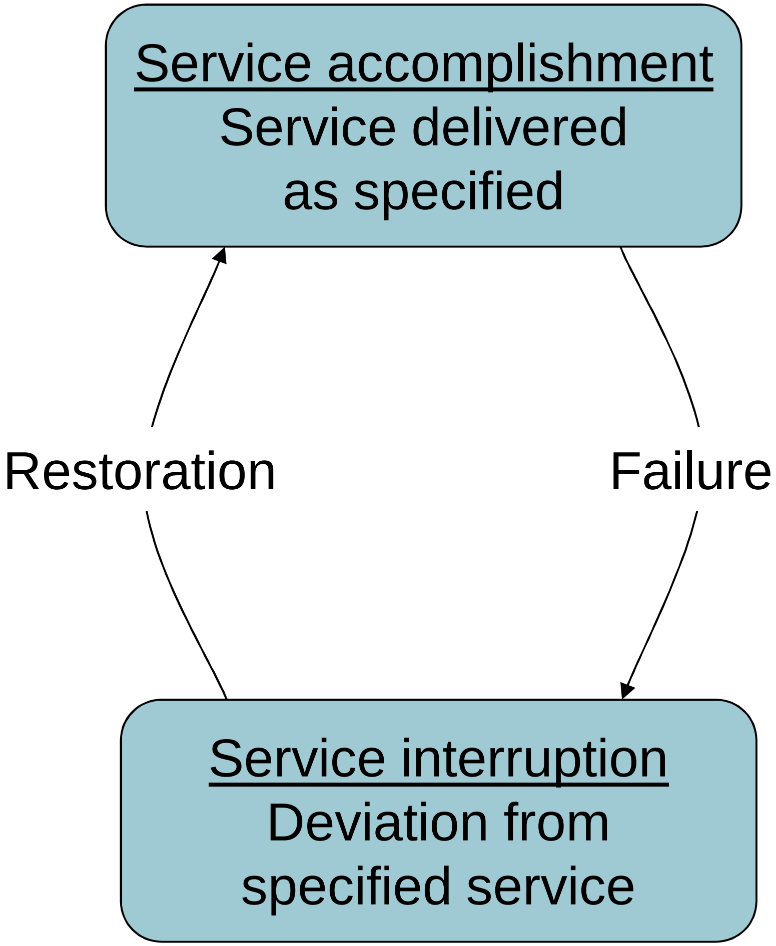
Chapter 6

RAID

I/O System Characteristics

- Dependability is important
  - Particularly for storage devices
- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)
  - Desktops & embedded systems
    - Mainly interested in response time & diversity of devices
  - Servers
    - Mainly interested in throughput & expandability of devices

Dependability



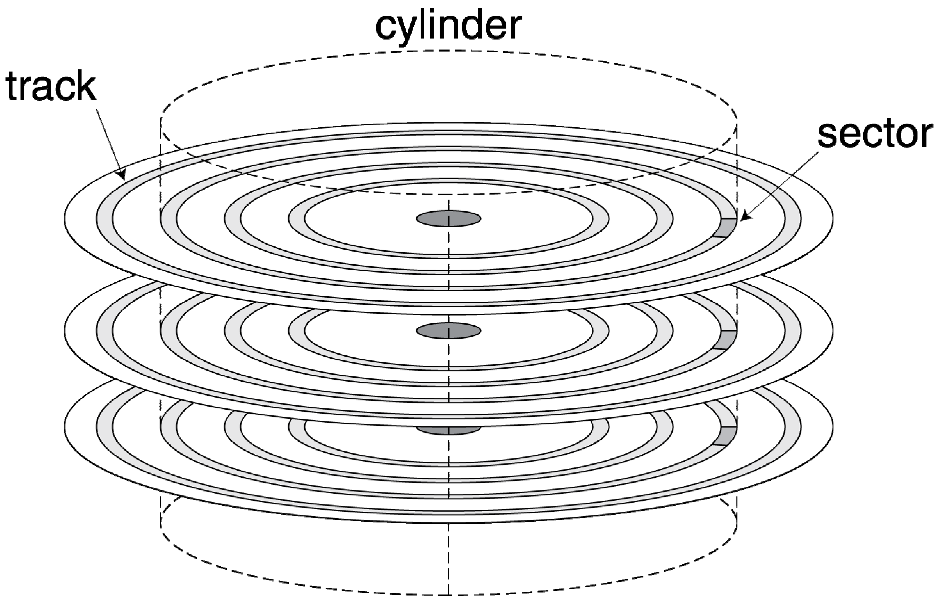
- Fault: failure of a component
  - May or may not lead to system failure

Dependability Measures

- Reliability: mean time to failure (MTTF)
- Service interruption: mean time to repair (MTTR)
- Mean time between failures
  - MTBF = MTTF + MTTR
- Availability = MTTF / (MTTF + MTTR)
- Improving Availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis and repair

Disk Storage

- Nonvolatile, rotating magnetic storage



I/O vs. CPU Performance

- Amdahl's Law
  - Don't neglect I/O performance as parallelism increases compute performance
- Example
  - Benchmark takes 90s CPU time, 10s I/O time
  - Double the number of CPUs/2 years
    - I/O unchanged

Year	CPU time	I/O time	Elapsed time	% I/O time
now	90s	10s	100s	10%
+2	45s	10s	55s	18%
+4	23s	10s	33s	31%
+6	11s	10s	21s	47%

RAID

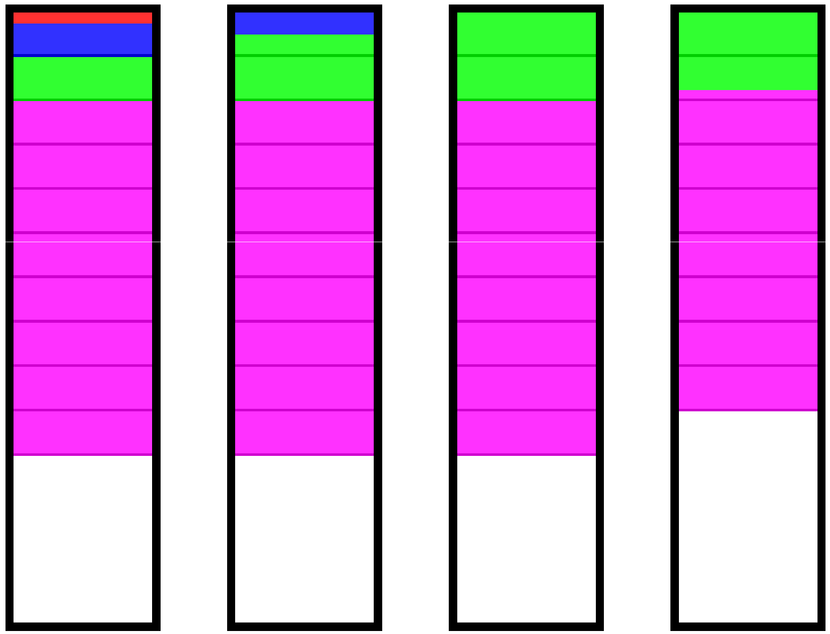
- Redundant Array of Inexpensive (Independent) Disks
  - Use multiple smaller disks (c.f. one large disk)
  - Parallelism improves performance
  - Plus extra disk(s) for redundant data storage
- Provides fault tolerant storage system
  - Especially if failed disks can be "hot swapped"
- RAID 0
  - No redundancy ("AID"?)
    - Just stripe data over multiple disks
  - But it does improve performance

Data Striping

- **Data striping** is the technique of segmenting logically sequential data, such as a file, in a way that
  - accesses of sequential segments are made to different physical storage devices.
- Striping is useful when
  - a processing device requests access to data more quickly than a storage device can provide access.

Data Striping (2)

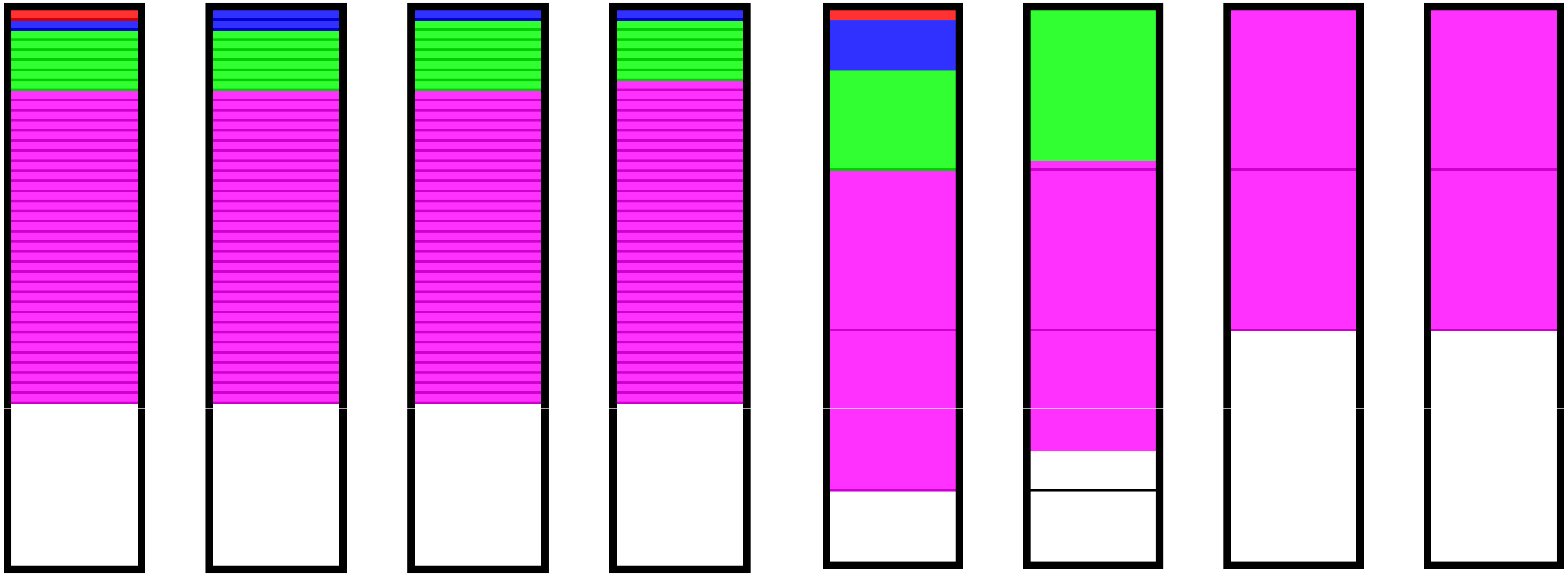
- files of different sizes are distributed between the drives on a four-disk, 16 kiB stripe size RAID 0 array
  - The red file is 4 kiB, the blue is 20 kiB, the green is 100 kiB, and the magenta is 500 kiB.



Data Striping (3)

- By performing segment accesses on multiple devices, multiple segments can be accessed concurrently.
- This provides more data access throughput, which avoids causing the processor to idly wait for data accesses.
- Striping is used across disk drives in RAID storage, network interfaces in Grid-oriented Storage, and RAM in some systems.

Data Striping (4)



- Left: four-disk RAID 0 array with stripe size of 4 kiB;
- Right: same array with same data, using a 64 kiB stripe size

Pros and Cons of Varying Stripe Size

- **Decreasing Stripe Size:** files are broken into smaller pieces, thus increasing the number of drives
  - increases transfer performance, but
  - decreases positioning performance.
- **Increasing Stripe Size:** Fewer drives are required to store files of a given size,
  - transfer performance decreases
  - positioning performance improves

RAID 1 & 2

- RAID 1: Mirroring
  - N + N disks, replicate data
    - Write data to both data disk and mirror disk
    - On disk failure, read from mirror
- RAID 2: Error correcting code (ECC)
  - N + E disks (e.g., 10 + 4)
  - Split data at bit level across N disks
  - Generate E-bit ECC
  - Too complex, not used in practice

Parity Bit

- Parity bits are used as the simplest form of error detecting code.
- A **parity bit** is a bit that is added to ensure that the number of bits with the value one in a set of bits is even or odd.
- An even parity bit will be set to "1" if the number of 1s + 1 is even, and an odd parity bit will be set to "1" if the number of 1s +1 is odd.

Examples of Parity Bit

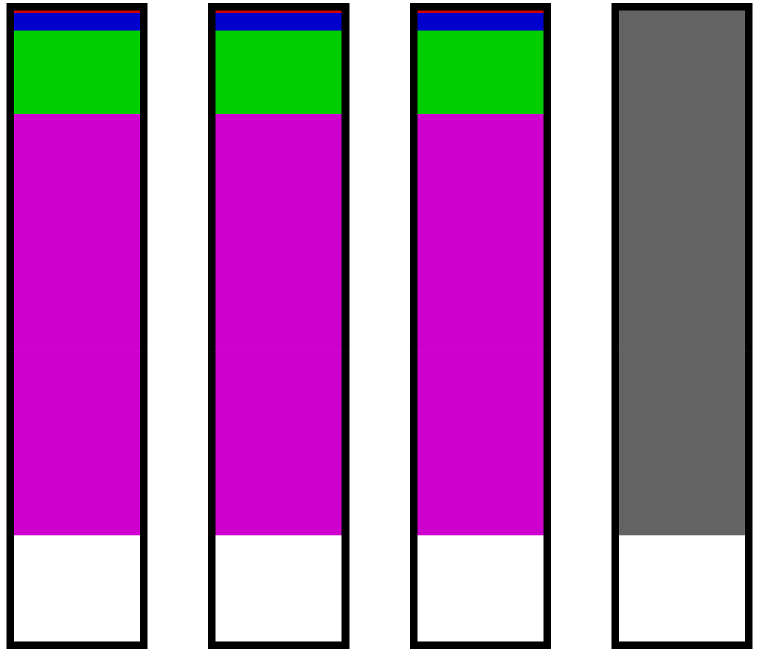
7 bits of data (count of 1 bits)	8 bits including parity	
	even	odd
0000000 (0)	**0**0000000 (0)	**1**0000000 (1)
1010001 (3)	**1**1010001 (4)	**0**1010001 (3)
1101001 (4)	**0**1101001 (4)	**1**1101001 (5)
1111111 (7)	**1**1111111 (8)	**0**1111111 (7)

RAID 3: Bit-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at byte level
  - Redundant disk stores parity
  - Read access
    - Read all disks
  - Write access
    - Generate new parity and update all disks
  - On failure
    - Use parity to reconstruct missing data
- Not widely used

RAID 3

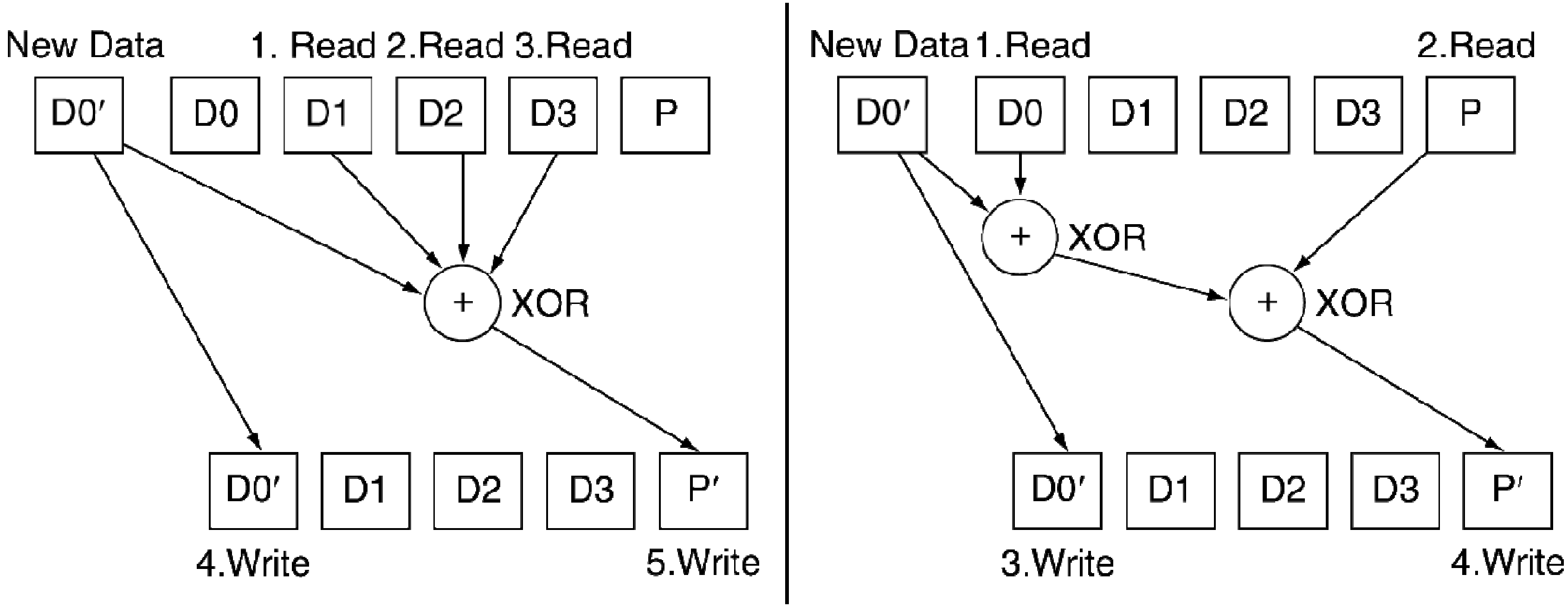
- In RAID 3, data is striped across multiple disks at a byte level;
  - Max stripe size is typically under 1024 bytes.
  - Since the blocks are so tiny, the boundaries between stripes can't be seen.



RAID 4: Block-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at block level
  - Redundant disk stores parity for a group of blocks
  - Read access
    - Read only the disk holding the required block
  - Write access
    - Just read disk containing modified block, and parity disk
    - Calculate new parity, update data disk and parity disk
  - On failure
    - Use parity to reconstruct missing data
- Not widely used

RAID 3 vs RAID 4



Using Parity Bit in RAID

- Parity data is used by some RAID levels to achieve redundancy.
- If a drive in the array fails, remaining data on the other drives can be combined with the parity data (using the Boolean XOR function) to reconstruct the missing data.

Using Parity Bit in RAID (2)

- The principle behind parity is simple:
  - take "N" pieces of data, and from them, compute an extra piece of data.
  - Take the "N+1" pieces of data and store them on "N+1" drives.
  - If you lose any *one* of the "N+1" pieces of data, you can recreate it from the "N" that remain, regardless of which piece is lost.

Using Parity Bit in RAID (3)

- Parity protection is used with striping, and the "N" pieces of data are typically the blocks or bytes distributed across the drives in the array.
- The parity information can either be stored on a separate, dedicated drive, or be mixed with the data across all the drives in the array.

Why Use XOR

- "XOR" is that it is a logical operation that if performed twice in a row, "undoes itself".
- If you calculate "A XOR B" and then take that result and do another "XOR B" on it, you get back A, the value you started with.
  - That is to say, "A XOR B XOR B = A".
- This property is exploited for parity calculation under RAID.

Using Parity Bit in RAID (4)

- For example, suppose two drives in a three-drive RAID 5 array contained the following data:
- Drive 1: **01101101**
  Drive 2: **11010100**
- To calculate parity data for the two drives, an XOR is performed on their data:
- **01101101** XOR **11010100 = 10111001**
- The resulting parity data, **10111001**, is then stored on Drive 3.

Using Parity Bit in RAID (5)

- Should any of the three drives fail, the contents of the failed drive can be reconstructed on a replacement drive by subjecting the data from the remaining drives to the same XOR operation.
- If Drive 2 were to fail, its data could be rebuilt using the XOR results of the contents of the two remaining drives, Drive 1 and Drive 3:

Using Parity Bit in RAID (6)

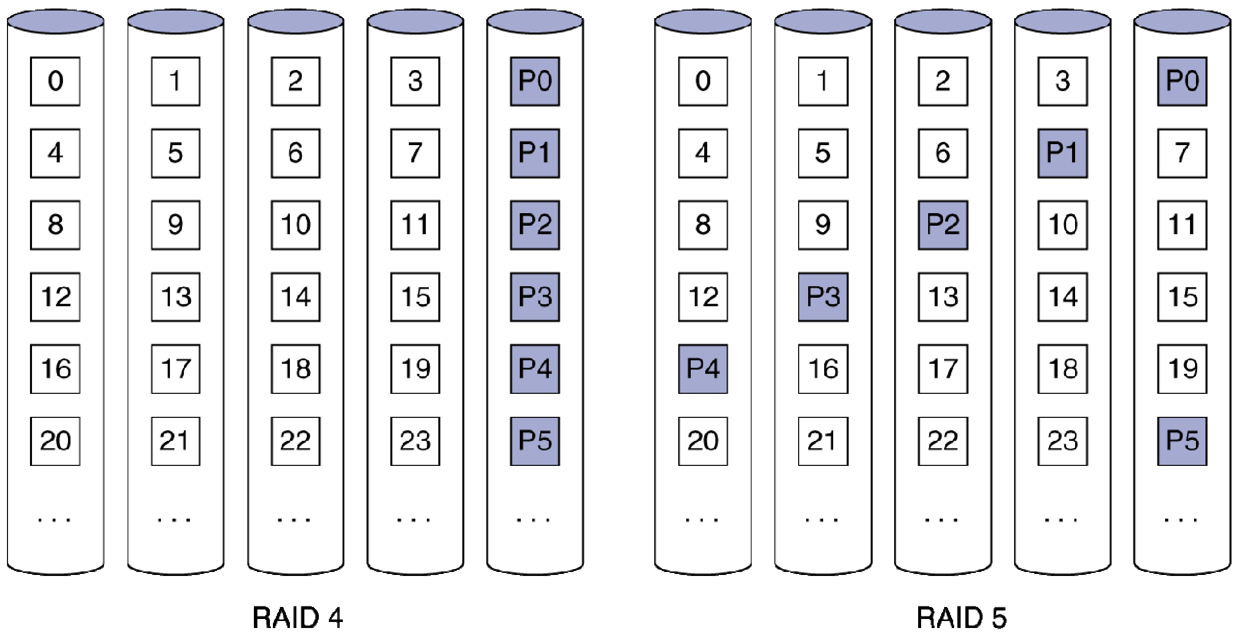
- Drive 1: **01101101**
  Drive 3: **10111001**
- as follows:
- **10111001** XOR **01101101** = **11010100**
- The result of that XOR calculation yields Drive 2's contents. **11010100** is then stored on Drive 2, fully repairing the array.

Using Parity Bit in RAID (7)

- This same XOR concept applies similarly to larger arrays, using any number of disks.
- In the case of a RAID 3 array of 12 drives, 11 drives participate in the XOR calculation shown above and yield a value that is then stored on the dedicated parity drive.

RAID 5: Distributed Parity

- N + 1 disks
  - Like RAID 4, but parity blocks distributed across disks
    - Avoids parity disk being a bottleneck
- Widely used



RAID 6: P + Q Redundancy

- N + 2 disks
  - Like RAID 5, but two lots of parity
  - Greater fault tolerance through more redundancy
- Multiple RAID
  - More advanced systems give similar fault tolerance with better performance

Mirroring vs. Parity with Striping

- Mirroring vs. Parity with striping
  - parity protects data against any single drive in the array failing without requiring the 50% "waste" of mirroring;
  - only one of the "N+1" drives contains redundancy information. (The overhead of parity is equal to (100/N)% where N is the total number of drives in the array.)
  - Parity with striping also allows to take the performance advantages of striping

Mirroring vs. Parity with Striping

- The chief disadvantages of striping with parity relate to complexity:
  - all those parity bytes have to be computed--millions of them per second!--and that takes computing power.

Read and Write Operations

- Mirroring (RAID 1)
- Striping without Parity (RAID 0)
- Striping with Parity (RAID 3-6)
  - Sequential write
  - Random write

- Refer to: http://www.pcguide.com/ref/hdd/perf/raid/concepts/perf_ReadWrite.htm

Use the following Website

- http://www.pcguide.com/ref/hdd/perf/raid/index.htm