Chapter 6

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. Just a few years later, in the mid 2000s, one could already buy processors with clock rates 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 after about 2004. Clock speed can apparently not go beyond a few GHz without running into physical issues such as overheating. Since 2004, computing architectures are mostly developing by 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: spectre, meltdown, rowhammer, memory deduplication, and even attacks on secure hardware like SGX. If we store our data on multiple machines, it may be safe assuming hackers cannot attack all machines concurrently. Data and software replication also helps availability, as computer systems do not need to be shut down 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: consistency, agreement, consensus, blockchain, ledger, event sourcing, etc.

Coordination problems will happen quite often in a distributed system. Even though every single node (computer, core, 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 with multiple nodes may tolerate some failures and continue to work correctly.

**Distributed Systems Overview**

We introduce some basic techniques to 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 Pascal Bissig, Philipp Brandes, Christian Decker, Manuel Eichelberger, Klaus-Tycho Förster, Arthur Gervais, Barbara Keller, Rik Melis, Darya Melnyk, Peter Robinson, 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


