6.033 Lecture 16
Fault Tolerance III

Americo De Filippo

June 28, 2023

References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Section: 9.2 / 9.3**

In this lecture we are focus on how to implement Recoverabilty (motivation for the topic in the last lecture).

How COMMIT() is implemented and used?

1 Version histories

The idea is the same of last time: Never modify the only copy. It generalise this idea in: never modify anything, create another version of it. You are gonna have a Cell storage and a Journal storage. When you assign a value to a cell storage you will overwrite the previus value. For this reason we have a Journal storage that is gonna keep track of the value and id of the one which has made the modification. The jornal will contain only the committed action, so in case of failure you will go onto the last commited action.

2 Logging

The way to think of a log is like a version history, but is data structure that will contain only the read and write operation sequentially. So we are gonna use Cell storage for read/write, then the log is going to be stored on a non-volatile medium and is sequentially.

2.1 The plan for recover using logging

When you fail the system run a recovery procedure from the log: for action that are uncommitted we have to look at the updating of the cell store (the log will help us). For committed actions we are going to install them (redo). When ABORT() is called we are going to undo current action.

2.2 How to use the log

Let's say we are going to update a variable, the log is going to take some kind of diary of all the update done to the system (the variable in it). A log is like an Append-only data structure: he is going to have two kind of record:

- **UPDATE**:
  id: 172
  undo: x = old [1]
  redo: x = new [2]
- **OUTCOME**:
  id: 172
  status: COMMITED or ABORTED or PENDING

2.3 When to write to log?

Disk-bound DB Is a disk where application are writing to a database where cell storage are implemented. Cell storage are actually into the disk. Futhermore, you have to maintain a log on a different disk. So when you write something to disk you have to update the log record. Now the question is when writing into the log? Let's say you write into the variable, and then you crash before writing into the log, you are in truble.

2.3.1 2 principle: WAL protocol and logging COMMIT()

The first principle is: Write ahead logging. Append to the log before writing to the cell storage. Suppose now you set x to some value and then you crash, you are garateed that the log has been written. The second principle is to log the COMMIT record before returining from the COMMIT().

2.3.2 How to implement ABORT()?

If the system calls ABORT(), it will look into the steps into the action cells and go into the undone field of those, and assign it (going back to the previus version of the every involved cells storages).

2.4 Recovery procedure

1. **Scan log backwards**
2. **Winners** COMMITED or ABORTED
   Losers everything else
3. **Redo** COMM. Winners
   Undo Losers

[1] in case of changing the value the old value would become the undo, for backing down
[2] this is where the new value is stored, in case has to be (re)done