Chapter 14

Introduction to Distributed Systems

Why Distributed Systems?

Today's computing and information systems are inherently *distributed.* Many companies are operating on a global scale, with thousands or even millions of machines on all the continents. Data is stored in various data centers, computing tasks are performed on multiple machines. At the other end of the spectrum, also your mobile phone is a distributed system. Not only does it probably share some of your data with the cloud, the phone itself contains multiple processing and storage units. Your phone is a complicated distributed architecture.

Moreover, computers have come a long way. In the early 1970s, microchips featured a clock rate of roughly 1 MHz. Ten years later, in the early 1980s, you could get a computer with a clock rate of roughly 10 MHz. In the early 1990s, clock speed was around 100 MHz. In the early 2000s, the first 1 GHz processor was shipped to customers. In 2002 one could already buy a processor with a clock rate between 3 and 4 GHz. If you buy a new computer today, chances are that the clock rate is still between 3 and 4 GHz, since clock rates basically stopped increasing. Clock speed can apparently not go beyond a few GHz without running into physical issues such as overheating. Since 2003, most advances in computing architectures are due to the multi-core revolution: Computers are becoming more parallel, concurrent, and distributed.

Finally, data is more reliably stored on multiple geographically distributed machines. This way, the data can withstand regional disasters such as floods, fire, meteorites, or electromagnetic pulses, for instance triggered by solar superstorms. In addition, geographically distributed data is also safer from human attacks. Recently we learned that computer hardware is pretty insecure. Scary attacks exist, with frightening names such as spectre, meltdown, rowhammer, memory deduplication. There are even attacks on hardware that is considered secure! If we store our data on multiple machines, it may be safe assuming hackers cannot attack all machines concurrently. Moreover, data and software replication also help availability as data remains accessible even if some machines need to be taken offline for maintenance.

In summary, today almost all computer systems are distributed, for different reasons:

- Geography: Large organizations and companies are inherently geographically distributed, and a computer system needs to deal with this issue anyway.
- Parallelism: To speed up computation, we employ multicore processors or computing clusters.
- Reliability: Data is replicated on different machines to prevent data loss.
- Availability: Data is replicated on different machines to allow for access at any time, without bottlenecks, minimizing latency.

Even though distributed systems have many benefits, such as increased storage or computational power, they also introduce challenging *coordination* problems. Some say that going from one computer to two is a bit like having a second child. When you have one child and all cookies are gone from the cookie jar, you know who did it!

Coordination problems are so prevalent, they come with various flavors and names. Probably there is a term for every letter of the alphabet: agreement, blockchain, consensus, consistency, distributed ledger, event sourcing, fault-tolerance, etc.

Coordination problems will happen quite often in a distributed system. Even though every single *node* (node is a general term for anything that computes, e.g., a computer, a multiprocessor core, a network switch, etc.) of a distributed system will only fail once every few years, with millions of nodes, you can expect a failure every minute. On the bright side, one may hope that a distributed system may have enough redundancy to tolerate node failures and continue to work correctly.

Distributed Systems Overview

We introduce some basic techniques for building distributed systems, with a focus on fault-tolerance. We will study different protocols and algorithms that allow for fault-tolerant operation, and we will discuss practical systems that implement these techniques.

We will see different models (and even more combinations of models) that can be studied. We will not discuss them in detail now, but simply define them when we use them. Towards the end of the course a general picture should emerge, hopefully!

The focus is on protocols and systems that matter in practice. In other words, in this course, we do not discuss concepts because they are fun, but because they are practically relevant.

Nevertheless, have fun!

Chapter Notes

Many good textbooks have been written on the subject, e.g. [AW04, CGR11, CDKB11, Lyn96, Mul93, Ray13, TS01]. James Aspnes has written an excellent freely available script on distributed systems [Asp14]. Similarly to our course, these texts focus on large-scale distributed systems, and hence there is some overlap with our course. There are also some excellent textbooks focusing on small-scale multicore systems, e.g. [HS08].

Some chapters of this course have been developed in collaboration with (former) PhD students, see chapter notes for details. Many colleagues and students have helped to improve exercises and script. Thanks go to Georg Bachmeier, Pascal Bissig, Philipp Brandes, Christian Decker, Manuel Eichelberger, Klaus-Tycho Förster, Arthur Gervais, Pankaj Khanchandani, Barbara Keller, Rik Melis, Darya Melnyk, Tejaswi Nadahalli, Peter Robinson, Jakub Sliwinski, Selma Steinhoff, Julian Steger, David Stolz, and Saravanan Vijayakumaran. Jinchuan Chen, Qiang Lin, Yunzhi Xue, and Qing Zhu translated this text into Simplified Chinese, and along the way found improvements to the English version as well. Thanks!

Bibliography

[Asp14] James Aspnes. Notes on Theory of Distributed Systems, 2014.

[AW04] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition).* John Wiley Interscience, March 2004.

[CDKB11] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design.* Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[CGR11] Christian Cachin, Rachid Guerraoui, and Lus Rodrigues. *Introduction to Reliable and Secure Distributed Programming.* Springer Publishing Company, Incorporated, 2nd edition, 2011.

[HS08] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[Mul93] Sape Mullender, editor. *Distributed Systems (2nd Ed.).* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993.

[Ray13] Michel Raynal. *Distributed Algorithms for Message-Passing Systems.* Springer Publishing Company, Incorporated, 2013.

[TS01] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.