for the parallel environment. These include utilities that allow users to add database nodes to an existing system configuration; utilities to repair damaged databases at a single node; and a utility called, Redistribute Nodegroup, that provides users effective control on data placement across the nodes of a nodegroup. With the Redistribute Nodegroup utility, users can achieve uniform distribution of data across nodes or control the distribution as desired.

6 Performance

DB2 PE demonstrates excellent scalability for a wide variety of queries as well as for utilities. Some of the performance benchmarking done to date includes measurement of execution times for data loading, index creation, and execution of queries ranging from simple scans and aggregations to complex multi-table joins (e.g. 6 table joins) and queries containing three level nested subqueries. The results of the most recent benchmarking studies will be presented.