Parallel and Distributed Databases

Presentation · February 2020

DOI: 10.13140/RG.2.2.33275.44323

1 author:

Md Hasan Shahriar
Universität Potsdam
8 PUBLICATIONS **13** CITATIONS

SEE PROFILE

Parallel and Distributed Databases

Md Hasan Shahriar

Principles of Data and Knowledge based Systems
University of Potsdam, Am Neuen Palais 10, 14469 Potsdam
shahriar@uni-potsdam.de, www.uni-potsdam.de

Abstract. Parallel and Distributed Databases are efficient ways to process faster and complex query processing. A parallel database can execute multiple tasks in parallel, which allows the database to utilize multiple cores and disks. Data spread across various hosts in a distributed database. Due to concern for system availability and autonomy, parallel and distributed database systems are turning out to be wide-spread.

1 Introduction

Modern applications are more resource-demanding and are usually based on multi-user systems. To provide more efficient query processing, faster machines are needed. According to Moor's law:

> The number of transistors on integrated circuit chips doubles approximately every two years.

The number of transistors in a chip is still increasing, but they are so tiny that more power can not be put in them. Hence, the clock speeds stall. In earlier days, we just had to optimize code for a single thread, now we face a paradigm shift and solve the tasks in Parallel. Based on the computation, the database implementation has to be determined, which is either parallel or distributed.

2 Background

2.1 Parallel Database System

Models of Parallelism Database operations, frequently being time-consuming and involving a lot of data, can generally profit from parallel processing.

Definition 1. *Parallel Database System is a database management system that is implemented on a tightly coupled multiprocessor which has shared memory.*

Components of a Parallel Machine At the heart of all parallel machines is a collection of processors. Each processor has its own local cache and local memory. Along with these processors are many disks, perhaps one or more per processor, or in some architectures a large collection of disks accessible to all processors directly. Additionally, parallel computers all have some communications facilities for passing information among processors.

2.2 Parallel Database Architectures

The goal of parallel architectures is to improve performance by connecting multiple CPU and disks in parallel and improve the availability of data, as it can be copied to multiple locations. Another aspect is to improve reliability, completeness, and accuracy of data, at the same time providing distributed access.

Classification of parallel architectures Parallel database architectures are defined into three groups. The most tightly coupled architecture shares the main memory. A less tightly coupled architecture shares only disks. The most frequently used architecture for databases does not share disk, the processors are interconnected and share data through message passing.

Shared-Memory Machines In this architecture, (see Fig. 1) each processor has access to all the memory of all processors. There is a single physical address space for the entire machine, rather than one address space for each processor. The large machines are of the NUMA (non-uniform memory access) type, which takes more time to access data in a memory that "belongs" to some other processor than it does to access its "own" memory or local cluster. This occurs as a critical issue, whether the data a processor needs, should be in its own cache or not.

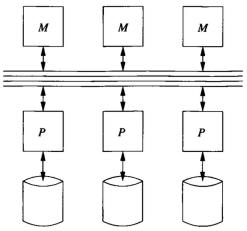

Fig. 1. A Shared-Memory Machine.

Shared-Disk Machines In this architecture, (see Fig. 2) every processor has its own memory, which is not directly accessible from other processors. However, the disks are accessible from any processors through the communication network. Disk controllers manage the potentially competing requests from different processors. This architecture is two types, depending on the units of transfer between the disks and processors. Disk farms called network-attached storage (NAS) store and transfer files. The alternative, storage area networks (SAN) transfer disk blocks to and from the processors.

Shared-Nothing Machines Here, all processors have their own memory and their own disk or disks, (see Fig. 3). All communication is via the network, from

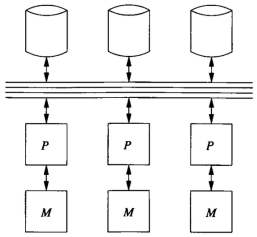

Fig. 2. A Shared-Disk Machine.

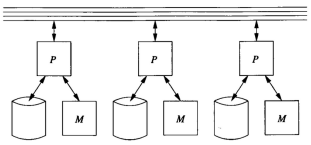

Fig. 3. A Shared-Nothing Machine.

processor to processor. The shared-nothing architecture is the most commonly used architecture for database systems. Shared-nothing machines are relatively inexpensive to build.

2.3 Map-Reduce Parallelism Framework

Map-reduce is a high-level programming system that permits numerous significant database procedures to be effectively composed. The user writes code for two functions, map and reduce. A master controller splits the input data into chunks and allocates several processors to execute the map function on every chunk. Other processors, or the same ones, are then assigned to implement the reduce function on pieces of the output from the map function.

2.4 I/O Parallelism

Data can be partitioned across multiple disks for parallel I/O. There are several partitioning techniques, such as Round-robin, Hash partitioning, and Range partitioning. The round-robin method sends the ith tuple inserted in the (i mod n)-th disk. Hash partitioning chooses a partitioning attribute and a hash function h with range 0 ..... n-1. The hash function h applies the partitioning attribute and sends tuple to disk i based on the return value. Range partitioning also chooses a partitioning attribute and a partitioning vector [v0, v1, ..., vn-2] and sends tuple to ith disk based on the vector.

2.5 Distributed Database System

Definition 2. *A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (Distributed DBMS) permits the management of the distributed database and makes the distribution transparent to the users.*

2.6 Distributed Database Environments

Distributed databases grouped into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in Fig. 4.

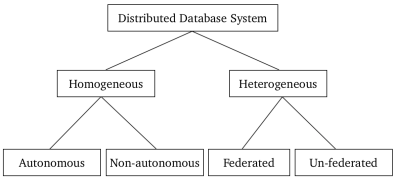

Fig. 4. Distributed Database Environments

Homogeneous distributed database In a homogeneous distributed database, every one of the sites uses identical database management systems and operating systems. Each site is aware of all other sites and cooperates to process user requests. The database is accessed through a single interface as if it is a single database. It can be of two types. If the database is independent and functions on its own, it is called autonomous. It is integrated by a controlling application and uses message passing. In a non-autonomous database, data is distributed over many nodes and a master DBMS coordinates data updates.

Heterogeneous distributed database In a heterogeneous distributed database, all the sites use different schemes, operating systems, and data models. It can be composed of various DBMS, i.e. relational, network, hierarchical, object-oriented. Hence, query processing is complex due to dissimilar schemes. Furthermore, Transaction processing is also complex due to dissimilar software. A site may not be aware of others, thus there are limited help processing user requests. They are of two types. Federated is independent in nature and integrated together; they function as a single database system. Un-federated employs a central coordinating module through which the databases are accessed.

2.7 Distributed Database Architecture

DDBMS architectures are developed on three parameters, Distribution, Autonomy and Heterogeneity.

Distribution states the physical distribution of data across different sites. Autonomy indicates the distribution of control and degree to which each constituent DBMS can operate independently. Heterogeneity refers to the uniformity or dissimilarity of the data models and system components. Some of the common architectural models are:

- Client-Server Architecture: It is a two-tier architecture. The server primarily encompasses data management, query processing, optimization, transaction management etc. The client mainly provides a user interface.
- Peer-to-Peer Architecture: Each peer acts both as a client and a server. The peers share their resources with other peers and coordinate their activities.
- Multi-DBMS Architecture: It is an integrated database system formed by a collection of two or more autonomous database systems.

2.8 Distributed Query Processing

Distributed Query Processing is the procedure of answering queries in a distributed environment where data is managed at multiple nodes. Query processing involves the transformation of a high-level query into a query execution plan which consists of lower-level query operators in some variation of relational algebra and it's execution. The goal is to produce a plan which is equivalent to the original query returning the same result but more efficient.

2.9 Distributed Commit

Distributed Commit is a complex protocol to determine whether to commit a distributed transaction uniformly. A transaction manager conveys the decision to all nodes where the transaction is being executed. When transaction is complete at a site, it reaches a "partially committed" state and waits for other sites to reach same. After receiving a message that all sites are ready, it starts to commit. Different distributed commit protocols are One-phase commit, Two-phase commit and Three-phase commit.

3 Related Work

Non-Uniform Memory Architecture (NUMA), Cluster (SAN/NAS) are few examples of Parallel database system. There are many distributed database implementations, such as Cassandra, Apache Hbase, MongoDB, Neo4j, CouchDB, Terrastore, FlockDB, Redis, Riak, OrientDB etc.

4 Conclusion

As distributed and multi-process applications are becoming a reality, parallel and distributed database design are growing as an innovative and relevant area. They require their own theory, definitions, and methodologies. This report has presented approaches to these database designs and approaches have been exemplified and compared.

References

1. H. Garcia-Molina, J. Ullman, and J.Widom.Database Systems: The Complete Book. Prentice Hall Press, second edition, 2008.
2. A. Kemper and A. Eickler.Datenbanksysteme - Eine Einführung, 10.Auflage. Oldenbourg, 2015. https://www3.in.tum.de/teaching/bookDBMSeinf.
3. Distributed DBMS - Database Environments, tutorialspoint.com
4. Comparative Analysis: Parallel and Distributed Database, Er. Ramandeep Kaur Brar,Er. Chinu,Er. Gurpreet Kaur, IJRASET, Volume 5 Issue V, May 2017. https://www.ijraset.com/fileserve.php?FID=7497
5. Database System Concepts, by Avi Silberschatz, Henry F. Korth, and S. Sudarshan.
6. Top 10 Open Source Big Data Databases https://bitnine.net/blog-useful-information/top-10-open-source-big-data-databases
7. Nearest Neighbor Search, Apostolos N. Papadopoulos, Yannis Manolopoulos, https://link.springer.com/chapter/10.1007
8. Distributed Database Design Methodologies, Stefano Ceri, Barbara Pernici, Gio Wiederhold, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1458038