Unreliable Clocks

Background

From a theoretical perspective, using clocks in a distributed system seems like a great idea. Even though separate nodes may not be able to communicate with each other, or perhaps have a conflict about the ordering of certain writes that came to them, using a timestamp should be an easy way to resolve every conflict!

However, this is not the case, as we cannot rely on them.

Clocks on Computers

On most computers, clocks are typically set by occasionally synchronizing with an NTP server (Network Time Protocol). Once this happens, computers will use their internal quartz clock to measure elapsed time until the next synchronization.

Clocks on Computers

On most computers, clocks are typically set by occasionally synchronizing with an NTP server (Network Time Protocol). Once this happens, computers will use their internal quartz clock to measure elapsed time until the next synchronization.

Issues:

- Synchronizing with NTP requires a network call which can take any amount of time
- Quartz Clock Drift - on average upto 7 seconds per day
- Cannot set clock of device we do not own (e.g. a client device like a phone)

Measuring Elapsed Time

Do not use your computer’s time of day clock for this!

- May jump around if synchronized with NTP
- Often ignores leap seconds

Instead, use a monotonic clock (specific value of the clock means nothing, basically just exists to measure time deltas)

Ordering Events

Do not use time of day clocks for this, as they could be out of sync! This may result in dropping data, or even seeming like a piece of data was sent backwards in time (if the sender clock is ahead of the recipient one).

Instead, should be using logical timestamps, only purpose is to measure a relative ordering of events (we will discuss these soon).

Summary

In distributed systems, clocks can really only generate a confidence interval that it is currently some time. While synchronising with NTP frequently enough can decrease the size of the interval, clocks are ultimately unreliable due to the network response time of NTP, as well as quartz drift on a local machine.

Yet still many database solutions use it for either ordering events or conflict resolution. Why? Because it’s simple. Is this good? Probably not, and they admit that.