104062203 陳涵宇

1. Problem Set

1. Explain why SSTF scheduling tends to favor middle cylinders over the innermost and outermost cylinders.

Because the average distance to any other track from the middle cylinders is the smallest, the disk tends to move away from the edge.

2. Requests are not usually uniformly distributed. For example, we can expect a cylinder containing the file-system metadata to be accessed more frequently than a cylinder containing only files. Suppose you know that 50 percent of the requests are for a small, fixed number of cylinders.

a. Would any of the scheduling algorithms discussed in this chapter be particularly good for this case? Explain your answer.

b. Propose a disk-scheduling algorithm that gives even better performance by taking advantage of this “hot spot” on the disk.

(a) SSTF may be good for this case, since it will have higher probability getting high frequency requests, the disk will also have high probability to tent to the target cylinder.

(b) By question 1, we can place the hot data close to the middle cylinder. SSTF also needs to be modified to prevent starvation, for example, aging. If the disk does not get any request for a particular time, then it can move to the region storing hot data, since the next request will be the hot data with higher probability.

3. Consider the following I/O scenarios on a single-user PC:

a. A mouse used with a graphical user interface

b. A tape drive on a multitasking operating system (with no device preallocation available)

c. A disk drive containing user files

d. A graphics card with direct bus connection, accessible through memory-mapped I/O

For each of these scenarios, would you design the operating system to use buffering, spooling, caching, or a combination? Would you use polled I/O or interrupt-driven I/O? Give reasons for your choices.

(a) This may need buffering to record mouse action during other higher priority operation is executed. It is better to use interrupt-driven I/O.

(b) This may need a combination of three ways. Buffering can be used to manage throughput between the tape drive and different sources/destinations.

Spooling can be used when multiple users try to read from or write to same data. Caching can be used to hold copies of data in the tape for faster access. Interrupt-driven I/O is great in this case.

(c) This may need a combination of buffering and caching. Buffering can be used to hold data when transiting from user space to disk. Caching can be used to hold data in disks for better performance. Interrupt-driven I/O is great for the slow transfer device such as disks.

(d) This may need buffering to control multiple accessing. Double buffering can be used to hold next image to show while displaying current image. In this case, both polling and interrupt-driven is useful for detecting inputs and I/O completions.

4. Typically, at the completion of a device I/O, a single interrupt is raised and appropriately handled by the host processor. In certain settings, however, the code that is to be executed at the completion of the I/O can be broken into two separate pieces. The first piece executes immediately after the I/O completes and schedules a second interrupt for the remaining piece of code to be executed at a later time. What is the purpose of using this strategy in the design of interrupt handlers?

The purpose is to ensure that the most critical part of the interrupt handling code is performed first, and the less critical part is performed later.