

# Reliable Computing I

## Lecture 8: Redundant Disk Arrays

Instructor: Mehdi Tahoori

INSTITUTE OF COMPUTER ENGINEERING (ITEC) – CHAIR FOR DEPENDABLE NANO COMPUTING (CDNC)

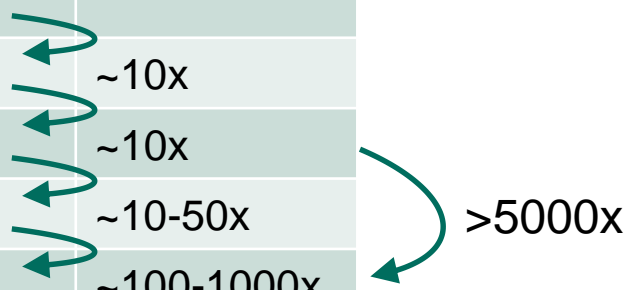


# Challenges and Solutions

## ■ Problem

- Large gap between CPU, memory and disk access times
- Lack of rapid performance improvements in disk technology relates to the mechanical nature of the disk

Memory	Access time	Slower by
Processor registers + L1	<1ns	
L2/L3 Cache memory	10-100ns	~10x
Main memory	100-1000ns	~10x
Solid-state disk	10-50 $\mu$ s	~10-50x
Magnetic disk	5-50ms	~100-1000x



## ■ Solutions:

- Decouple CPU performance and disk performance: caching
- Increase storage device parallelism: disk arrays (RAID)

RAID=Redundant Array of Independent/Inexpensive Disks

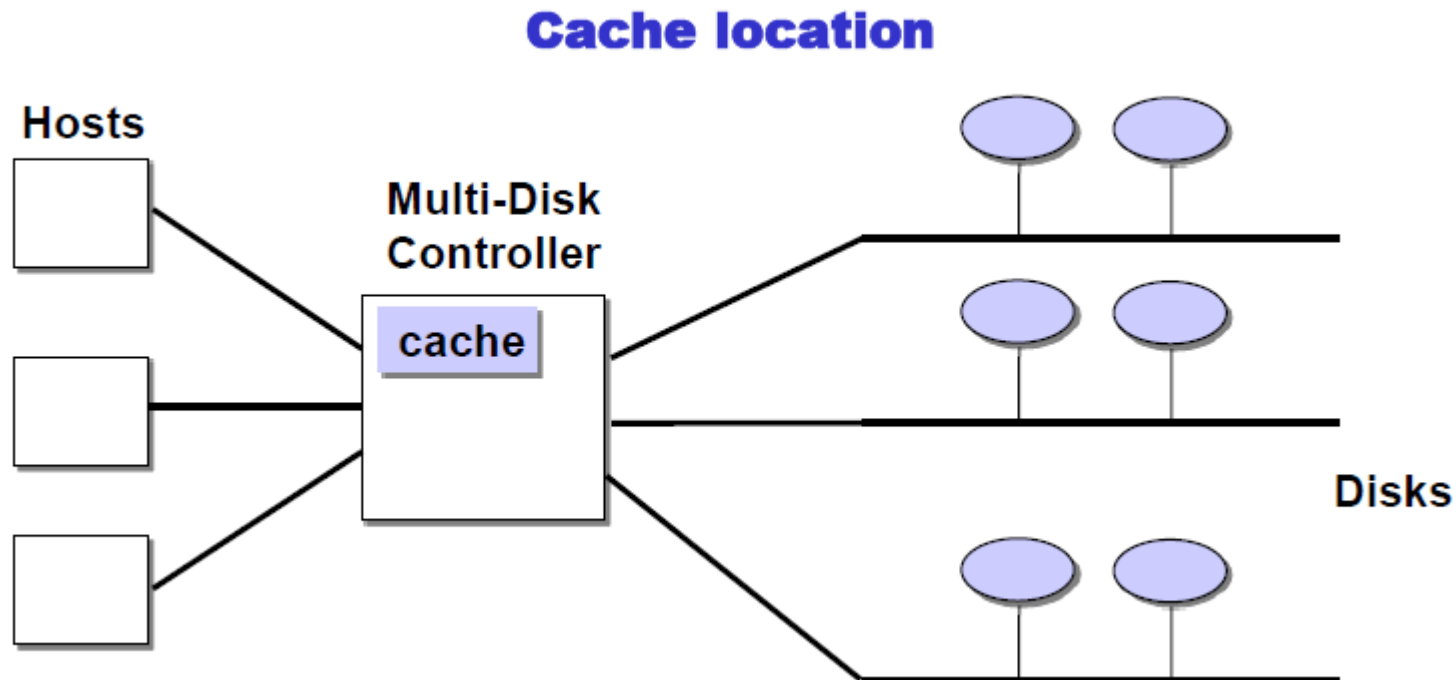
# Cache Performance and Design Considerations

- **Cache:** a fast memory between the hosts and the disks where data is temporarily stored
- Recently accessed data is saved in the cache to improve read performance
- Write to cache and give control to the application before writing to disk
  - Nonvolatile memory to protect data against power failures
  - Ensure data consistency between cache and disk
- Cache performance measured by its miss rate
- Cache size
  - increasing the cache beyond its optimal size has diminishing returns
    - the miss rate decreases with increased cache size but stabilizes after a certain point
    - manufacturers typically offer caches between 0.1% and 0.3% of disk size

# Cache Performance and Design Considerations

- Access behavior of the applications
  - *temporal locality*: a referenced data block tends to be referenced again in the near future
  - *spatial locality*: if a data block is referenced, then nearby data blocks will also soon be accessed
- Replacement algorithms
  - Random replacement (RR) – easy to implement but poor performance
  - Least Recently Used (LRU) – the most popular; exploits the temporal locality
  - Least Frequently Used (FRU) – based on frequency of access count

- Read-ahead strategies: (prefetching)
  - exploits spatial locality by anticipating future requests to data and bringing the data to the cache



# RAID (Redundant Array of Inexpensive Disks)

- Disk Array: separate disks grouped into one logical disk
- **Data striping** for improving performance
  - data is distributed transparently over multiple disks to make them appear as a single fast, large disk
  - parallelism
    - independent requests can be serviced in parallel by separate disks
  - stripe unit: bit, byte, sector, track
- Redundancy for improving reliability
  - a large number of disks lowers the overall reliability of the disk array
    - $N$  disks have  $1/N$  the reliability of a single disk (independent failures)
    - A 600 disk array with 300,000 hours MTTF for each disk will experience one failure every three weeks
- Data replication or parity encoding to tolerate disk failures

# RAID-1: Mirrored Disks

- Mirroring (shadowing) is the simplest redundancy scheme
- Frequently used in database systems where availability and transaction rate are more important than storage efficiency
- All data are duplicated:
  - a complete backup is available when a disk fails
- Disks are grouped into mirror pairs: one copy of each data block stored on each disk in the pair
- High availability at the expense of a high storage overhead
  - #redundancy disks = #data disks
- Tolerates up to  $N/2$  disk failures

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
D0	D0	D1	D1	D2	D2
D3	D3	D4	D4	D5	D5
D6	D6	D7	D7	D8	D8
D9	D9	D10	D10	D11	D11

# RAID-2: Hamming-coded

- Data striped in bits
- **N** data disks and **G** redundancy disks storing a Hamming error correcting code computed over the data stored in each stripe
- $G \sim \log(N+G)$ : increase of storage efficiency as N increases
  - a single redundancy disk is sufficient to detect a single disk failure
  - but more disks are required to identify, which one has failed and to perform error correction
- Only one request can be serviced at a time:
  - each request (read/write) accesses all disks

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4, 5, 6
<b>d0</b>	<b>d1</b>	<b>d2</b>	<b>d3</b>	h0-3
<b>d4</b>	<b>d5</b>	<b>d6</b>	<b>d7</b>	h4-7
<b>d8</b>	<b>d9</b>	<b>d10</b>	<b>d11</b>	h8-11
<b>d12</b>	<b>d13</b>	<b>d14</b>	<b>d15</b>	h12-15

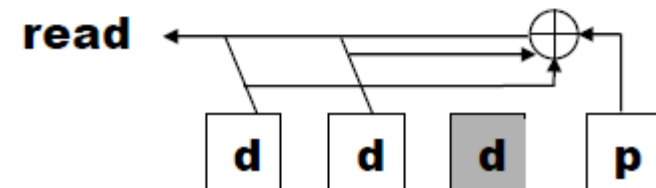
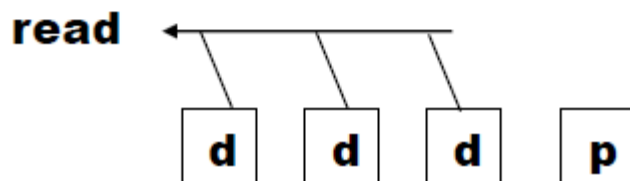
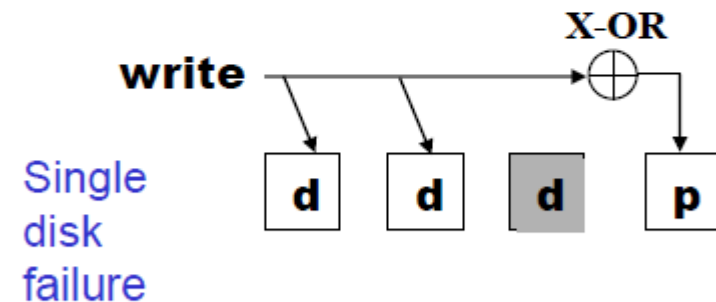
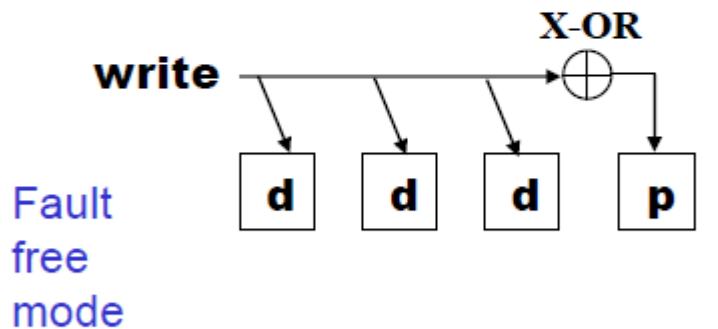


# RAID-3: Bit-Interleaved

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
<b>d0</b>	<b>d1</b>	<b>d2</b>	<b>d3</b>	<b>d4</b>	<i>p(0-4)</i>
<b>d5</b>	<b>d6</b>	<b>d7</b>	<b>d8</b>	<b>d9</b>	<i>p(5-9)</i>
<b>d10</b>	<b>d11</b>	<b>d12</b>	<b>d13</b>	<b>d14</b>	<i>p(10-14)</i>
<b>d15</b>	<b>d16</b>	<b>d17</b>	<b>d18</b>	<b>d19</b>	<i>p(15-19)</i>

*d* - byte  
*p(x-y)* - parity computed over *dx* to *dy*

- Errors detected using parity
- One parity disk needed
- One request can be serviced at a time



# RAID-4: Block Interleaved

- Similar to RAID-3 except that data is striped in blocks instead of bits or bytes
- Striping unit is large
  - small reads access a single disk
- Several concurrent requests can be serviced in parallel
- The parity disk can become a bottleneck: parity is updated for each write

Disk 0   Disk 1   Disk 2   Disk 3   Disk 4   Disk 5

<b>D0</b>	<b>D1</b>	<b>D2</b>	<b>D3</b>	<b>D4</b>	P0-4
<b>D5</b>	<b>D6</b>	<b>D7</b>	<b>D8</b>	<b>D9</b>	P5-9
<b>D10</b>	<b>D11</b>	<b>D12</b>	<b>D13</b>	<b>D14</b>	P10-14
<b>D15</b>	<b>D16</b>	<b>D17</b>	<b>D18</b>	<b>D19</b>	P15-19

**D**        - *data block*  
**P<sub>x-y</sub>**   - *parity computed over D<sub>x</sub> to D<sub>y</sub>*

# RAID-5: Block Interleaved Distributed Parity

- Parity is distributed among all the disks to avoid the parity bottleneck
- Several possible parity distribution strategies
  - e.g., left-asymmetric distribution strategy
- Tolerates one disk failure

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
<b>D0</b>	<b>D1</b>	<b>D2</b>	<b>D3</b>	P0-3
<b>D4</b>	<b>D5</b>	<b>D6</b>	P4-7	<b>D7</b>
<b>D8</b>	<b>D9</b>	P8-11	<b>D10</b>	<b>D11</b>
<b>D12</b>	P12-15	<b>D13</b>	<b>D14</b>	<b>D15</b>
P16-19	<b>D16</b>	<b>D17</b>	<b>D18</b>	<b>D19</b>

# RAID-6: P+Q Redundancy

- P = parity                      Q = Reed-Solomon Code
- Tolerates the failure of up to two disks
- Higher availability but lower write performance compared to RAID-5
- Two redundancy disks (parity + Reed-Solomon code)

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
<b>D0</b>	<b>D1</b>	<b>D2</b>	<b>D3</b>	P0-3	Q0-3
<b>D6</b>	<b>D7</b>	P4-7	Q4-7	<b>D4</b>	<b>D5</b>
P8-11	Q8-11	<b>D8</b>	<b>D9</b>	<b>D10</b>	<b>D11</b>
<b>D12</b>	<b>D13</b>	<b>D14</b>	<b>D15</b>	P12-15	Q12-15
<b>D18</b>	<b>D19</b>	P16-19	Q16-19	<b>D16</b>	<b>D17</b>

# RAID Levels Recap

## ■ RAID 2-4

- Striped in bits/bytes/blocks
- Parity on separate parity disk(s)

## ■ RAID 5-6

- Parity is distributed across disks
- RAID5: Tolerate one disk failure
- RAID6: Tolerate two disk failures