RAID

- **Redundant Arrays of Inexpensive Disks**
  - Using lots of disk drives improves:
    - Performance
    - Reliability
  - Alternative: Specialized, high-performance hardware
  - RAID delivers better price/performance than high-end disks

- **Performance**
  - Read data from n disks at once → reads are n times faster

- **Reliability**
  - Store multiple copies of data
  - If one disk fails, no data is lost and the system continues to run

- **Three main concepts**
  - Mirroring
  - Striping
  - Parity
Mirroring

• Use two disks that are identical copies of each other
  – Primary goal: fault-tolerance
    • If one disk fails, use the other one
  – Writes must be done to both disks at once
  – Improved random read performance
    • Can do two random reads at one time
  – Sequential read performance mostly unaffected

This Is What Mirroring Looks Like

This Is What Mirroring Looks Like
Striping

- Spread data across n disks
- First disk gets blocks 1, n+1, 2n+1, etc.
- Second disk gets blocks 2, n+2, 2n+2, etc.
- Improved random read performance
  - Can do as many as n reads at the same time
  - But each read must go to a specific disk
  - Thus multiple reads can conflict if unlucky
- Sequential reads are very fast
  - Especially for long reads (many blocks from each disk)
  - Read in parallel from all disks
- Each write goes to a single disk
Parity

• Mirroring delivers fault-tolerance through redundancy
• Storage utilization is rather poor
  – Only 50% of disk capacity is useful
  – The other 50% is overhead for fault tolerance
• Parity checks deliver fault-tolerance with less redundancy
  – Use n+1 disks
  – Store data on n of the disks
  – Last disk contains parity data
    • XOR of other n disks
    • Compare ith bit on each disk
    • Even number of 1s → ith parity bit is 0
    • Odd number of 1s → ith parity bit is 1
  – Any one disk fails → no data is lost
Parity Example

• Three servers + 1 parity server
  – Server 1 stores “110011”
  – Server 2 stores “011011”
  – Server 3 stores “110101”
  – Server P stores “011101”
    • Number of 1s = 2,3,1,1,2,3
    • Even, Odd, Odd, Odd, Even, Odd

• Suppose Server 2 fails
  – “110011”, “??????”, “110101”, “011101”
  – Take XOR of remaining servers to reconstruct
    • Number of 1s = 2,3,1,2,1,3
    • Even,odd,odd,even,odd,odd
    • 011011
RAID Levels

• RAID 0
  – Striping (without parity)
  – Pros:
    • Good performance
    • No redundancy (no wasted capacity)
  – Cons:
    • Poor fault-tolerance (worse than no RAID!)

• RAID 1
  – Mirroring
  – Pros:
    • Good fault-tolerance
    • Very fast recovery
  – Cons:
    • Wastes storage capacity
    • Performance not as good as other RAID levels
RAID Levels

- RAID 2: Not used.
- RAID 3 and 4:
  - Striping with dedicated parity disk
  - Stripe size = byte for RAID 3, block for RAID 4
  - Pros:
    - Good performance
    - Good fault-tolerance with little redundancy
    - Reasonably fast recovery
  - Cons:
    - Parity disk is a bottleneck for writes

This You Data Level

Is Would Using Four.

How Store RAID

(Parity 1)
(Parity 2)
(Parity 3)
(Parity 4)
RAID Levels

- **RAID 5**
  - Striping with distributed parity
  - Servers “take turns” being the parity server
  - Pros and Cons similar to RAID 3 and 4
    - Avoids write bottleneck associated with RAID 3 and 4
    - Performance degrades following disk failure
Multi-Level RAID

• The RAID ideas can be hierarchically combined
• Most common combination are:
  – RAID 1+0 – stripes of mirrors
  – RAID 0+1 – mirror of stripes
**RAID 1+0 vs. RAID 0+1**

- Difference is what happens when a disk fails
  - RAID 1+0
    - One stripe becomes unmirrored
    - Failure of the other disk in that stripe leads to data loss
  - RAID 0+1
    - One mirror becomes invalid
    - Failure of any disk in the other stripe leads to data loss