© Andrew Quinn

RAID

Andrew Quinn

Learning Objectives:

1. What is the overall structure that an operating system uses to manage persistence?
2. How does an operating system interact with storage devices?
3. What should our mental model of a hard drive be?

Announcements:

1. Have you started working on Project 3?

Before we move up to the file system from the block-level analysis, let us discuss more complex forms of block devices. In particular, we'll discuss RAID—Redundant Arrays of Inexpensive Disks.

With all of their moving parts, hard disks tend to fail pretty frequently. In the 90s, consumers could purchase expensive enterprise-grade disks, that failed infrequently, or inexpensive consumer-grade disks, which failed frequently. RAID ask: can we chain together a few of those inexpensive disks so that they appear to fail less frequently?

It turns out that the answer is yes. And, RAID actually gives a few benefits beyond reliability. By chaining together multiple disks, a single system can often improve the storage capacity of their system. Second, by chaining together multiple disks, a single system can often improve the performance of their system.

RAID typically works with a hardware device called a *RAID controller*. RAID controllers are quite sophisticated devices with roughly the same complexity as an entire server. We won't talk about their hardware internals. By placing RAID behind a hardware device, RAID controllers provide the illusion of a single large disk to the rest of the system.

We'll talk about four separate RAID "models" for how data can be arranged in a RAID device. We'll discuss the capacity, performance, and reliability of the various options:

RAID 0—STRIPPING. RAID 0 "stripes" data across all of the drives in the RAID—there is actually no redundancy! The capacity of a RAID 0 device is $N * B$, where B is the number of blocks per drive and N is the number of drives. Stripping reliability is quite poor: it loses data on a single disk failure! However, stripping does pretty well in terms of performance. If we assume that the drives making up the raid have a random-access throughput of R and a sequential-access throughput of S, then RAID would provide a random throughput of $N * R$ and a sequential throughput of $N * S$. These figures would hold for both reads and writes.

RAID 1—MIRRORING. RAID 1 mirrors data across drives. Typically, this mirroring is done so that each drive in the array has a duplicate. The capacity of a RAID 1 device is thus $N * B/2$, since half of the drives are used for redundancy. A RAID 1 setup can support a single drive failure before losing data. A RAID 1 setup has a bit of a performance problem, though. Each logical disk write results in two physical disk writes, so the bandwidth for

© Andrew Quinn

sequential writes is $N * S/2$ and for random writes is $N * R/2$. Reads are a bit better; depending on how the system schedules blocks to be read, it may be able to get the full bandwidth of all of its disks (i.e., $N * R$ or $N * S$). Note: RAID 1 will not improve read *latency* (even though it seems like it should be possible!) How does RAID 1 effect write latency?

RAID 4—PARITY. RAID 4 uses what is called a parity block to reduce the amount of mirroring that it needs to perform. The idea is to group blocks together and calculate their parity (usually by executing the exclusive-or of the bits in the block). In the event of a failure, you can use parity to recreate any lost blocks. Pretty cool!

The capacity of RAID 4 is $(N - 1) * B$, while its reliability is supporting a single disk failure. Performance is a bit more complex to calculate. Reads are relatively easy: the array can use all of the disks at once and get a read performance of $(N - 1) * R$ or $(N - 1) * S$, depending on the workload. For sequential writes, then the system will *typically* write out all of the blocks in a group at the same time. This can happen in parallel and keep all of the drives busy, so the effective sequential write bandwidth would be $(N-1)*S$. Random writes are a bit more expensive. The Parity block will have to change alongside whatever data block is updated. It turns out that there are multiple ways to determine the new parity block, but either approach requires reading the blocks *before* writing. In either case, the bandwidth is equal to the bandwidth of one random read and one random write to the parity disk, or $R/2$.

RAID 5—ROTATING PARITY. RAID 5 does parity, but rotates the parity blocks across the drives in the system. Its analysis is *almost* identical, except it has much better random write behavior since it uses all of the drives in the system for random writes. Thus, the random write bandwidth is $N/4*R$; the factor of 4 arises because of the 4 operations that the system does for each random write.