Redundant Array of Inexpensive Disks (RAID)

Kartik Gopalan

Chapter 5 Tanenbaum's book

First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

RAID — Original Motivation

- Replacing large and expensive mainframe hard drives (IBM 3310) by several cheaper Winchester disk drives

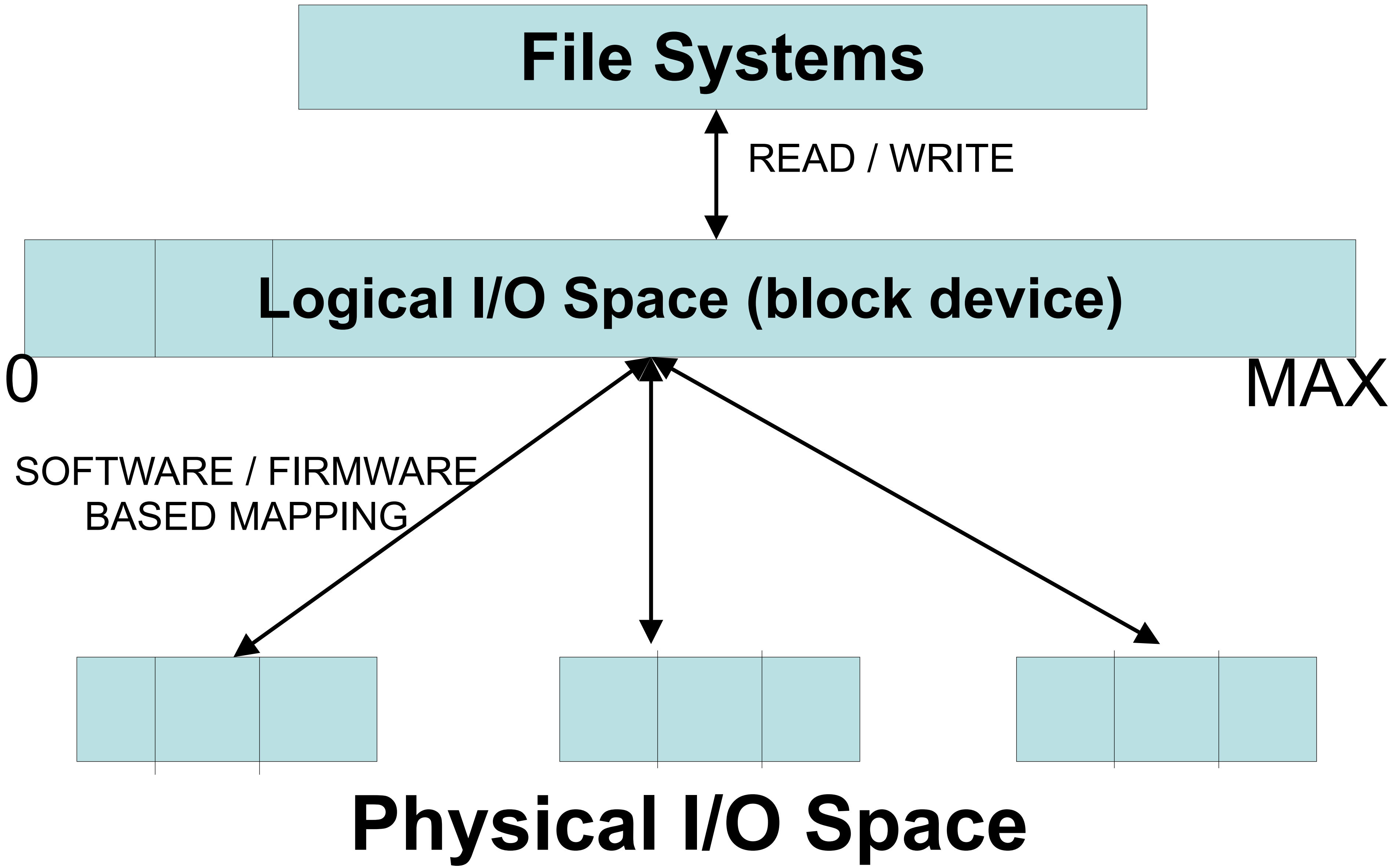
- Will work but introduces a data reliability problem:
  - Consider Mean Time To Failure (MTTF)
  - Assume MTTF of a disk drive is 30,000 hours
  - MTTF for a set of *n* drives is 30,000/*n*
    - n = 10 means MTTF of 3,000 hours

RAID — Today's Motivation

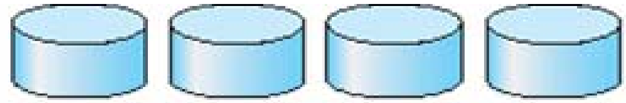
- "Cheap" hard drives are now big enough for most applications

- We use RAID today for
  - Increasing disk throughput by allowing parallel access
  - Eliminating the need to make disk backups
    - Disks are too big to be backed up efficiently

Logical-to-Physical I/O Address Space Mapping

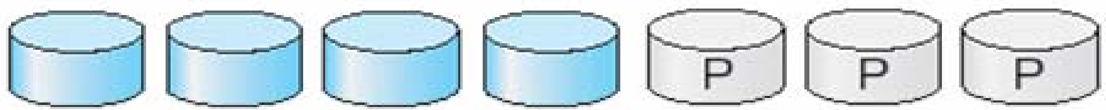


Several Levels of RAID

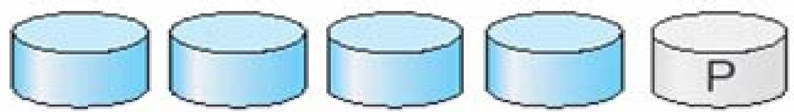


(a) RAID 0: non-redundant striping.

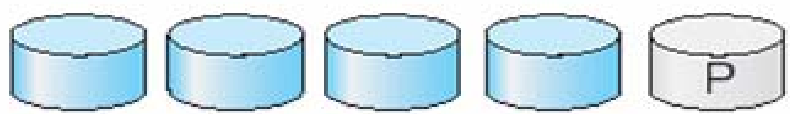
(b) RAID 1: mirrored disks.



(c) RAID 2: memory-style error-correcting codes.



(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

RAID 0

- Striping
  - Spread the data over multiple disk drives
- No fault tolerance
- But, much better I/O throughput
  - Number of I/O operations per second

DISK 0	DISK 1	DISK 2	DISK 3
BLOCK 0	BLOCK 1	BLOCK 2	BLOCK 3
BLOCK 4	BLOCK 5	BLOCK 6	BLOCK 7
BLOCK 8	BLOCK 9	BLOCK 10	BLOCK 11
BLOCK 12	BLOCK 13	BLOCK 14	BLOCK 15

RAID 1

- Mirroring
  - Two copies of each disk block
- Advantage
  - Simple to implement
  - Fault-tolerant
- Disadvantage
  - Requires twice the disk capacity

DISK 0	DISK 1	MIRROR 0	MIRROR 1
BLOCK 0	BLOCK 1	BLOCK 0	BLOCK 1
BLOCK 2	BLOCK 3	BLOCK 2	BLOCK 3
BLOCK 4	BLOCK 5	BLOCK 4	BLOCK 5
BLOCK 6	BLOCK 7	BLOCK 6	BLOCK 7

RAID 2

- Use an **error (detection + correction)** code instead of duplicating the data blocks
- Meant for disks that don't have built-in error detection.
- Modern disks support built-in error detection, so this level is mostly unused.

DISK 0	DISK 1	DISK 2	PARITY 1	PARITY 2
BLOCK 0	BLOCK 1	BLOCK 2	F(0,1,2)	
BLOCK 3	BLOCK 4	BLOCK 5	F(3,4,5)	
BLOCK 6	BLOCK 7	BLOCK 8	F(6,7,8)	
BLOCK 9	BLOCK 10	BLOCK 11	F(9,10,11)	

F = FUNCTION FOR ERROR DETECTION + CORRECTION

XOR Primer

- Truth Table
  - 0 XOR 0 = 0
  - 0 XOR 1 = 1
  - 1 XOR 0 = 1
  - 1 XOR 1 = 0

- Associative and commutative

- Lost any one bit? XOR the rest to recover.
  - If 1 XOR (lost bit) = 0
  - then lost bit = 1 XOR 0 = 1

- Extends to any number of data bits
  - If 1 XOR (lost bit) XOR 1 = 0
  - then lost bit = 1 XOR 1 XOR 0 = 0

RAID 3

- N+1 disk drives: N data drives + 1 parity drive
- Data Block b[k] partitioned into N fragments b[k,1], b[k,2], ... b[k,N]
- Parity drive contains **XOR** (**exclusive or)** of these N fragments
  - p[k] = b[k,1] **XOR** b[k,2] **XOR ... XOR** b[k,N]
- Upon a failure, reconstruct the lost fragments by XOR of corresponding fragments from remaining drives.
  - b[k,i] = p[k] **XOR** b[k,1] **XOR ...** b[k,i-1] **XOR** b[k,i+1] **… XOR** b[k,N]
- Simple to implement in firmware/software
- Permits only one I/O operation at a time over entire array

DISK 0	DISK 1	DISK 2	PARITY
BLOCK [0,1]	BLOCK [0,2]	BLOCK [0,3]	PARITY 0
BLOCK [1,1]	BLOCK [1,2]	BLOCK [1,3]	PARITY 1
BLOCK [2,1]	BLOCK [2,2]	BLOCK [2,3]	PARITY 2
BLOCK [3,1]	BLOCK [3,2]	BLOCK [3,3]	PARITY 3

RAID 4

- Requires N+1 disk drives (as in RAID 3)
  - N data drives + 1 Parity drive
- Data striped at block granularity (as in RAID 0)
  - Disk 1 has block 1, disk 2 has block 2, and so on.
- Parity drive contains **exclusive or** of the N blocks in stripe
  - p[k] = b[Nk] **XOR** b[Nk+1] **XOR ... XOR** b[Nk+N-1]
- Multiple Read I/O operations can be processed in parallel
- But how about parallel writes I/O operations?

DISK 0	DISK 1	DISK 2	PARITY
BLOCK 0	BLOCK 1	BLOCK 2	PARITY 0
BLOCK 3	BLOCK 4	BLOCK 5	PARITY 1
BLOCK 6	BLOCK 7	BLOCK 8	PARITY 2
BLOCK 9	BLOCK 10	BLOCK 11	PARITY 3

RAID 5

- Single parity drive of RAID-4 is involved in every write
  - Will limit write parallelism
  - Exercises one parity disk more than others
- Solution in RAID-5
  - Distribute the parity blocks among all N+1 drives
- Up to N/2 parallel writes

DISK 0	DISK 1	DISK 2	DISK 3
BLOCK 0	BLOCK 1	BLOCK 2	PARITY 0
BLOCK 3	BLOCK 4	PARITY 1	BLOCK 5
BLOCK 6	PARITY 2	BLOCK 7	BLOCK 8
PARITY 3	BLOCK 9	BLOCK 10	BLOCK 11

The write problem

- Every time a block is updated, the parity must be updated as well.
- Assume we want to update the kth block bk_{old} to bk_{new}
- Before writing bk_{new}
    - p_{old} = $b0_{old}$ **XOR** $b1_{old}$ **XOR ...** bk_{old} … XOR bN_{old}
- After writing bk_{new} , we can naively recompute p_{new} as follows
    - p_{new} = $b0_{old}$ **XOR** $b1_{old}$ **XOR ...** bk_{new} … XOR bN_{old}

- Naive solution incurs <u>high overhead</u>
    - N-1 reads
        - Read all old data blocks except the block being written (bk_{new})
    - 2 writes
        - bk_{new} and p_{new}

Second (smarter) solution

- Assume we want to update the kth block bk_{old} to bk_{new}
- Before writing bk_{new}

  (A) p_{old} = $b0_{old}$ **XOR** $b1_{old}$ **XOR ...** bk_{old} … XOR bN_{old}

- Moving bk_{old} to left hand side

  (B) p_{old} XOR bk_{old} = $b0_{old}$ **XOR** $b1_{old}$ **XOR ...** XOR bN_{old}

- Naive solution to compute p_{new}

  (C) p_{new} = $b0_{old}$ **XOR** $b1_{old}$ **XOR ...** bk_{new} … XOR bN_{old}

- Moving bk_{new} to left hand side

  (D) p_{new} XOR bk_{new} = $b0_{old}$ **XOR** $b1_{old}$ **XOR ...** XOR bN_{old}

- Combining (B) and (D)

  (E) p_{new} XOR bk_{new} = p_{old} XOR bk_{old}

- Smarter solution: moving bk_{new} to right hand side
  - p_{new} = bk_{new} XOR p_{old} XOR bk_{old}
- Smarter Solution requires
  - 2 reads
    - bk_{old} and p_{old}
    - Or just one read of p_{old} if bk_{old} was read into memory earlier
  - 2 writes
    - bk_{new} and p_{new}

Comparison

	RAID 0 (N disks)	RAID 1 (N disks)	RAID 2 (N+1) disks	RAID 3 (N+1) disks	RAID 4 (N+1) disks	RAID 5 (N+1) disks
Fault-tolerance	None	All 1-disk and most 2-disk failures	1-disk failure with error detection and correction	1-disk failure with Error Correction	1-disk failure with Error Correction	1-disk failure with Error Correction
Max. READ Parallelism	N	N	N	1 (none)	N	N+1
Max. WRITE Parallelism	N	N/2	1 (none)	1 (none)	1 (none)	(N+1)/2
Space Overhead	0%	100%	(k/N)x100% for K parity disks	(1/N)x100%	(1/N)x100%	(1/N)x100%

Conclusion

- RAID original purpose was to take advantage of commodity drives that were smaller and cheaper than conventional disk drives
    - Replace a single large drive by an array of smaller drives

- Nobody does that anymore!

- Today: Main purpose of RAID is to build **fault-tolerant** storage systems that do not need backups and deliver **high throughput**.

- Low cost of disk drives makes RAID-1 attractive for small installations
    - We have now very cheap RAID controllers
- Otherwise prefer
    - RAID-3 for simplicity
    - RAID-5 for higher parallelism

- Often combined with NVRAM to improve write performance