Parallel and Distributed Transaction Processing

We studied transaction processing in centralized databases earlier, covering concurrency control in Chapter 18 and recovery in Chapter 19. In this chapter, we study how to carry out transaction processing in parallel and distributed databases. In addition to supporting concurrency control and recovery, transaction processing in parallel and distributed databases must also deal with issues due to replication of data, and of failures of some nodes.

Both parallel and distributed databases have multiple nodes, which can fail independently. The main difference between parallel and distributed databases from the view point of transaction processing is that the latency of remote access is much higher, and bandwidth lower, in a distributed database than in a parallel database where all nodes are in a single data center. Failures such as network partitioning and message delays are much less likely within a data center than across geographically distributed sites, but nevertheless they can occur; transaction processing must be done correctly even if they do occur.

Thus, most techniques for transaction processing are common to both parallel and distributed databases. In the few cases where there is a difference, we explicitly point out the difference. And as a result, in this chapter, whenever we say that a technique is applicable to distributed databases, it should be interpreted to mean that it is applicable to distributed databases as well as to parallel databases, unless we explicitly say otherwise.

In Section 23.1, we outline a model for transaction processing in a distributed database. In Section 23.2, we describe how to implement atomic transactions in a distributed database by using special commit protocols.

In Section 23.3 we describe how to extend traditional concurrency control techniques to distributed databases. Section 23.4 describes concurrency control techniques for the case where data items are replicated, while Section 23.5 describes further extensions including how multiversion concurrency control techniques can be extended to deal with distributed databases, and concurrency control can be implemented with
heterogeneous distributed databases. Replication with weak degrees of consistency is discussed in Section 23.6.

Most techniques for dealing with distributed data require the use of coordinators to ensure consistent and efficient transaction processing. In Section 23.7 we discuss how coordinators can be chosen in a distributed fashion, robust to failures. Finally, Section 23.8 describes the distributed consensus problem, outlines solutions for the problem, and then discusses how these solutions can be used to implement fault-tolerant services by means of replication of a log.

Bibliographical Notes

Textbook discussions of distributed databases are offered by [Oszu and Valduriez (2010)]. [Breitbart et al. (1999b)] presents an overview of distributed databases.

The implementation of the transaction concept in a distributed database is presented by [Gray (1981)] and [Traiger et al. (1982)]. The 2PC protocol was developed by [Lampson and Sturgis (1976)]. The three-phase commit protocol is from [Skeen (1981)]. [Mohan and Lindsay (1983)] discusses two modified versions of 2PC, called presume commit and presume abort, that reduce the overhead of 2PC by defining default assumptions regarding the fate of transactions.

Distributed clock synchronization is discussed in [Lamport (1978)]. Distributed concurrency control is covered by [Bernstein and Goodman (1981)].

Distributed optimistic concurrency control with reduced rollback [Agrawal et al. (1987)]

Distributed snapshot isolation is described in [Binnig et al. (2014)] and [Schenkel et al. (1999)]. A distributed snapshot isolation technique with serializability checks based on timestamps, along with an implementation of the techniques on HBase, is discussed in [Padhye and Tripathi (2015)].
Experience in building a database using Amazon’s S3 cloud-based storage is described in [Brantner et al. (2008)]. An approach to making transactions work correctly in cloud systems is discussed in [Lomet et al. (2009)].

Transaction processing in federated database systems is discussed in [Mehrotra et al. (2001)]. The ticket scheme is presented in [Georgakopoulos et al. (1994)]. 2LSR is introduced in [Mehrotra et al. (1991)].

Techniques for combining concurrency control with commit protocols based on consensus, to reduce overheads, are described by [Kraska et al. (2013)], [I. Zhang and Ports (2015)] and [Mu et al. (2016)].

Bibliography


Further Reading


Credits

The photo of the sailboats in the beginning of the chapter is due to ©Pavel Nesvadba/Shutterstock.