Lecture Outline:

- Disk Scheduling
- NAND Flash Memory
- RAID: Redundant Array of Inexpensive/Independent Disks

In this lecture, we describe disk-scheduling algorithms that schedule the order of disk I/Os to improve the system performance. We then examine the characteristics of FLASH memory and we introduce the Flash Translation Layer specification, which allows a Linear Flash device to look like a FAT disk. We conclude with a description of Redundant Array of Inexpensive/Independent Disks (RAID).

1 Disk Scheduling

An operating system should use the hardware efficiently in terms of access time (i.e. seek time and rotation latency) and disk bandwidth. To achieve fast access time and large disk bandwidth the operating system may reschedule the servicing of disk I/O requests in a good order. Whenever a process requests an I/O to or from a disk, if the disk drive and the controller are available can be serviced immediately. Otherwise the request is placed in a queue. This queue may have a lot of pending requests. The operating system can reschedule the servicing of these requests to achieve a better performance.

One of the most famous and simple disk scheduling algorithm is the Elevator Algorithm. It serves the requests in the order they are in the disk. Assume a sequence of requests for different locations in the disk (Figure 1).

![Figure 1: Request order](Figure 1: Request order)

The operating system can reschedule them (i.e. change their order) to improve efficiency, by serving
them in the order they are in the disk (Figure 2 and Figure 3). In this simple example we ignore the rotational latency.

![Figure 2: Serve order (Blue numbers)](image)

In some multithreaded architectures split-phase transaction mechanisms have been adopted as an efficient way to overlap computation and communication in distributed memory systems. In our case multiple commands to the device can be requested though a SCSI or a SATA bus as shown in Figure 4. Devices choose in which order they may execute the operations (i.e. schedule the servicing).

![Figure 3: Request and Serve order](image)

In this case the operating system should provide synchronization with the disk drive for safety and the device should apply a disk scheduling (i.e. reschedule the order of operations) which will improve the efficiency of the system.
2 Flash Memory (NAND)

In this section we introduce the basics of NAND flash memory. Flash memory used primarily in memory cards, USB drives, and MP3 players. But as the quest for lower-power, lighter and more robust products continues, nowadays flash memory seems to be an ideal solution for a wider range of applications.

Flash memory stores information in an array of memory cells made from floating-gate transistors connected in series. Each array is grouped into a series of blocks, which are the smallest erasable entities in a NAND Flash device. So there is a mechanism that writes in a block (page write), but does not rewrite until the entire block is been erased (Figure 5). This technology first copy the whole block into a register, erase the block and then reprograms the entire block by using the new data and by retrieving the old data from the register.

![Figure 5: NAND flash memory](image)

Flash memory characteristics:

- Fast page-read access time.
- 1-time page write.
- Block erase

2.1 Limitations

There are two limitations by using Flash memory:

- **Block erasure**: One limitation of flash memory is that although it can be read or programmed in a random access fashion, it must be erased a block at a time.

- **Memory wear**: Another limitation is that flash memory has a finite number of erase-write cycles (Around 100,000 write-erase-cycles, before the wear begins to deteriorate the integrity of the storage).

The goal is to increase the speed and the distribution of the memory wear (wear leveling).
Flash Translation Layer (FTL)

![FTL diagram]

Figure 6: FTL

Flash Translation Layer (FTL) specification, allows a Flash device to look like a FAT disk, but still have effective wear levelling, by writing to a different cell each time. This method is good for certain amount of erase-cycles (Usually around 100,000 cycles). Flash dynamically remaps blocks and use copy on write (CoW) in order to spread write operations between sectors and in this way to spread the wear around.

![CoW and remapping diagram]

Figure 7: CoW and remapping

Note that if we write sequentially, eventually we will need CoW, but we will have worse garbage collection.
We will not give implementation details for this technique.

3 RAID: Redundant Array of I. Disks

Having a large number of disks in a system, one may want to improve the rate at which data can be read or written, if the disks are operated in parallel and improve the reliability of data storage, because redundant information can be stored on multiple disks (i.e. After one disk’s failure, we do not lose the data). Disk-organization techniques called RAID are used to address the performance and reliability issues.

The best performance can be achieved by arranging data sequentially:
+ (OP1, OP2) \( \propto \vert LBA_1 - LBA_2 \vert \)
+ (A, A+1) \( \propto \vert A - A' \vert \), \( A' \neq A \)

Figure 8: Read / Write Consistency
RAID Levels

RAID 0

In RAID Level 0 (also called striping), each segment is written to a different disk, until all drives in the array have been written to. In fact it is a deterministic mapping:

- Chunk = LBA mod size
- Drive = Chunk mod N (N drives)
- Chunk on drive: \( \lfloor \frac{Chunk}{N} \rfloor \)

The I/O performance of a RAID-0 array is significantly better than a single disk. This is true on small I/O requests, as several can be processed simultaneously, and for large requests, as multiple disk drives can become involved in the operation.

This level of RAID is the only one with no redundancy. If one disk in the array fails, data is lost (e.g. If drives 0 and 1 have separated file systems then it will be impossible to recover data after a disk fail).

RAID 1

In RAID Level 1 (also called mirroring), each disk is an exact duplicate of all other disks in the array. When a write is performed, it is sent to all disks in the array. When a read is performed, it is only sent to one disk. This is the least space efficient of the RAID levels.

A RAID-1 array normally contains two disk drives. This gives adequate protection against drive failure. It is more reliable from RAID 0 because to loose data both drives should fail.
RAID-1 arrays with multiple mirrors are often used to improve performance in situations where the data on the disks is being read from multiple programs or threads at the same time. By being able to read from the multiple mirrors at the same time, the data throughput is increased, thus improving performance.

Figure 10: RAID 1

RAID 3

In RAID Level 3 (also called bit-interleaved parity organization), is taking in account the fact that disk controllers can detect whether a sector has been read correctly, so a single parity bit can be used for error correction as well as for detection. Every I/O to the array will access all drives in the array, regardless of the type of access (read/write) or the size of the I/O request. During a write, RAID-3 stores a portion of each block on each data disk. It also computes the parity for the data, and writes it to the parity drive. In some implementations, when the data is read back in, the parity is also read, and compared to a newly computed parity, to ensure that there were no errors.

RAID-3 provides a similar level of reliability to RAID-4 and RAID-5, but offers much greater I/O bandwidth on small requests. In addition, there is no performance impact when writing. Unfortunately, it is not possible to have multiple operations being performed on the array at the same time, due to the fact that all drives are involved in every operation. RAID-3 also has configuration limitations. The number of data drives in a RAID-3 configuration must be a power of two. The most common configurations have four or eight data drives.

RAID 4

RAID Level 4 is defined as blockwise striping with parity. The parity is always written to the same disk drive. This can create a great deal of contention for the parity drive during write operations. For reads, and large writes, RAID-4 performance will be similar to a RAID-0 array containing an
equal number of data disks. For small writes, the performance will decrease considerably. (e.g. for 20 random writes/sec we need 10 operations/sec in data drives and 40 operations/sec for parity drive).

A write request for one block is issued by a program.

1. The RAID software determines which disks contain the data, and parity, and which block they are in.
2. The disk controller reads the data block from disk.
3. The disk controller reads the corresponding parity block from disk.
4. The data block just read is XORed with the parity block just read.
5. The data block to be written is XORed with the parity block.
6. The data block and the updated parity block are both written to disk.

**RAID 5**

RAID Level 5 is defined as blockwise striping with parity. It differs from RAID-4, in that the parity data is not always written to the same disk drive (i.e. rotating the parity, such that operations are split evenly across the drives). RAID-5 has all the performance issues and benefits that RAID-4 has, except as follows:

1. Since there is no dedicated parity drive, there is no single point where contention will be created. This will speed up multiple small writes.
2. Multiple small reads are slightly faster. This is because data resides on all drives in the array. It is possible to get all drives involved in the read operation.