Part II Conclusion

Performance and scalability are important properties of any database system. The storage engine and node-local read-write path can have a larger impact on *performance* of the system: how quickly it can process requests locally. At the same time, a subsystem responsible for communication in the cluster often has a larger impact on the *scalability* of the database system: maximum cluster size and capacity. However, the storage engine can only be used for a limited number of use cases if it's not scalable and its performance degrades as the dataset grows. At the same time, putting a slow atomic commit protocol on top of the fastest storage engine will not yield good results.

Distributed, cluster-wide, and node-local processes are interconnected, and have to be considered holistically. When designing a database system, you have to consider how different subsystems fit and work together.

Part II began with a discussion of how distributed systems are different from single-node applications, and which difficulties are to be expected in such environments.

We discussed the basic distributed system building blocks, different consistency models, and several important classes of distributed algorithms, some of which can be used to implement these consistency models:

Failure detection
: Identify remote process failures accurately and efficiently.

Leader election
: Quickly and reliably choose a single process to temporarily serve as a coordinator.

Dissemination
: Reliably distribute information using peer-to-peer communication.

Anti-entropy
: Identify and repair state divergence between the nodes.

Distributed transactions
: Execute series of operations against multiple partitions atomically.

Consensus
: Reach an agreement between remote participants while tolerating process failures.

These algorithms are used in many database systems, message queues, schedulers, and other important infrastructure software. Using the knowledge from this book, you'll be able to better understand how they work, which, in turn, will help to make better decisions about which software to use, and identify potential problems.

Further Reading

At the end of each chapter, you can find resources related to the material presented in the chapter. Here, you'll find books you can address for further study, covering both concepts mentioned in this book and other concepts. This list is not meant to be complete, but these sources contain a lot of important and useful information relevant for database systems enthusiasts, some of which is not covered in this book:

Database systems
: Bernstein, Philip A., Vassco Hadzilacos, and Nathan Goodman. 1987. *Concurrency Control and Recovery in Database Systems*. Boston: Addison-Wesley Longman.

  Korth, Henry F. and Abraham Silberschatz. 1986. *Database System Concepts*. New York: McGraw-Hill.

  Gray, Jim and Andreas Reuter. 1992. *Transaction Processing: Concepts and Techniques* (1st Ed.). San Francisco: Morgan Kaufmann.

  Stonebraker, Michael and Joseph M. Hellerstein (Eds.). 1998. *Readings in Database Systems* (3rd Ed.). San Francisco: Morgan Kaufmann.

  Weikum, Gerhard and Gottfried Vossen. 2001. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. San Francisco: Morgan Kaufmann.

  Ramakrishnan, Raghu and Johannes Gehrke. 2002. *Database Management Systems* (3 Ed.). New York: McGraw-Hill.

  Garcia-Molina, Hector, Jeffrey D. Ullman, and Jennifer Widom. 2008. *Database Systems: The Complete Book* (2 Ed.). Upper Saddle River, NJ: Prentice Hall.

  Bernstein, Philip A. and Eric Newcomer. 2009. *Principles of Transaction Processing* (2nd Ed.). San Francisco: Morgan Kaufmann.

Elmasri, Ramez and Shamkant Navathe. 2010. *Fundamentals of Database Systems* (6th Ed.). Boston: Addison-Wesley.

Lake, Peter and Paul Crowther. 2013. *Concise Guide to Databases: A Practical Introduction*. New York: Springer.

Härder, Theo, Caetano Sauer, Goetz Graefe, and Wey Guy. 2015. *Instant recovery with write-ahead logging*. Datenbank-Spektrum.

Distributed systems

Lynch, Nancy A. *Distributed Algorithms*. 1996. San Francisco: Morgan Kaufmann.

Attiya, Hagit, and Jennifer Welch. 2004. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. Hoboken, NJ: John Wiley & Sons.

Birman, Kenneth P. 2005. *Reliable Distributed Systems: Technologies, Web Services, and Applications*. Berlin: Springer-Verlag.

Cachin, Christian, Rachid Guerraoui, and Lus Rodrigues. 2011. *Introduction to Reliable and Secure Distributed Programming* (2nd Ed.). New York: Springer.

Fokkink, Wan. 2013. *Distributed Algorithms: An Intuitive Approach*. The MIT Press.

Ghosh, Sukumar. *Distributed Systems: An Algorithmic Approach* (2nd Ed.). Chapman & Hall/CRC.

Tanenbaum Andrew S. and Maarten van Steen. 2017. *Distributed Systems: Principles and Paradigms* (3rd Ed.). Boston: Pearson.

Operating databases

Beyer, Betsy, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016 *Site Reliability Engineering: How Google Runs Production Systems* (1st Ed.). Boston: O'Reilly Media.

Blank-Edelman, David N. 2018. *Seeking SRE*. Boston: O'Reilly Media.

Campbell, Laine and Charity Majors. 2017. *Database Reliability Engineering: Designing and Operating Resilient Database Systems* (1st Ed.). Boston: O'Reilly Media. +Sridharan, Cindy. 2018. *Distributed Systems Observability: A Guide to Building Robust Systems*. Boston: O'Reilly Media.