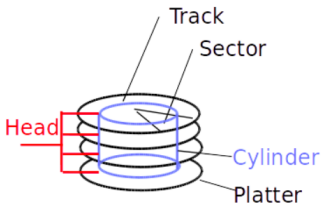
Section 10: Disks, Performance, and Queuing Theory

April 10th, 2020

Contents

1 Vocabulary

- **I/O** In the context of operating systems, input/output (I/O) consists of the processes by which the operating system receives and transmits data to connected devices.
- **Controller** The operating system performs the actual I/O operations by communicating with a device controller, which contains addressable memory and registers for communicating the the CPU, and an interface for communicating with the underlying hardware.
- **Response Time** Response time measures the time between a requested I/O operating and its completion, and is an important metric for determining the performance of an I/O device.
- **Throughput** Another important metric is throughput, which measures the rate at which operations are performed over time.
- **Hard Disk Drive (HDD)** - A storage device that stores data on magnetic disks. Each disk consists of multiple **platters** of data. Each platter includes multiple concentric **tracks** that are further divided into **sectors**. Data is accessed (for reading or writing) one sector at a time. The **head** of the disk can transfer data from a sector when positioned over it.



- **Seek Time** - The time it takes for an HDD to reposition its disk head over the desired track.
- **Rotational Latency** - The time it takes for the desired sector to rotate under the disk head.
- **Transfer Rate** - The rate at which data is transferred under the disk head.
- **Checksum** - A mathematical function which maps a (typically large) input to a fixed size output. Checksums are meant to detect changes to the underlying data and should change if changes occur to the underlying data. Common checksum algorithms include CRC32, MD5, SHA-1, and SHA-256.
- **Replication** - Replication or duplication is a common technique for preserving data in the face of disk failure or corruption. If a disk fails, data can be read from the replica. If a sector is corrupted, it will be detected in the checksum. The data can then be read from another replica.
- **Queuing Theory** Here are some useful symbols: (both the symbols used in lecture and in the book are listed)
  - μ is the average service rate (jobs per second)
  - T_{ser} or S is the average service time, so $T_{ser} = \frac{1}{\mu}$
  - λ is the average arrival rate (jobs per second)
  - U or u or ρ is the utilization (fraction from 0 to 1), so $U = \frac{\lambda}{\mu} = \lambda S$
  - T_q or W is the average queuing time (aka waiting time) which is how much time a task needs to wait before getting serviced (it does not include the time needed to actually perform the task)
  - T_{sys} or R is the response time, and it's equal to $T_q + T_{ser}$ or $W + S$
  - L_q or Q is the average length of the queue, and it's equal to λT_q (this is Little's law)

2 Disks

What are the major components of disk latency? Explain each one.

Queuing time - How long it spends in the OS queue
Controller - How long it takes to send the message to the controller
Seek - How long the disk head has to move
Rotational - How long the disk rotates for
Transfer - The delay of copying the bytes into memory

In class we said that the operating system deals with bad or corrupted sectors. Some disk controllers magically hide failing sectors and re-map to "back-up" locations on disk when a sector fails.

If you had to choose where to lay out these "back-up" sectors on disk - where would you put them? Why?

Should spread them out evenly, so when you replace an arbitrary sector your find one that is close by.

How do you think that the disk controller can check whether a sector has gone bad?

Using a checksum - this can be efficiently checked in hardware during disk access.

Can you think of any drawbacks of hiding errors like this from the operating system?

Excessive sector failures are warning signs that a disk is beginning to fail.

3 I/O Performance

This question will explore the performance consequences of using traditional disks for storage. Assume we have a hard drive with the following specifications:

- An average seek time of 8 ms
- A rotational speed of 7200 revolutions per minute (RPM)
- A controller that can transfer data at a maximum rate of 50 MiB/s

We will ignore the effects of queueing delay for this problem.

1. What is the expected throughput of the hard drive when reading 4 KiB sectors from a random location on disk?

   The time to read the sector can be broken down into three parts: seek delay, rotational delay, and data transfer delay. We are already given the expected seek delay: 8 ms.

   We can assume that, on average, the hard disk must complete 1/2 revolution before the sector we are interested in reading moves under the read/write head.

   Given that the disk makes 7200 revolutions per minute, the time to complete a revolution is 60 sec/7200 Revolution $\approx$ 8.33 ms per revolution.

   The time to complete 1/2 revolution, the expected rotational delay, is $\sim$ 4.17 ms.

   If the controller can transfer 50 MiB per second, it will take:

   $$4 \times 2^{10} \text{ bytes} \times \frac{1 \text{ sec.}}{50 \times 2^{20} \text{ bytes}} \approx 0.00781 \text{ ms}$$

   to transfer 4 KiB of data.

   In total, it takes 8 ms + 4.17 ms + 0.00781 ms $\approx$ 12.18 ms to read the 4 KiB sector, yielding a throughput of 4 KiB/12.18 ms $\approx$ 328.5 KiB/s

2. What is the expected throughput of the hard drive when reading 4 KiB sectors from the same track on disk (i.e., the read/write head is already positioned over the correct track when the operation starts)?

   Now, we can ignore seek delay and only need to account for rotational delay and data transfer delay.

   We already know that the expected rotational delay is 4.17 ms and we know that the expected data transfer delay is 0.00781 ms.

   Therefore, it takes a total of 4.17 ms+0.00781 ms $\approx$ 4.18 ms to read the 4 KiB sector, yielding a throughput of 4 KiB/4.18 ms $\approx$ 957 KiB/s.

3. What is the expected throughput of the hard drive when reading the very next 4 KiB sector (i.e. the read/write head is immediately over the proper track and sector at the start of the operation)?

   Now, we can ignore both rotational and seek delays. The throughput of the hard disk in this case is limited only by the controller, meaning we can take full advantage of its 50 MiB/s transfer rate.

   Note that this is roughly a 156$\times$ improvement over the random read scenario!

4. What are some ways the Unix Fast File System (FFS) was designed to deal with the discrepancy in performance we just saw?

   - Attempt to keep contents of a file contiguous on disk (first-fit block allocation)
   - Break disk into a set of *block groups* — sets of adjacent tracks, each with its own free space bitmap, inodes, and data blocks
   - Keep a file's header information (inode) in same block group as its data blocks
   - Keep files in the same directory in the same block group

4 Queuing Theory

Explain intuitively why response time is nonlinear with utilization. Draw a plot of utilization (x axis) vs response time (y axis) and label the endpoints on the x axis.

Even with high utilization (99%), some of the time (1%), the server is idle, which is a waste. All this wasted time adds up, and in the steady state, the queue becomes very long.
Graph should be linear-ish close to $u = 0$ and grow asymptotically toward ∞ at $u = 1$.

If 50 jobs arrive at a system every second and the average response time for any particular job is 100ms, how many jobs are in the system (either queued or being serviced) on average at a particular moment? Which law describes this relationship?

$50 \times 0.1 = 5$ (5 jobs at any time). This is Little's law.

Is it better to have N queues, each of which is serviced at the rate of 1 job per second, or 1 queue that is serviced at the rate of N jobs per second? Give reasons to justify your answer.

One server that can process N jobs per millisecond is faster. Better response time ($\frac{1}{N}$ sec vs 1 sec) and better utilization (no load-balancing problems), which gives you lower queuing delays on average.

What is the average queueing time for a work queue with 1 server, average arrival rate of λ, average service time S, and squared coefficient of variation of service time $\mathbf{C}$?

$T_q = T_{ser}(\frac{u}{1-u})(\frac{C+1}{2})$ where $u = \lambda S$

What does it mean if $\mathbf{C} = 0$? What does it mean if $\mathbf{C} = 1$?

If $\mathbf{C} = 0$, then your arrival rate is regular and deterministic, which means that tasks arrive at a constant rate.
If $\mathbf{C} = 1$, then your arrival rate can be modeled as a Poisson distribution, and the interval between arrivals can be modeled as a exponential distribution.

5 Tying it all together

Assume that you have a disk with the following parameters:

- 1TB in size
- 6000RPM
- Data transfer rate of 4MB/s (4×10^6 bytes/sec)
- Average seek time of 3ms
- I/O controller with 1ms of controller delay
- Block size of 4000 bytes

What is the average rotational delay?

$\frac{1}{2} \times \frac{60\text{sec/minute}}{6000\text{RPM}} = 5\text{ms}$

What is the average time it takes to read 1 random block? Assume no queuing delay.

$\frac{4{,}000\text{bytes}}{4{,}000{,}000\text{bytes/sec}} = 1\text{ms}$, and $1 + 3 + 5 + 1 = 10\text{ms}$

Will the actual measured average time to read a block from disk (excluding queuing delay) tend to be lower, equal, or higher than this? Why?

It will be lower, because the operating system will use a disk scheduling algorithm to improve locality. This model assumes the disk is always seeking to a random location.

Assume that the average I/O operations per second demanded is 50 IOPS. Assume a squared coefficient of variation of $\mathbf{C} = 1.5$. What is the average queuing time and the average queue length?

$$T_q = T_{ser}(\frac{u}{1-u})(\frac{\mathbf{C}+1}{2})$$
$$u = \lambda T_{ser}$$
$$u = 50\text{IOPS} \times 0.01\text{sec}$$
$$u = 0.5$$
$$T_q = 10\text{ms}(\frac{0.5}{1-0.5})(\frac{1.5+1}{2})$$
$$T_q = 12.5\text{ms}$$
$$L_q = T_q\lambda$$
$$L_q = 0.0125 \times 50$$
$$L_q = 0.625 \text{ operations}$$