Distributed Video Systems
Chapter 5
Issues in Video Storage and Retrieval
Part 2 - Disk Array and RAID

Jack Yiu-bun Lee
Department of Information Engineering
The Chinese University of Hong Kong

Contents

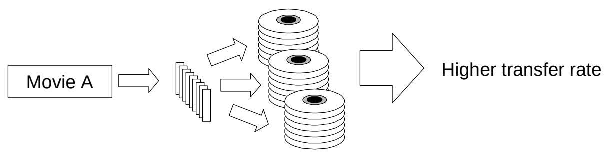
5.1 Disk Array Basics

Jack Y.B. Lee

- Data Striping
  - Spindle-synchronized mode

One logical stripe (B_0)

	D_0	D_1	D_2
B_0	b_0	b_1	b_2
B_1	b_3	b_4	b_5
B_2	b_6	b_7	b_8
B_3	b_9	b_{10}	b_{11}
B_4	b_{12}	b_{13}	b_{14}



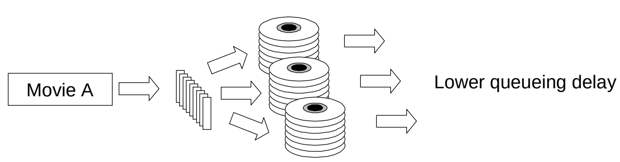
5.1 Disk Array Basics

Jack Y.B. Lee

- Data Striping
  - Split-schedule mode

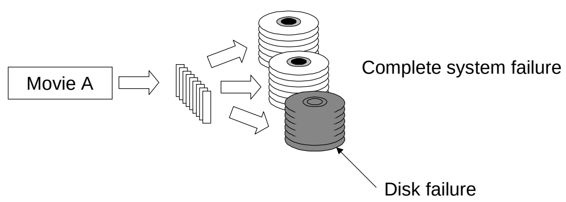
One logical unit of a disk (B_0 on D_0)

	D_0	D_1	D_2
B_0	b_0	b_1	b_2
B_1	b_3	b_4	b_5
B_2	b_6	b_7	b_8
B_3	b_9	b_{10}	b_{11}
B_4	b_{12}	b_{13}	b_{14}



5.1 Disk Array Basics

- Reliability
  - While disk array can break the disk I/O bottleneck, the overall reliability becomes a problem.



- Solution
  - Redundant Disk Arrays

5.2 Redundant Array of Inexpensive Disks

- History
  - First proposed by Patterson et al. in 1988.
  - Defined RAID levels 1 to 5
- Principles
  - Compute redundant data from existing data so that data lost in a failed disk can be recovered by computation.
  - A method for computing redundant data is needed
    - Example: Parity, Hamming code, and Reed-Solomon codes.
  - A method for distributing redundant data is needed
    - Centralized and distributed

5.2 Redundant Array of Inexpensive Disks

- RAID Organizations
  - RAID Level 0 (Non-Redundant)

	D_0	D_1	D_2
B_0	b_0	b_1	b_2
B_1	b_3	b_4	b_5
B_2	b_6	b_7	b_8
B_3	b_9	b_{10}	b_{11}
B_4	b_{12}	b_{13}	b_{14}

  - Best write performance (why?)
  - But *not* the best read performance (why?)
  - Widely used in supercomputing environments where performance and capacity is more important than reliability.

5.2 Redundant Array of Inexpensive Disks

- RAID Organizations
  - RAID Level 1 (Mirrored)

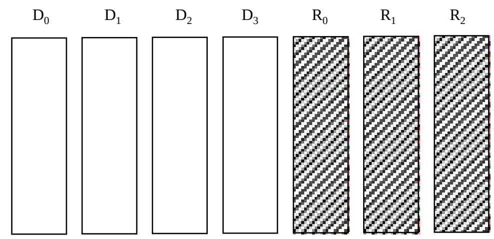
	D_0	D_1
B_0	b_0	b_0
B_1	b_1	b_1
B_2	b_2	b_2
B_3	b_3	b_3
B_4	b_4	b_4

  - 100% storage overhead.
  - Best read performance (why?)
  - Widely used in database applications where availability and transaction rate are more important than storage efficiency.

5.2 Redundant Array of Inexpensive Disks

Jack Y.B. Lee

- RAID Organizations
  - RAID Level 2 (Memory-Style ECC)

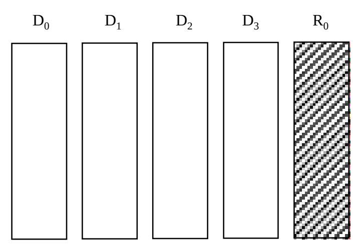


- Hamming coded, <100% storage overhead;
  More disks -> lower storage overhead.
- Requires spindle synchronization.
- Disk-failure detection is not needed.
- Unpopular.

5.2 Redundant Array of Inexpensive Disks

Jack Y.B. Lee

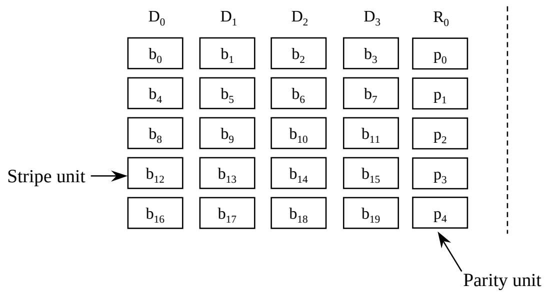
- RAID Organizations
  - RAID Level 3 (Bit-Interleaved Parity)



Parity disk is not used for reading during normal operation.

- Parity coded, storage overhead = 1/(*N*-1).
- Requires spindle synchronization.
- Disk-failure detection is needed.
- Tolerable to single-disk failure.
- Suitable for high-bandwidth applications.

- RAID Organizations
  - RAID Level 4 (Block-Interleaved Parity)



Parity Calculation

$p_0 = b_0 \oplus b_1 \oplus b_2 \oplus b_3$

$p_1 = b_4 \oplus b_5 \oplus b_6 \oplus b_7$

$p_2 = b_8 \oplus b_9 \oplus b_{10} \oplus b_{11}$

$p_3 = b_{12} \oplus b_{13} \oplus b_{14} \oplus b_{15}$

$p_4 = b_{16} \oplus b_{17} \oplus b_{18} \oplus b_{19}$

- Parity coded, storage overhead = 1/(N-1).
- Disk-failure detection is needed.
- Tolerable to single-disk failure.

- RAID Organizations
  - RAID Level 4 (Block-Interleaved Parity)
    - Small reads
      - Involves one of the disks, hence split-schedule is desirable in this case.
    - Large reads
      - Span more disks, reading in parallel improves transfer rate.
    - Large writes
      - Span all disks, parity can be computed from the new data.
    - Small writes
      - Span a single disk:
        - Read old data
        - Read parity
        - Compute new parity using old data, new data, & parity
        - Write new data
        - Write new parity

5.2 Redundant Array of Inexpensive Disks

Jack Y.B. Lee

- RAID Organizations
  - RAID Level 4 (Block-Interleaved Parity)
    - Small writes
      - 4 I/Os are needed for a single write!
      - 2 of the 4 I/Os are performed on the parity disk.
      - There is only one parity disk and hence it often becomes the bottleneck in writing.
    - Read after disk failure
      - Large reads
        - Read whole stripes and use parity for lost stripe recovery.
      - Small reads
        - Still need to read whole stripe in case the needed stripe is in the failed disk.

5.2 Redundant Array of Inexpensive Disks

Jack Y.B. Lee

- RAID Organizations
  - RAID Level 5 (Block-Interleaved Distributed Parity)

D_0	D_1	D_2	D_3	D_4
b_0	b_1	b_2	b_3	$\mathbf{p_0}$
b_5	b_6	b_7	$\mathbf{p_1}$	b_4
b_{10}	b_{11}	$\mathbf{p_2}$	b_8	b_9
b_{15}	$\mathbf{p_3}$	b_{12}	b_{13}	b_{14}
$\mathbf{p_4}$	b_{16}	b_{17}	b_{18}	b_{19}

Left-Symmetric Parity Placement

Parity Calculation

$p_0 = b_0 \oplus b_1 \oplus b_2 \oplus b_3$

$p_1 = b_4 \oplus b_5 \oplus b_6 \oplus b_7$

$p_2 = b_8 \oplus b_9 \oplus b_{10} \oplus b_{11}$

$p_3 = b_{12} \oplus b_{13} \oplus b_{14} \oplus b_{15}$

$p_4 = b_{16} \oplus b_{17} \oplus b_{18} \oplus b_{19}$

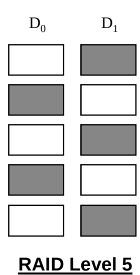
- Parity units are distributed across all disks.
- Removed the parity-disk bottleneck.
- Read performance better than RAID Level 4. (Why?)

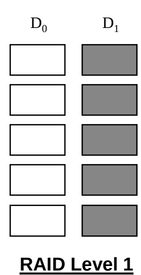
5.2 Redundant Array of Inexpensive Disks

- RAID Organizations
  - RAID Level 6 (P+Q Redundancy)
    - Similar to RAID 5
      - Distributed parity
      - Block interleaved
      - Read-modify-write
    - Differences
      - Uses Reed-Solomon codes to protect against double-disk failure using two more disks.
      - In small writes, 6 I/Os (instead of 4) are required.

5.3 Performance and Cost Comparisons

- Similarity of different RAID levels
  - RAID 5 with 2 disks is very similar to RAID 1
    - Differences are mainly implementations like
      - Parity versus replica
      - Scheduling algorithms
      - Disk layout optimizations





5.3 Performance and Cost Comparisons

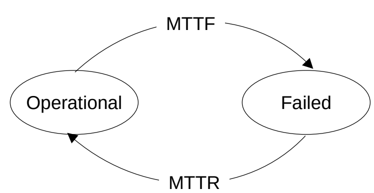
Jack Y.B. Lee

- Similarity of different RAID levels
  - RAID 5 with stripe unit much smaller than the average request size is very similar to RAID 3
    - Because a request always involve all disks in the array.
    - Parity placement is different.
- In Practice
  - RAID 2 and RAID 4 are usually inferior to RAID 5
  - RAID 5 can simulate RAID 1 and 3
  - So in most cases the problem is just selecting different stripe sizes and parity group sizes for RAID 5.

5.4 Reliability

Jack Y.B. Lee

- Device Reliability
  - Metric
    - MTTF
      - **M**ean-**T**ime-**T**o-**F**ailure (usually measured in hours)
    - MTTR
      - **M**ean-**T**ime-**T**o-**R**epair (ditto)
    - MTBF
      - **M**ean-**T**ime-**B**etween-**F**ailures (ditto)



5.4 Reliability

- Device Reliability
  - RAID Level 5
    - MTBF

$$MTBF_{array} = \frac{(MTTF_{disk})^2}{N(G-1)MTTR_{disk}}$$

Single-disk

Parity group size

Number of disks

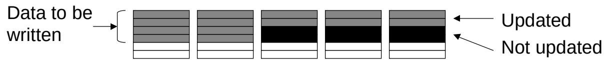
- Example
  - N=100, G=16, $MTTF_{disk}$=200,000hrs, $MTTR_{disk}$=1hr
  - $MTBF_{array}$=3,000 years!
  - *So a RAID can in fact be more reliable than a disk!*

5.4 Reliability

- System Crashes and Parity Inconsistency
  - System Crashes
    - Power failure, operator error, hardware breakdown, etc.
  - Implications
    - Disk I/O operations can be interrupted
    - Write operations are affected because
      - data blocks are updated but not parity block
      - and vice versa.
  - Consequences
    - The data block to be written could be corrupted.
    - The associated parity block could become inconsistent and cannot be used to recover lost data in case of a disk failure. (More serious, why?)

5.4 Reliability

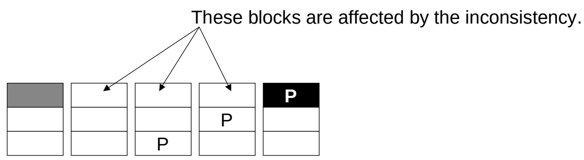
- System Crashes and Parity Inconsistency
  - For Bit-Interleaved RAID Levels (e.g. 3)
    - Inconsistency only affects the data being written.



    - This is because a stripe is small and a write operation usually spans many stripes.
    - Nothing need to be done as data consistency after system crash is not guaranteed in non-RAID disks anyway.
    - It is up to the application to deal with the erroneous data.

5.4 Reliability

- System Crashes and Parity Inconsistency
  - For Block-Interleaved RAID Levels (e.g. 5)
    - Inconsistency can affect other data not related to the one being written.



The data block is updated but the system crashes before the parity block is updated.

    - In hardware RAID controllers, non-volatile RAM is used to temporary store the parity information before the write operation to guard against system crashes.
    - Hence software implementation usually have inferior performance.

5.4 Reliability

Jack Y.B. Lee

- Uncorrectable Bit-Errors
  - Causes
    - Incorrectly written data
    - Magnetic media aging
  - Common Error Rates
    - Uncorrectable bit error: one in 10^{14} bits read
  - Example
    - Reconstruction of a failed disk in a 100GB disk array.
      - Assumed 200 million sectors are to be read from the normal disks and each read is independent.
      - A BER of 10^{14} implies one 512 byte sector cannot be read for every 24 billion sectors read.
    - Pr{All $2\text{x}10^8$ sectors ok}= $\left(1-\frac{1}{2.4\times10^{10}}\right)^{2\times10^8}=99.2\%$

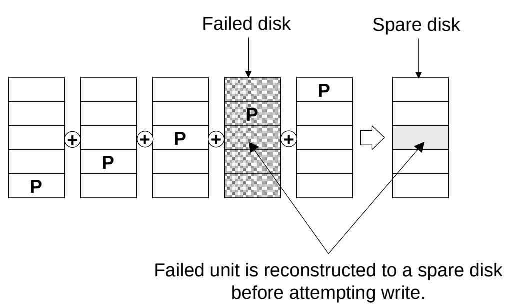
5.5 Implementation Considerations

Jack Y.B. Lee

- Regenerating Parity After a System Crash
  - Problem
    - After a system crash, some parity blocks may become inconsistent due to interrupted writes.
    - All the parity will have to be regenerated unless the inconsistent parity units can be identified.
  - Solution
    - Hardware RAID
      - Log state of parity (consistent/inconsistent) into stable storage (e.g. NVRAM);
      - Mark parity unit as inconsistent before write;
      - Regenerate all inconsistent parity units after crash.

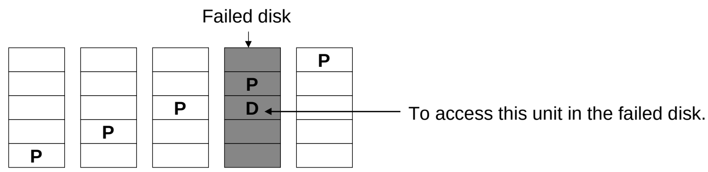
- Regenerating Parity After a System Crash
  - Solution
    - Software RAID
      - Serious performance degradation will be resulted if the disk is used as stable storage to log the parity states.
        - Mark parity as inconsistent before write;
        - Write parity;
        - Mark parity as consistent after write.
      - Delayed Updating
        - Do not remark a parity immediately after write;
        - Put it in a pool;
        - If the parity is to be updated later again, then remarking is not needed.
        - Larger pool size -> better “hit rate” but more parity units to be generated after crash.

- Operating with a Failed Disk
  - Problem
    - Potential data loss in case of system crashes because parity information can be lost during writes.
  - Solution 1: Demand Reconstruction

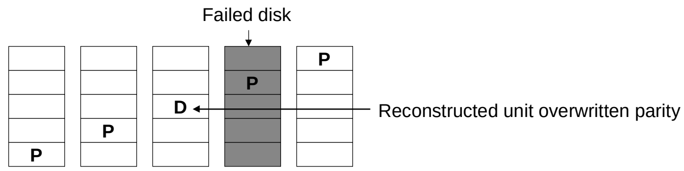


5.5 Implementation Considerations

- Operating with a Failed Disk
  - Solution 2: Parity Sparing
    - After a disk failure:



    - Data unit is reconstructed to overwrite the parity unit:

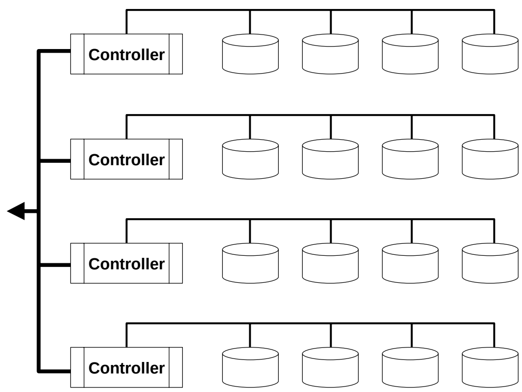


5.5 Implementation Considerations

- Operating with a Failed Disk
  - Solution 2: Parity Sparing
    - After relocation, a system crash only affects the data unit being written.
    - When the failed disk is repaired/replaced, the relocated data unit is simply copied to the new disk and remarked as no longer relocated.
    - Extra meta state information is required to keep the list of relocated data units.

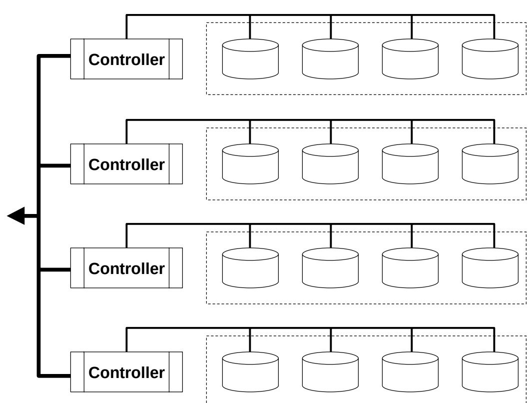
Jack Y.B. Lee

- Orthogonal RAID
  - How to arrange the parity groups?



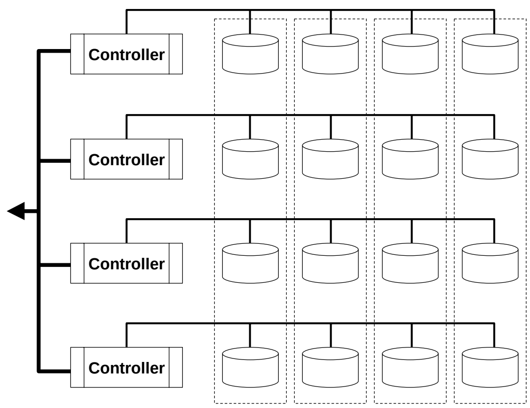
Jack Y.B. Lee

- Orthogonal RAID
  - Option 1: G=4, Per Controller



5.5 Implementation Considerations

- Orthogonal RAID
  - Option 2: *G*=4, One Disk Per Controller

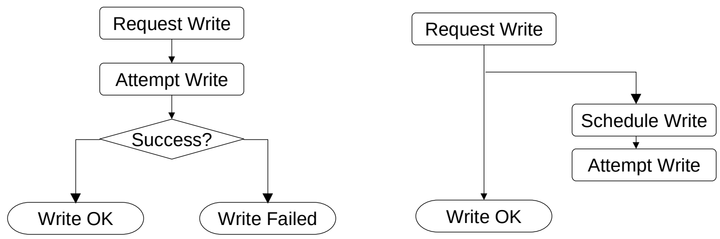


5.6 Improving Small Write Perf. for RAID 5

- RAID Level 5
  - Comparatively good performance in all areas except small write:
    - Read-Modify-Write (4 I/Os)
      - Read old data unit and associated parity unit; (2 reads)
      - Compute new parity;
      - Write new data unit and parity unit. (2 writes)
    - General Performance Degradation
      - Response time approximately doubled; (why not 4x?)
      - Throughput reduced by a factor of 4.
    - Application Implications
      - Unsuitable for applications generating lots of small writes (e.g. transaction processing system).
      - Usually mirroring (RAID-1) is preferred for TPS.

5.6 Improving Small Write Perf. for RAID 5

- Method 1: Buffering and Caching
  - Write Buffering



Synchronous Write *Asynchronous Write*

5.6 Improving Small Write Perf. for RAID 5

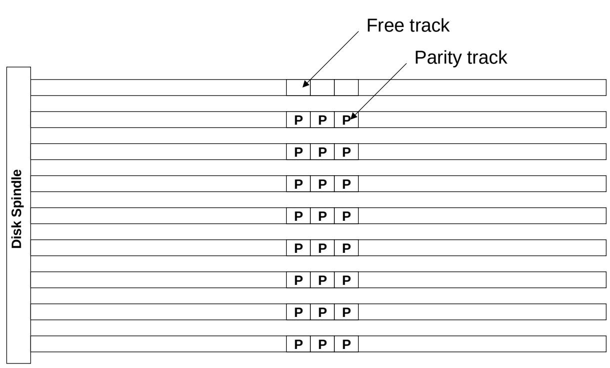
- Method 1: Buffering and Caching
  - Write Buffering
    - Advantages
      - Improved response time (report success immediately to application);
      - Potential saving in case the unit is overwritten before commit;
      - Potential saving in grouping sequential writes in one I/O;
      - More efficient disk scheduling as more queued requests are available to the disk scheduler (e.g. SCAN).
    - Problems
      - Potential data loss in case of system crash unless non-volatile memory is used for buffering the writes;
      - Requires additional hardware support (e.g. NVRAM);
      - Little improvement in throughput;
      - Does not work under heavy load (why?).

5.6 Improving Small Write Perf. for RAID 5

- Method 1: Buffering and Caching
  - Caching
    - Read Caching
      - If the old data unit is in the disk cache, then it can be used to compute the new parity, reducing from 4 to 3 I/Os.
      - Very common in TPS applications where old value is read into the application and rewritten back after modification.
    - Write Caching
      - Caching recently written parity can eliminate the need to read old parity if any of the stripe units belonging to the same stripe is modified again.
      - If successful, this can further reduce from 3 to 2 I/Os.

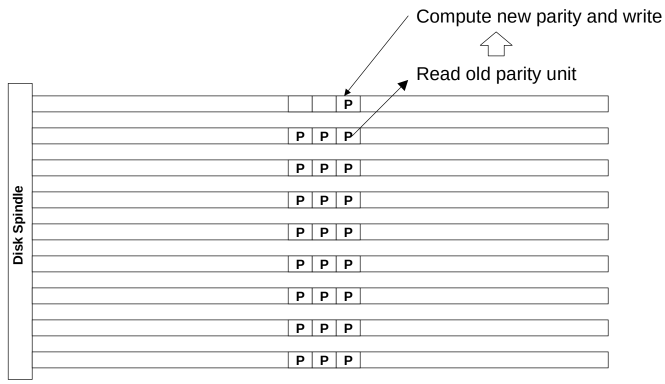
5.6 Improving Small Write Perf. for RAID 5

- Method 2: Floating Parity
  - Clusters parity units into cylinders and leave one free track per such parity cylinder.



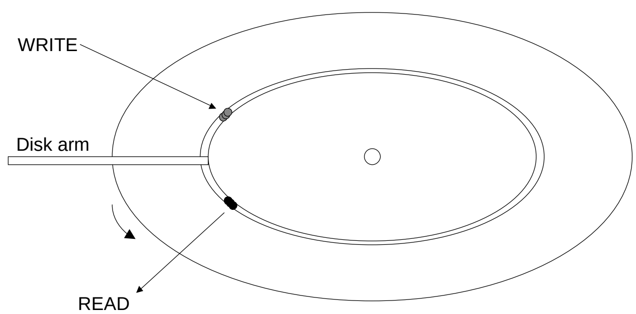
5.6 Improving Small Write Perf. for RAID 5
Jack Y.B. Lee

- Method 2: Floating Parity
  - Read-Modify-Write:



5.6 Improving Small Write Perf. for RAID 5
Jack Y.B. Lee

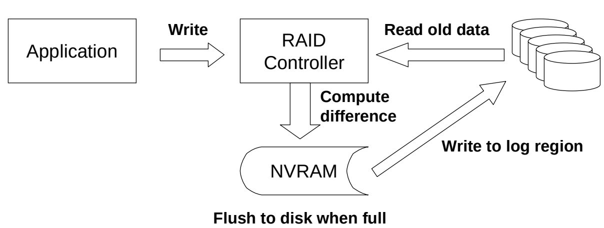
- Method 2: Floating Parity
  - Performance Gain:



- No seeking is needed in parity write;
- Rotational latency is also reduced.

- Method 2: Floating Parity
  - Performance Gain:
    - Example: 16 tracks per cyclinder
      - Around 65% of the time, the block next to the parity block is a free block;
      - Average number blocks to skip to get to the nearest unallocated block is between 0.7~0.8;
      - Hence only an additional ms is needed for the parity write.
    - Tradeoffs:
      - A directory for the locations of free blocks and parity blocks must be maintained in memory.
        - Around 1MB for a RAID with 4~10 500MB disks.
      - Exact geometry of the disk drives is needed to schedule the disk head to an unallocated block.
        - Most likely to be implemented in the RAID controller.

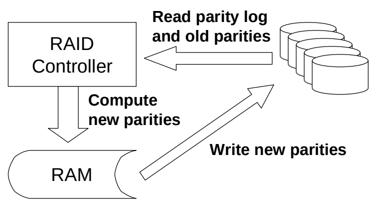
- Method 3: Parity Logging
  - Principle
    - Delays the read of old parity and the write of the new parity.
  - First Part - Delayed Parity Update



5.6 Improving Small Write Perf. for RAID 5

Jack Y.B. Lee

- Method 3: Parity Logging
  - Second Part - Batch Update

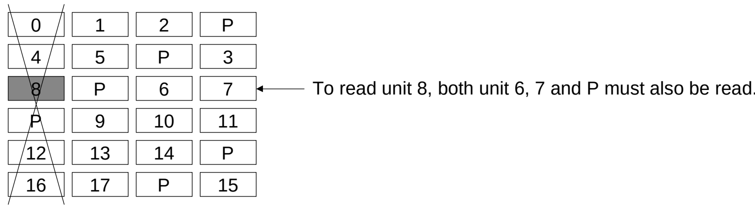


  - Performance Gain
    - Larger transaction size (via batching) reduces disk overhead substantially;
    - Reduces overhead from 4 disk accesses to a little more than 2 disk accesses (comparable to mirroring).

5.7 Declustered Parity

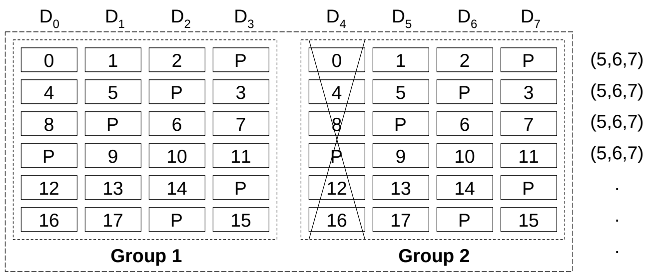
Jack Y.B. Lee

- Performance Degradation After a Disk Failure
  - Small Reads



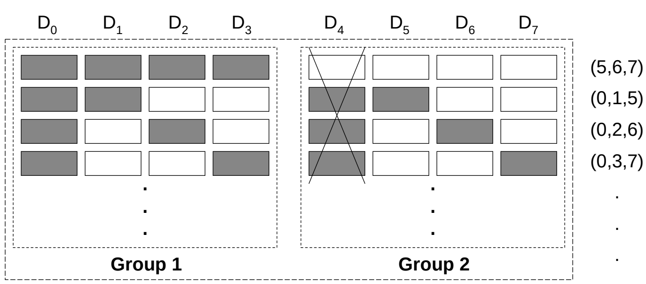
  - In the worst-case, disk load is increased by 100% after a disk failure.

- Performance Degradation After a Disk Failure
  - Small Reads in Large RAID with multiple parity groups:
    - 8 disks with two groups (*G*=4):



- Group 1 is not affected but reading unit 6 still imposes 100% overhead on group 2.
- Disastrous for applications such as video server.

- Performance Degradation After a Disk Failure
  - Small Reads in Large RAID with multiple parity groups:
    - 8 disks, two groups (*G*=4), with declustering:



- Overhead in reading unit 6 is spread across all disks in the system.

5.7 Declustered Parity

Jack Y.B. Lee

- Performance Degradation After a Disk Failure
  - Design of the Block-Placement Policy
    - How to design the placement of blocks from multiple parity groups so that the extra load distribution is uniform across all disks in the system?
    - Method 1: Enumeration
      - Repeat all possible mappings;
      - Given *N* disks and parity group size *G*,

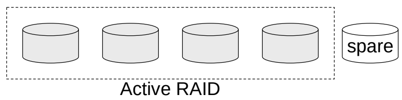
        number of mappings = $\binom{N}{G}$
      - Example: *N*=8, *G*=4 => 70 mappings.
    - Method 2: Theory of Balanced Incomplete Block Designs
      - See M.Hall, *Combinatorial Theory (2nd Ed.)*, Wiley-Interscience, 1996.

5.7 Declustered Parity

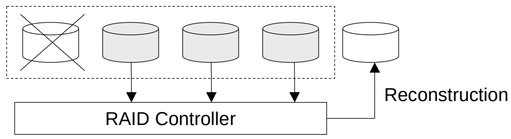
Jack Y.B. Lee

- Tradeoffs
  - Less reliable than standard RAID in double-disk failure
    - Why?
  - The complex parity group mappings could disrupt the sequential placement of large data objects across the disks.

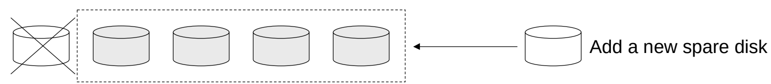
- Online spare disks



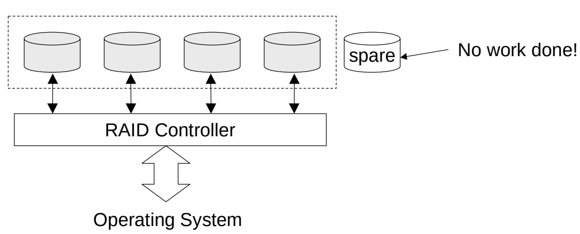
- Reduce window-of-vulnerability after a disk failure



- After reconstruction



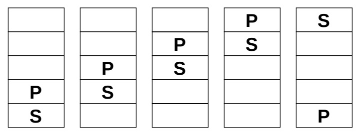
- Motivation
  - Normal Operation



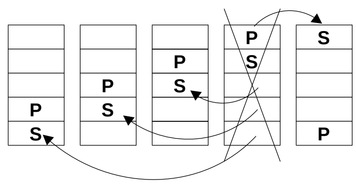
  - Shortcomings
    - The spare disk sits idle without sharing the load of the system;
    - Without active I/O, failure in the spare disk could occur unnoticed.

5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - No dedicated spare disk

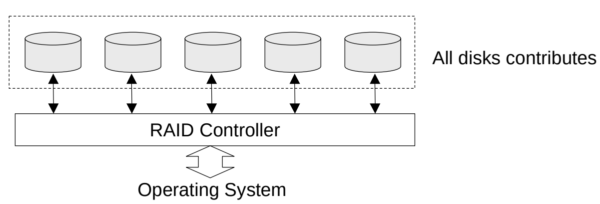


  - Failed units are reconstructed to spare units

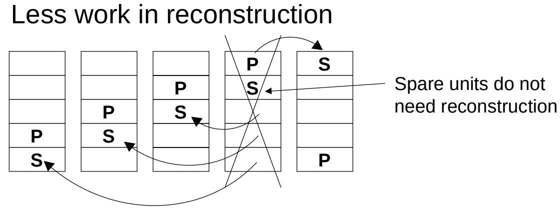


5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - Advantages
    - Improving performance during normal operation

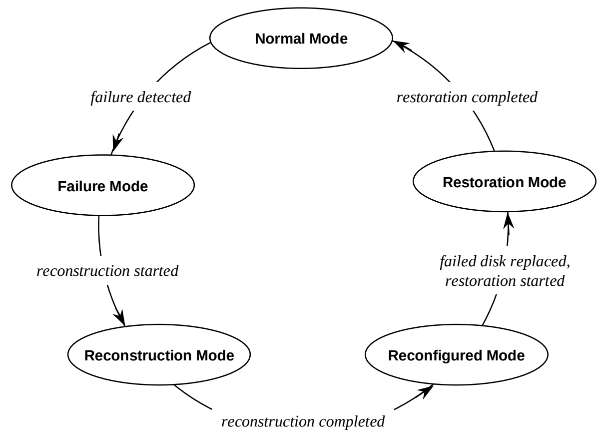


    - Less work in reconstruction



5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - State-transition for a RAID with sparing



5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - Data organization before failure

RAID-5 w/ distributed Sparing

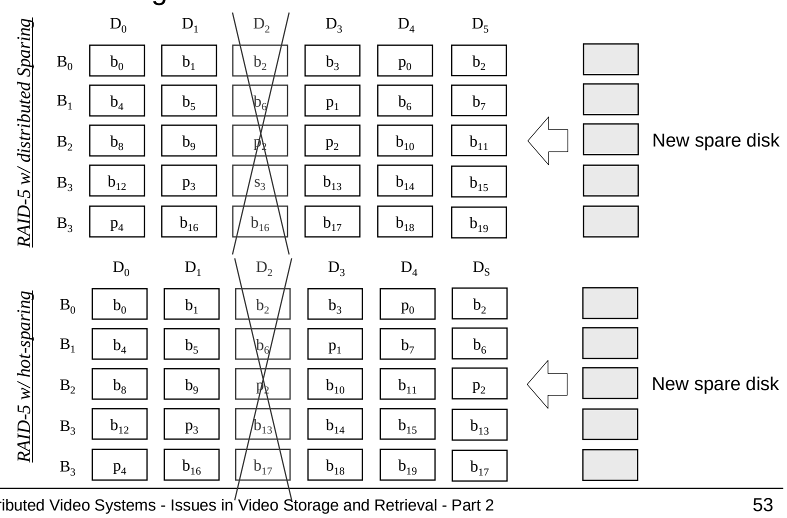
	D_0	D_1	D_2	D_3	D_4	D_5
B_0	b_0	b_1	b_2	b_3	p_0	s_0
B_1	b_4	b_5	b_6	p_1	s_1	b_7
B_2	b_8	b_9	p_2	s_2	b_{10}	b_{11}
B_3	b_{12}	p_3	s_3	b_{13}	b_{14}	b_{15}
B_3	p_4	s_4	b_{16}	b_{17}	b_{18}	b_{19}

RAID-5 w/ hot-sparing

	D_0	D_1	D_2	D_3	D_4	D_S
B_0	b_0	b_1	b_2	b_3	p_0	s_0
B_1	b_4	b_5	b_6	p_1	b_7	s_1
B_2	b_8	b_9	p_2	b_{10}	b_{11}	s_2
B_3	b_{12}	p_3	b_{13}	b_{14}	b_{15}	s_3
B_3	p_4	b_{16}	b_{17}	b_{18}	b_{19}	s_4

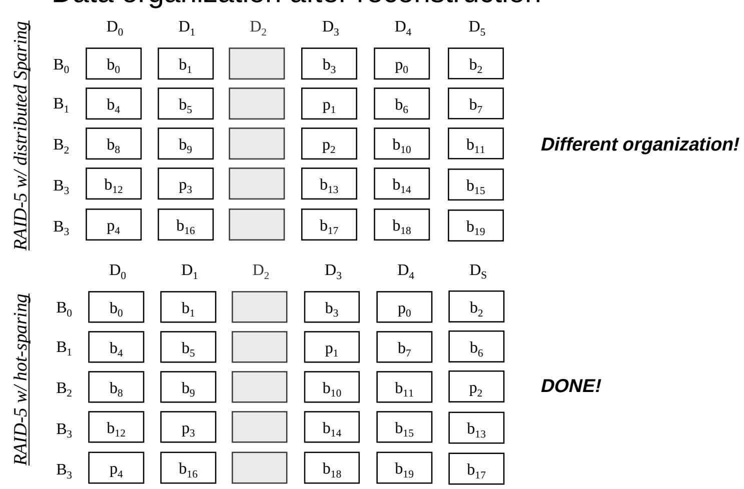
5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - Data organization after reconstruction



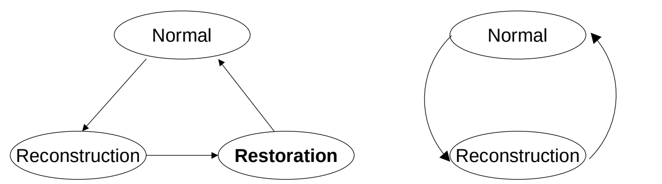
5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - Data organization after reconstruction



5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - Tradeoffs
    - One more step in full system recovery



Distributed Sparing   *Standard Spare Disk (Hot Sparing)*

– A restoration phase is necessary to copy all reconstructed data to a new disk to replace the failed disk.

5.8 Exploiting On-Line Spare Disks

- Distributed Sparing
  - Tradeoffs
    - Potential performance degradation before restoration
      - Data originally in a single disk now distributes across multiple disks after reconstruction;
      - There may be implications for applications doing lots of large, sequential reads and writes (e.g. video server);
    - Performance gain in normal operation may not be usable
      - Video server requires performance guarantees
      - Usually worst-case assumption is used
      - The extra capacity in normal operation cannot be used, otherwise the service of some users will be disrupted when a disk fails.

5.8 Exploiting On-Line Spare Disks

- Parity Sparing
  - Principle
    - Store extra parity information in spare units

			P0	**P1**
		P0	**P1**	
	P0	**P1**		
P0	**P1**			
P1				**P0**

  - Potential Benefits
    - Improving small read/write operations after a disk failure;
    - Creating two parity groups to improve reliability;
    - Implement P+Q redundancy.

References

- The materials in this chapter are based on:

  - P.M.Chen, *et al*.,
    "RAID: High-Performance, Reliable Secondary Storage,"
    ACM Computing Surveys.
  - J. Chandy, *et al*.,
    "Failure Evaluation of Disk Arrsy Organizations,"
    Proc. International Conference on Distributed Computing Systems, May 1993.
  - D.A.Patterson, *et al*.,
    "A Case for Redundant Array of Inexpensive Disks (RAID),"
    Proc. International Conference on Management of Data (SIGMOD), June 1998.