SE227
COMPUTER SYSTEMS ENGINEERING
RECITATION 6: RAID

QINGQING CHENG

1. Levels of RAID

1.1. **RAID 0.** Simplest approach is to alternate the physical addresses on the physical disk. For example, if you have m disks, then you place the virtual memory location of the kth location on the $k \pmod{m}$th disk.

1.2. **RAID 1.** You have two copies of the same disk. This improves redundancy by creating a new disk with the same information. You can improve performance on reads because if you are reading a chunk of data, you can break that chunk up into different start places and read these smaller chunks concurrently.

Alternatively, you can create more than just two copies of the data. This improves the chances that at least one disk has not failed, but improves the chances that any given disk fails.

1.3. **RAID 2.** Using error correcting codes. Let's say you have a 7 bit string 7654321. Three of these bits are check bits, and the other four bits actually store the data values. You compute the check bits and make sure the data bits are correct.

$$p_1 = XOR(d_1, d_3, d_4) \tag{1}$$
$$p_2 = XOR(d_1, d_2, d_4) \tag{2}$$
$$p_3 = XOR(d_1, d_2, d_3) \tag{3}$$

If all of the parity bits p_1, p_2, p_3 pass, then we probably have the correct answer. Each column has at least 2 check bits. For example d_2 occurs in both p_2 and p_3. This code only works if there is at most 1 bit that has been flipped to the wrong value.

1.4. **RAID 3.** Instead of using the more expensive error correcting code, we have any number of data disks and a single check disk which is just the XOR of all the data disks. If the check disk is incorrect, then you know that one of the data disks is incorrect as well.

1.5. **RAID 4.** Parity bits are stored on a single data.

1.6. **RAID 5.** If you have some disks, you alternative responsibility of data and check disks for different addresses. You rotate the information of the parity information across the disk. It is unlikely that all the blocks you choose will have the same check disks so hopefully you can spread the check bits evenly across the disks. This prevents your single check disk from failing and losing your redundancy.

1.7. **Recursive RAID.** You have a bunch of RAID 0 arrays. Then you combine the entire set of RAID 0 arrays using RAID 1. This is called RAID 0+1.

2. Reasons of many different types of RAID

Application has reads and writes to the RAID controller. Then the RAID controller does the actual reads and writes to the actual disk. It makes a bunch of disks look like a single disk. You can think of each disk as a private variable to the RAID controller, and RAID abstracts everything away.

The motivation for RAID:

- Reliability. You want to be able to retain access to your data even when you have failures. Most people who care about reliability are implementing servers.
- Cost. You can buy a lot of cheap disks for a cost much lower than a single fault tolerant disk.
- Performance. Increase disk throughput. For example, making games better.

2.1. **Reliability.** This is improved via redundancy.

2.2. **Performance.** One basic mechanism to improve performance: interleaving. Take data that you would put on just one disk and put it on multiple disks so you can read and write data concurrently on many disks.

3. Suitable Scenarios for Different Types

3.1. **RAID0.** Have a high demand for performance and a low demand for security.

3.2. **RAID1.** Have a high demand for reliability, repairability and redundancy.

3.3. **RAID2.** Have a high demand for repairability, redundancy and a low demand for performance.

3.4. **RAID3.** Have a high demand for fault-detection, performance and a low demand for fault-correction.

3.5. **RAID4.** Similar to RAID3, expect: Have a high demand for large/grouped quick transfer and a low demand for concurrent read/write.

3.6. **RAID5.** Similar to RAID3, expect: Have a high demand for read performance and a low demand for write performance.

3.7. **Recursive RAID.** Have a high demand for reliability, performance and a low demand for low-cost.

4. A Case For A Highly Reliable System

4.1. **Fallible Components.** Components of this system whose failure could result in a loss of service can be disk bit-flip, disk corruption, package loss during network transfer.

4.2. **Recursive RAID.** For disk bit-flip, we'd better use mirrors to recover in the consideration of security. And use RAID0 to guarantee performance.