UNIVERSITY OF WISCONSIN
Computer Sciences Department

Operating Systems Qualifying Exam **Fall 2017**

Instructions: There are *six* questions on this exam; answer *all six* questions.

Question 1. Storing Objects

You are designing a new object-based distributed storage system that supports gets and puts of entire objects; assume you initially know nothing else about the workload.

A. Your first design decision is whether you will build on top of an existing update-in-place local file system, such as Linux's ext4, or you will build directly on the raw block storage device. What are some of the pros and cons of each approach? In your answer, make sure you touch on the range of functionality that a local file system tends to provide.
B. Assume you decided to build your object-based distributed storage on top of the existing local file system; you plan to use a single file per machine to store the meta-data that your object store needs. Your next decision is to determine how your layer will allocate objects to local files. Assume your two design choices are to allocate all objects to a single file or to allocate each object to its own file. If you know nothing about the workload, what are the pros and cons of each approach?
C. Assume you now know something about the workload, such as the distribution of object sizes, the mix of reads and writes, and mix of random or sequential accesses. Can you describe a workload that will perform better with the object per file approach? Why will it perform better? Can you describe a workload that will perform better with the single file approach? Why will it perform better?

Question 2. Scheduling

Proportional share scheduling has been applied to many devices, include CPUs, disks and networks.

A. What is proportional share scheduling and when is it most useful and when is it not useful?
B. Given a multilevel feedback queue (MLFQ) scheduler, under what conditions (scheduler parameters or workloads) can you achieve proportional share scheduling between processes?
C. Lottery scheduling uses randomness to achieve proportional share. Given a queue `ready_queue` of processes, each with a number of tickets, sketch the code to implement a lottery.
D. Most operating systems do not use lotteries for proportional share scheduling. What are the (i) benefits and (ii) drawbacks and benefits of using lotteries?

Question 3. Consistent Hashing

Chord is built atop "consistent hashing", a method for distributing objects over a collection of machines. In this question, we examine consistent hashing and its benefits and costs.

A. Before discussing consistent hashing, let's understand more traditional approaches. When using simple static hashing schemes, what problems does one face in a cluster environment?
B. Describe how consistent hashing works, including the notion of a successor and the role of the finger table. How does it overcome the problems of static hashing approaches?
C. Now let's focus a bit more on the finger table. What is its role, and why is it important?
D. What happens if information in the finger table, for whatever reason, is incorrect? Can the system detect a problem? Can it recover?

Question 4. The Fast File System

The Fast File System (FFS) paper is a classic example of how to build a file system for a traditional hard drive. In this question, we examine how many of its assumptions and techniques hold up in the modern world.

A. Describe FFS policies for file and directory allocation. What are its main goals? How does it achieve them?
B. Now, let's change the medium underneath from a hard drive to a flash-based SSD. Assume the SSD is ideal, in that it works as follows: each block is accessible in constant time. Which aspects of FFS policies are needed for this SSD? Which are not?
C. Now assume the SSD has an internal (in memory) mapping table, which, given the logical address of a block, contains the block's physical (on device) address. This mapping table cannot entirely fit into device memory, and thus inactive portions are swapped out of device memory to accommodate popular translations. Given an SSD with such a mapping table, are FFS policies useful? Discuss.

Question 5. Byzantine Agreement

One solution to the Byzantine Generals problem makes the following assumptions about the underlying system:

A. Every message that is sent is delivered correctly.
B. The receiver of a message can reliably determine the identity of the process from which it was (most recently) sent.
C. The receiver of a signed message can reliably determine whether the message has been modified since it was signed.
D. Anyone can verify the authenticity of a signature.

Discuss whether each of these assumptions is reasonable in light of the way distributed systems really work. If an assumption is reasonable, describe how a system would support such a feature. Where applicable, provide examples from systems that you know. If an assumption is unreasonable, explain why the feature is difficult to provide.

Question 6. WSClock

The WSClock algorithm (in part) is an approximation to Working Set.

A. In what ways is the approximation more efficient to implement than pure Working Set?
B. Under what conditions will WSClock provide a good approximation of Working Set and what conditions will it provide a poor one?
