Cost of disk failure is high!

Unlike other components, disk is the one that if it fails your data is gone -- can always replace the CPU or the cable, or restart the file system. you can replace the disk, but then your system doesn't work.

Disk Failures
whole disk failures:
electrical fault, broken wire, etc.
sector failures:
can read or write from the wrong place (e.g., if drive is bumped)
can fail to read or write because head is too high, or coating too thin
disk head can contact the drive surface and scratch it

Coping with sector errors:

1) Silently return the wrong answer
2) Detect failure -- fail fast
each sector has a header with a checksum
every disk read fetches a sector and its header
firmware compares checksum to data
returns error to driver if checksum doesn't match
3) Correct / mask
Re-read if firmware signals error (if an alignment problem)
These are general approaches -- e.g., could have checkssums at the application level (in files), or a backup to recover / correct from a bad file.

Checkssums help a lot; very very small fractions of uncaught errors; 100s in trillions yet lots of disks * very many read/write operations/sec --> non-negligible error rate

write (sector, data):
write(disk1, sector, data)
write(disk2, sector, data)

read (sector, data)
data = read (disk1, sector)
if error
data = read (disk2, sector)
if error
    return error
return data

fault tolerance
extensive and independence assumption (more extensive to ensure independence)

Different concept for fault tolerance:
recoverability
detect failure (fail fast)
recover from the failure

forward recovery vs backward recovery
forward
    application specific
backward
    more general
    not always applicable
        side effects, e.g., cannot “roll back” the cash ATM gives out
doesn’t help w/ deterministic problems
    crash, roll-back, run again, hit the same problem, repeat!

transfer(from_acct, to_acct, amount)
read_disk (from_bal, from_acct)
    from_bal = from_bal - amount
write_disk (from_bal, from_acct)

→ crash here? // need all or nothing atomicity

read_disk (to_bal, from_acct)
    to_bal = to_bal + amount
write_disk (to_bal, to_acct)
Atomicity

Atomic action (or transaction in database lingo)
hide the fact that atomic action might consist of multiple steps; great modularity

Two types:
- All-or-nothing atomicity -> for failure recovery; recoverability
- Before-or-after atomicity -> sequencing of concurrent activities; isolation

All-or-nothing atomicity
“A sequence of steps is an all-or-nothing action if, from the point of view of its invoker, the sequence always either
• completes, or
• aborts
in such a way that it appears that the sequence had never been undertaken in the first place. That is, it backs out.”

Before-or-after atomicity
“Concurrent actions have the before-or-after property if their effect from the point of view of their invokers is the same as if the actions occurred either completely before or completely after one another.”

Other properties:
- consistency: durability: the data written by a transaction should last (forever)

ACID transactions:
- A: Atomicity: all-or-nothing property
- C: Consistency: maintain application-level invariants
- I: Isolation: before-or-after property
- D: Durability: the data written by a transaction should last (forever)

Recoverability
1. fail fast
design fail fast components
detect errors
2. repair
   run some recovery procedure
3. restart

**Recoverable sector**
recoverable put/get for a single sector
use recoverable sector as a building block to implement:

app/os -> buffer -> disk
   protect against: buffer is corrupted so bad data written to disk

careful put/get (sec, data) // now no need to worry about hw failures
if something fails in the middle, you are stuck unless you have a copy of the data

build a recoverable sector out of three sectors
   s1, s2, s_valid
   s1 is the copy to be read
   recoverable_put will write to s2
   change s_valid if everything goes well
   if failure in the middle then we are ok

   what will happen during the middle of writing s_valid?
   read of the chooser will return invalid, then crash must have happened during its write

   **golden rule: never ever modify the only copy!**