6.033 Lecture 18
Fault Tolerance V

Americo De Filippo

July 5, 2023

References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Section: 9.6**

In the previus lecture we have talk about locking in order to achieve isolation. We have ackledge a tequinque called 2 phase locking, in order to avoid cycle in the action graph, hence obtain serializability. The golder rule of 2PL is not release until you haven't acquire everything.

1 Deadlocks

A deadlock is a common problem when you use locks. The idea is that there would be some kind of stuck state where 2 or more actions have to wait until some other release and at the same time the other has to wait for the same release another resource [1]. A solution of this can be to use a timer on every actions, when the timer goes up the action will abort and therefore release the resourse.

1.1 Waits-for-graph

Imagine you have a database system, anytime you want to acquire or release a lock you send some kind of request to a lock manager. From this you would create a graph and search for a cycle, when you find it just kill one of them. You would look into the data structure every time every one acquire an resouce or with some time spacing between the checks.

2 Log & Locks

When you abort you have to undo, therefore have the locks for the resource. But suppose you crash, you have to redo or undo (which need access to the locks

[1] look for cycle in Holt's graph

of those resource). The question to think about is how to trace the locks into the log, but you don't have to store the locks, 'cause in order to write a data into the log you implicity had the lock on that data.

3 Applications

3.1 Transactions

Consistecy, Durability. Constistency means that if some action commits there has to be some invariants that must hold. Durability means that the changes made by an atomic action has to last for some time.

3.1.1 Constitency in centrilized systems

In this system there will be rules, like integrity rules that will allow some action of commit only if some kinds of system integrity tests are passed. This is done in order to protect the system.

3.1.2 Constistency in distribuited data

One example could be the DNS that mantain some expiration time in order to keep the cache consistent.

Strong cons The natural definition is that every read return the result of the last write onto the data. This is really hard to garantee.

Eventual cons What this said is that there could be period of time where the system works in the background the make sure that all the copies look the same as the last write onto the data.

Web caches The semantics here is that you actually want to return some good data to the client. This means that the cache does not return the data right ahead but if-modified-since the last write on that data in the cache, it would ask the web server to send the newer version of that data.

3.1.3 Cache constistency

There are 2 idea in order to achieve this. This is supposed on system which are build on shared bus.

Write-thru cache The basic idea is that when something is updated into the cache, the same variable is updated also into the main memory

Snoopy cache You would just look into the shared bus and notice if some entry that you have stored in your own cache is going to change, once you noticed that you would just go ahead into the main memory and update your entry (or just invalidate that data entry in your cache).