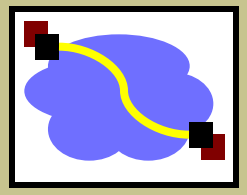
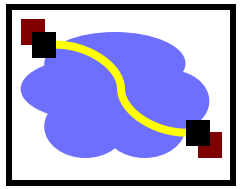


416 Distributed Systems

RAID, Feb 16 2018

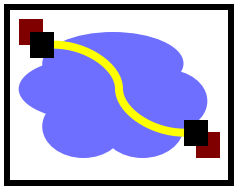
Thanks to Greg Ganger and Remzi Arapaci-Dusseau
for slides

Outline



- Using multiple disks
  - Why have multiple disks?
  - problem and approaches
- RAID levels and performance

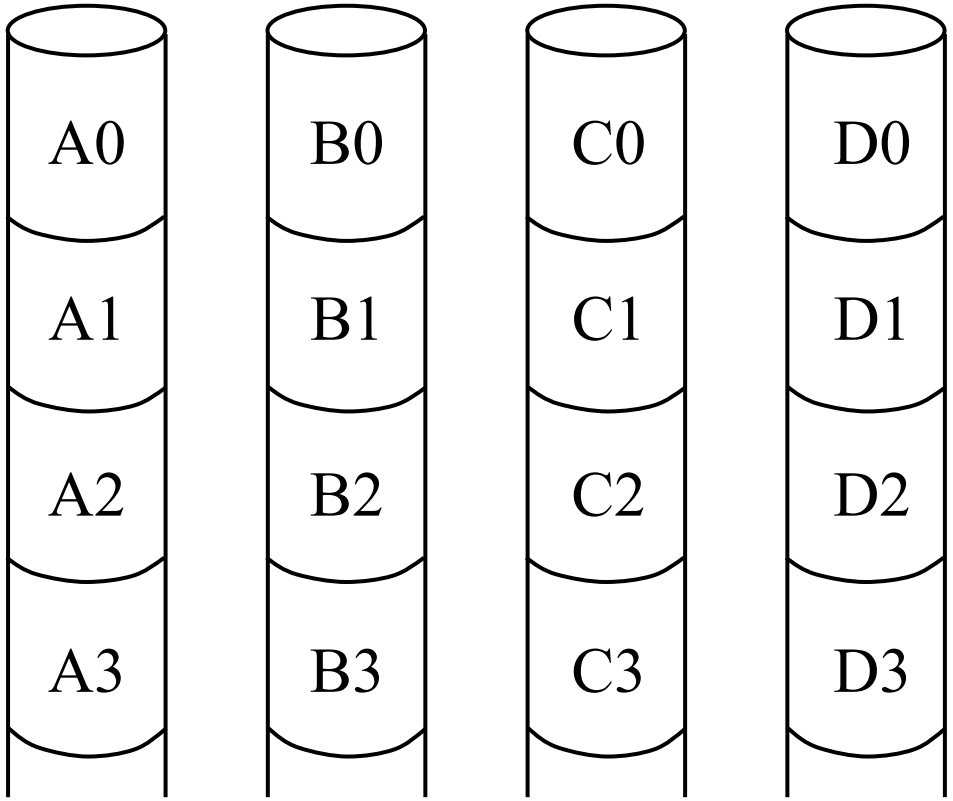
Motivation: Why use multiple disks?



- Capacity
  - More disks allows us to store more data
- Performance
  - Access multiple disks in parallel
  - Each disk can be working on independent read or write
  - Overlap seek and rotational positioning time for all
- Reliability
  - Recover from disk (or single sector) failures
  - Will need to store multiple copies of data to recover

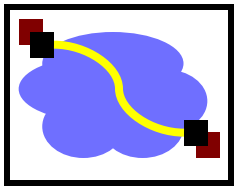
- So, what is the simplest arrangement?

Just a bunch of disks (JBOD)



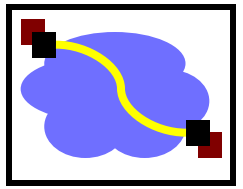
- Yes, it's a goofy name
  - industry really does sell "JBOD enclosures"

Disk Subsystem Load Balancing

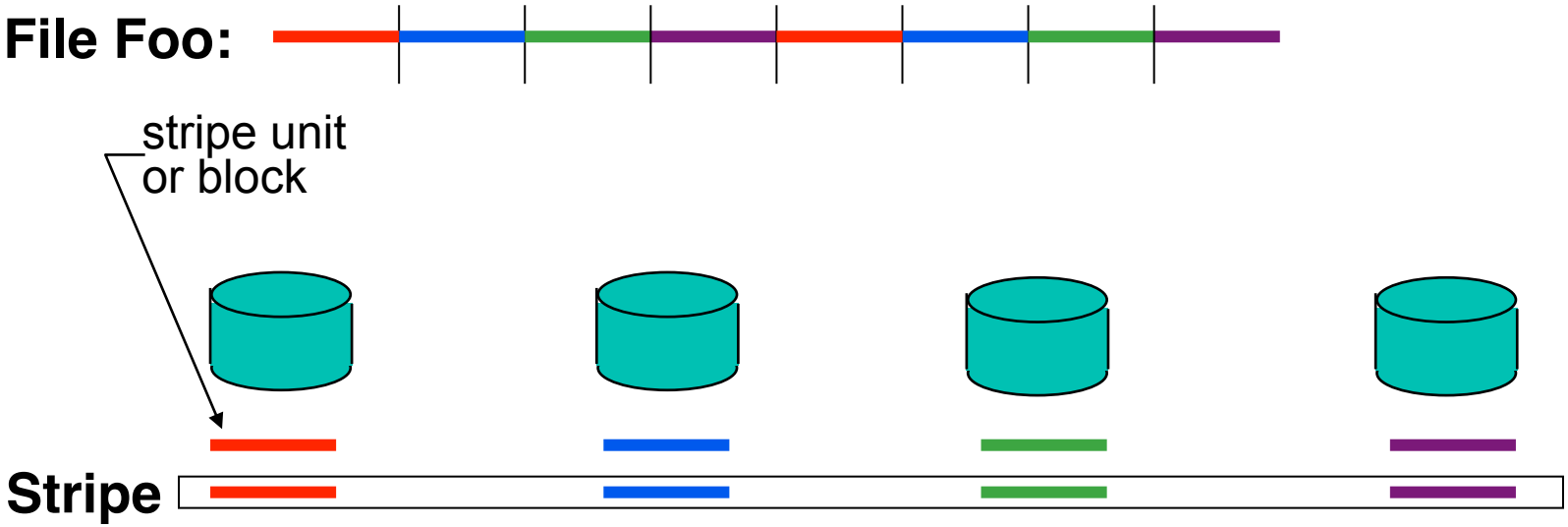


- I/O requests are almost never evenly distributed
  - Some data is requested more than other data
  - Depends on the apps, usage, time, ...
- What is the right data-to-disk assignment policy?
  - Common approach: Fixed data placement
    - Your data is on disk X, period!
    - For good reasons too: you bought it or you're paying more...
  - Fancy: Dynamic data placement
    - If some of your files are accessed a lot, the admin(or even system) may separate the "hot" files across multiple disks
      - In this scenario, entire files systems (or even files) are manually moved by the system admin to specific disks
  - Alternative: Disk striping
    - Stripe all of the data across all of the disks

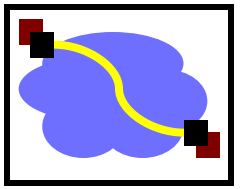
Disk Striping



- Interleave data across multiple disks
  - Large file streaming can enjoy parallel transfers
  - High throughput requests can enjoy thorough load balancing
    - If blocks of hot files equally likely on all disks (really?)

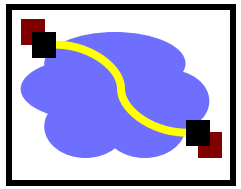


Disk striping details



- How disk striping works
  - Break up total space into fixed-size stripe units
  - Distribute the stripe units among disks in round-robin
  - Compute location of block #B as follows
    - disk# = B%N (%=modulo,N = #ofdisks)

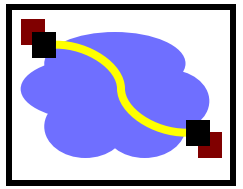
Now, What If A Disk Fails?



- In a JBOD (independent disk) system
  - one or more file systems lost
- In a striped system
  - a part of each file system lost

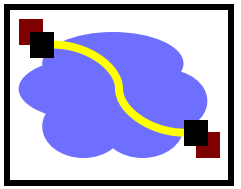
- Backups can help, but
  - backing up takes time and effort
  - backup doesn't help recover data lost during that day
    - Any data loss is a big deal to a bank or stock exchange

Tolerating and masking disk failures

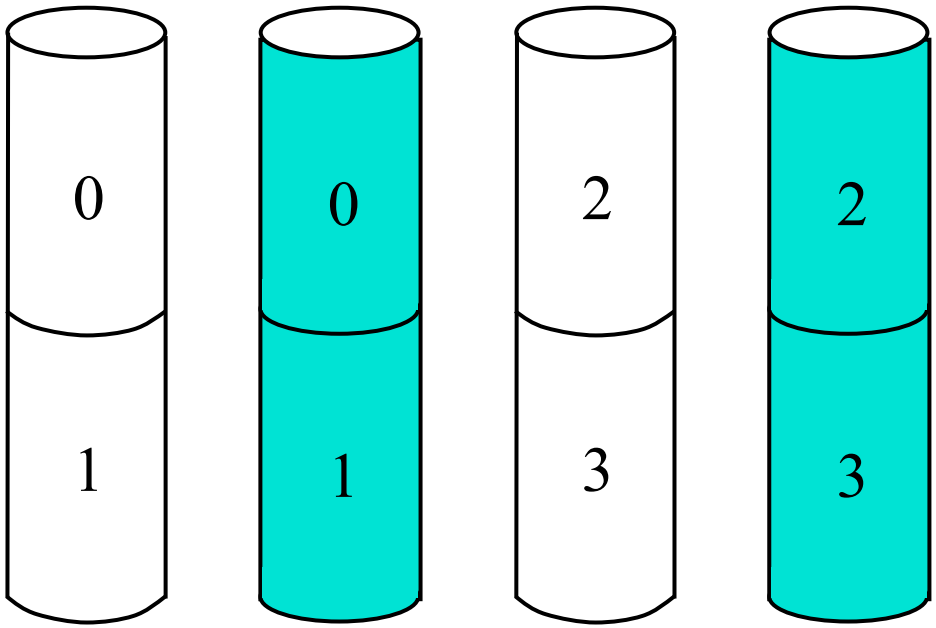


- If a disk fails, it's data is gone
  - may be recoverable, but may not be
- To keep operating in face of failure
  - must have some kind of data redundancy
- Common forms of data redundancy
  - replication
  - error-correcting codes

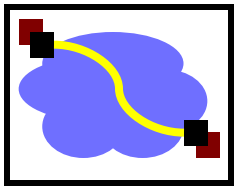
Redundancy via replicas



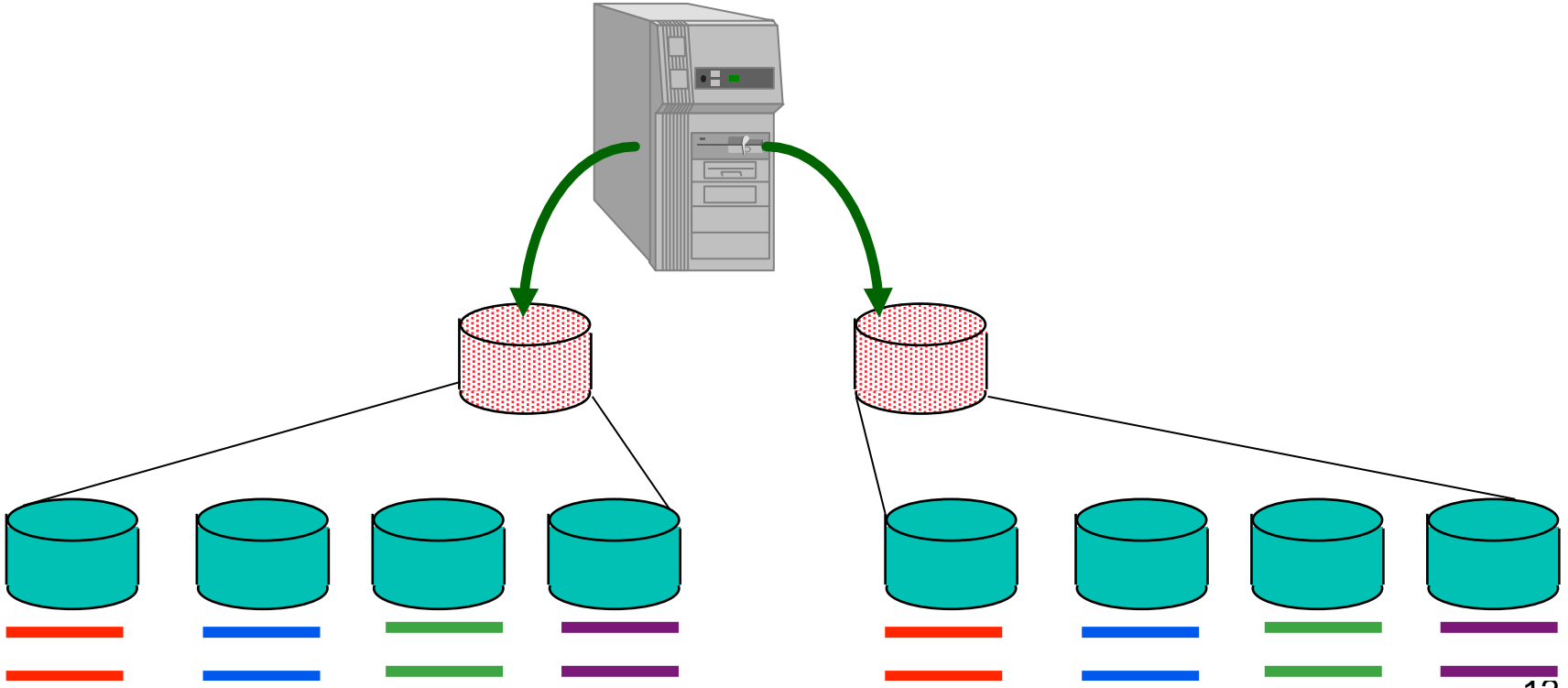
- Two (or more) copies
  - mirroring, shadowing, duplexing, etc.
- Write both, read either



Mirroring & Striping

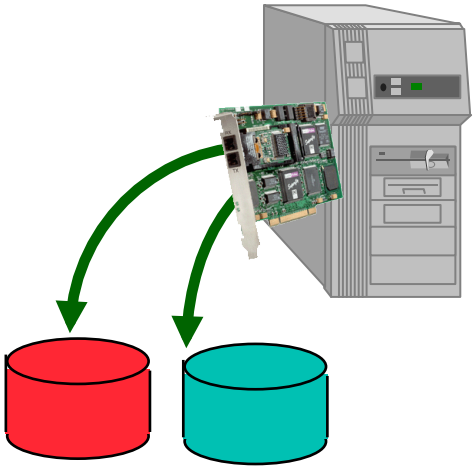


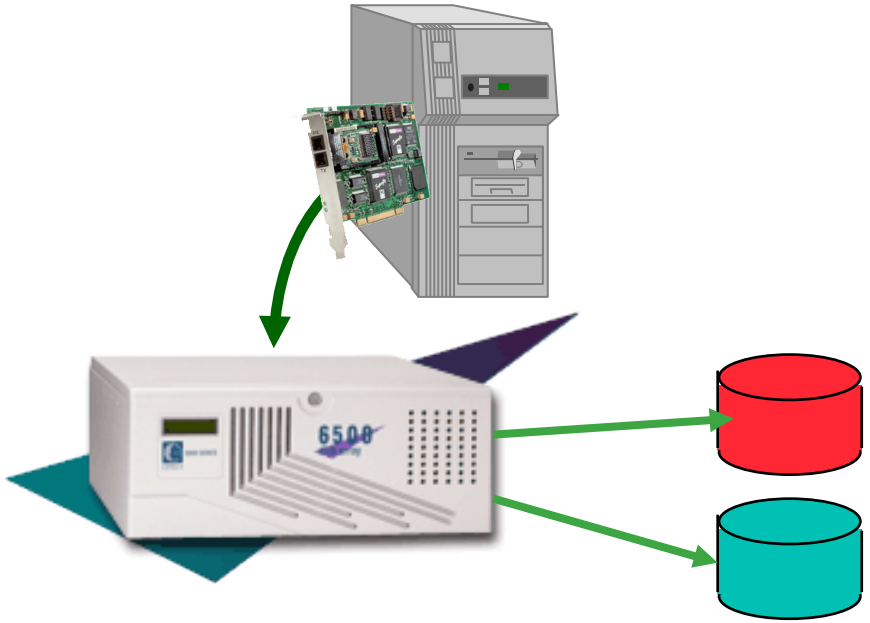
- Mirror to 2 virtual drives, where each virtual drive is really a set of striped drives
  - Provides reliability of mirroring
  - Provides striping for performance (with write update costs)



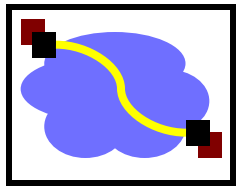
Implementing Disk Mirroring

- Mirroring can be done in either software or hardware
- Software solutions are available in most OS's
- Hardware solutions
  - Could be done in Host Bus Adaptor(s)
  - Could be done in Disk Array Controller



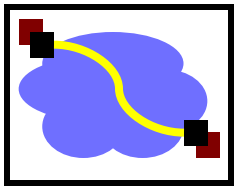


Lower Cost Data Redundancy

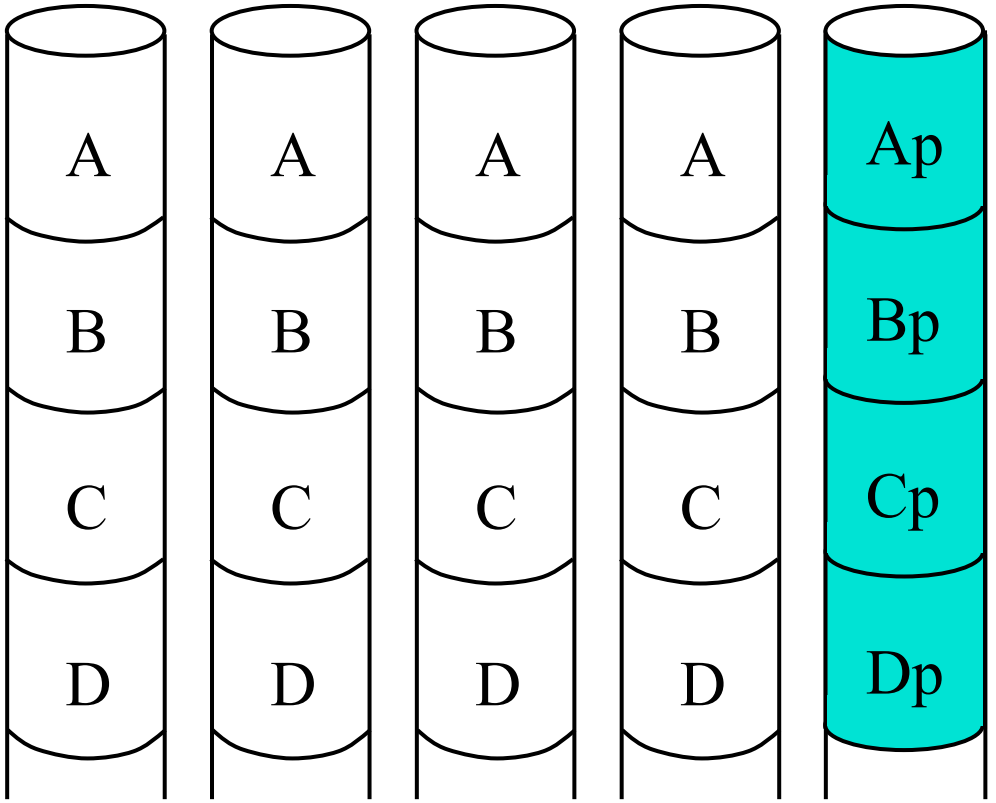


- Single failure protecting codes
  - general single-error-correcting code is overkill
    - General code finds error and fixes it
- Disk failures are self-identifying (a.k.a. erasures)
  - Don’t have to find the error
- Parity is single-disk-failure-correcting code
  - recall that parity is computed via XOR
  - it’s like the low bit of the sum

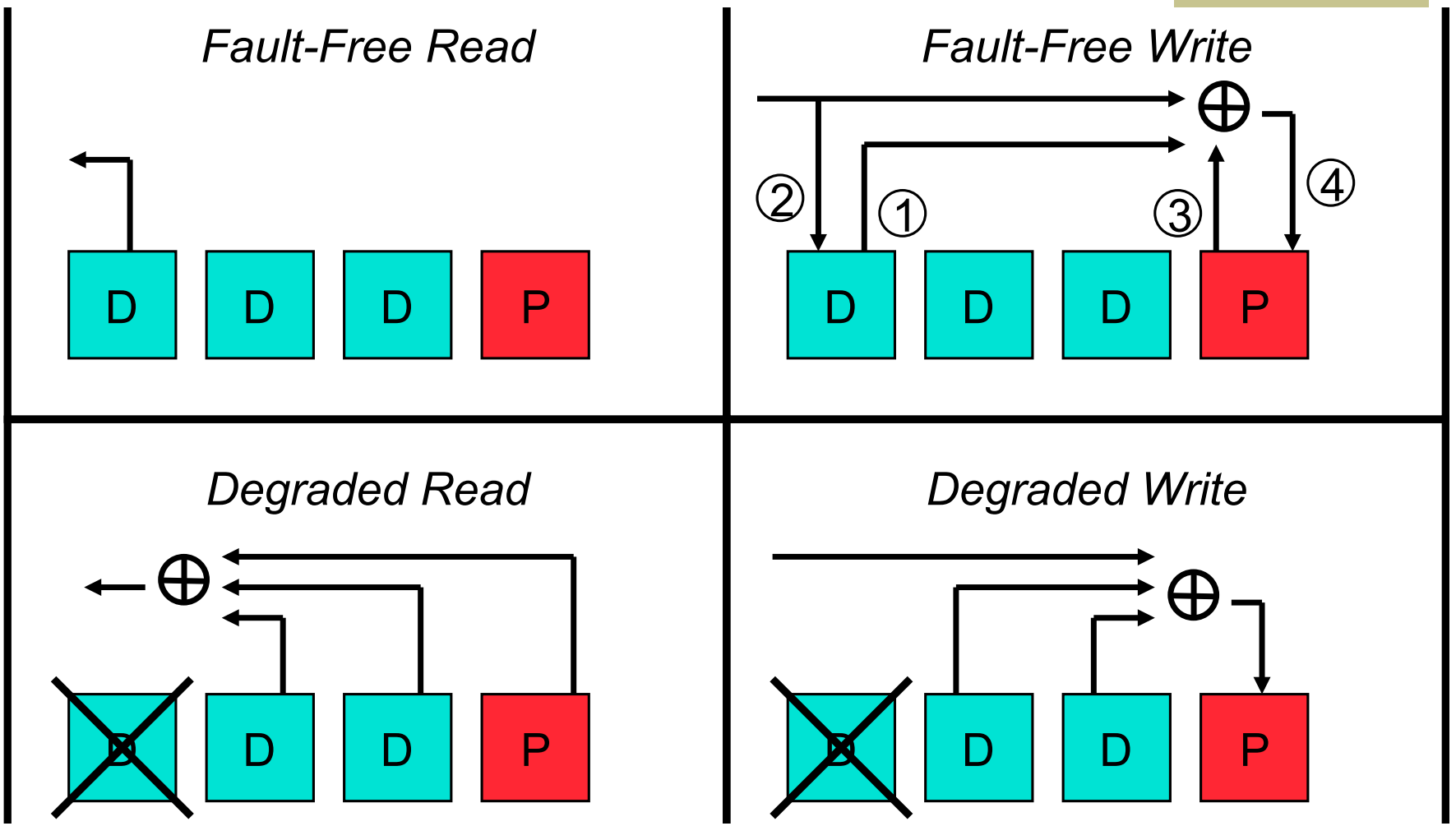
Simplest approach: Parity Disk



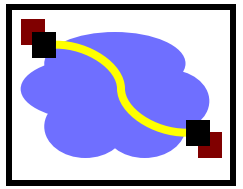
- One extra disk
- All writes update parity disk
  - Potential bottleneck
  - (different data in different As, Bs, Cs, Ds)
  - (Ap contains parity for all As)



Updating and using the parity



The parity disk bottleneck



- Reads go only to the data disks
  - But, hopefully load balanced across the disks
- All writes go to the parity disk
  - And, worse, usually result in Read-Modify-Write sequence
  - So, parity disk can easily be a bottleneck