6.033 Lecture 15
Fault Tolerance II

Americo De Filippo

June 27, 2023

References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Section: 9.1**

One of the most important concept in this area of computer science is **Atomiciy**.

Failures One way of dealing with it is via Replicate a component and then vote. But the problem with it is that often is really expencive and do not work always.

Recoverability Allow to fail, but in order that you can detect the failure and you start some kind of recover procedure. Let's say a child want to learn to walk one way could have been, make the child never fall, another way instead could (and is) make the child fall but giving him a procedure to stand up again.

1 The transfer problem

xfer(from, to, $ amount), the approch is "all or nothing" (no middle states). So during the recover procedure the system has to have a way of backingout in order to abort the action made before the failure.

1.1 Concurrency

Image that you have 2 transfer running at the same time, on the same data item. As we now we have to be careful with the order of execution of the 2 threads. So you would like to see some kind of ordered actions, this is done via **Isolation** of actions [1]. Of course we don't want to run one after the other (we'd lose concurrency).

[1] "Do all before or all after"

2 Atomcity

The basic idea is to hide the fact that some sequence of action are composed. Another interpretation is that atomic means: REC + ISO.

2.1 Recoverability

General plan is to design module to be fail-fast. Once the failure is detected you run some kind of repair procedure and it restarts (allow the module to continue working).

The Golden Rule Never modify the only copy.

1. Recoverable sector: coming out with a scheme that will allow us to do read/write on a sector, that will allow us to resuscitate our system.
2. Version history: this will recoverable sector as bootstrap, and are going to build up on for being able to have a general solution.
3. Logs

2.2 Model

The first assumption that we make is that there is not concurrency, we'll also assume that there is no hardware errors (check sum for every sector) and there are only software errors.

2.3 careful_put, _get

This two procedure will enable us to have a system that is able to do not worry about hardware failure, this is achieved by controlling the sectors in which is allowed to read and to write. We won't allow anyone to access sectors which are ambiguos or half written.

2.4 Commit point

Before the commint point every one that want to read the data will get the older version of it, instead after the version is updated.