Operating Systems Qualifying Exam
Fall 2018

University of Wisconsin-Madison
Department of Computer Sciences

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

Question 1: Reconstructing Lost Data in RAIDs

In this question, you will explore how to recover lost data in a **RAID-5** array containing N disks.

a. Assume that a disk has completely failed in a RAID-5 array and a new replacement disk has been added. Your task is to write the code to reconstruct the lost information onto the new replacement disk. Assuming you can directly access each of the remaining N-1 working disks with a local block number to read or write, what basic steps must you perform to correctly reconstruct the lost information on the new disk?

b. Does your reconstruction algorithm make any assumptions about the correctness of the information stored on the other N-1 disks? Are there changes you could make to the RAID-5 storage system to make those assumptions more likely to be true or to reduce your dependence on these assumptions?

c. Now that you have a working reconstruction algorithm, your next task is to optimize its performance. First, how could you optimize the time taken to perform reconstruction if you know you are operating on an array of Hard Disk Drives (HDDs)? Second, how might knowing characteristics of the workload impact how you perform reconstruction?

Question 2: Copy-on-Write

Copy-on-write is a way for processes to share pages (or segments) that are logically distinct. When one process sends some data to a second process with copy semantics, nothing needs to be copied immediately, or perhaps ever.

a. Explain how standard memory-management hardware can be used to implement copy-on-write cheaply.

b. Describe the extra costs of copy-on-write, and explain when it is better to just copy the data directly.

c. Describe how copy-on-write can be used to improve system performance for: (i) message passing, (ii) forking a new process, and (iii) file I/O operations.

Question 3: The Google File System

One of the keys to the success of the Google File System is the knowledge of the expected workload that the designers had in mind when building it.

a. Describe features of the typical Google workload, as discussed in the paper. Which types of file accesses were common? Which were rare? How was this different than typical distributed file systems? What other workload aspects were important?
b. Now describe the aspects of the GFS design that took advantage of these workload characteristics, focusing on the most important ones. Which design elements would not have worked well under more common file system workloads?
c. Assume you could change one aspect of the Google workload, with the perverse goal of making GFS perform poorly. What workload aspect would you change, and why would it harm GFS performance?

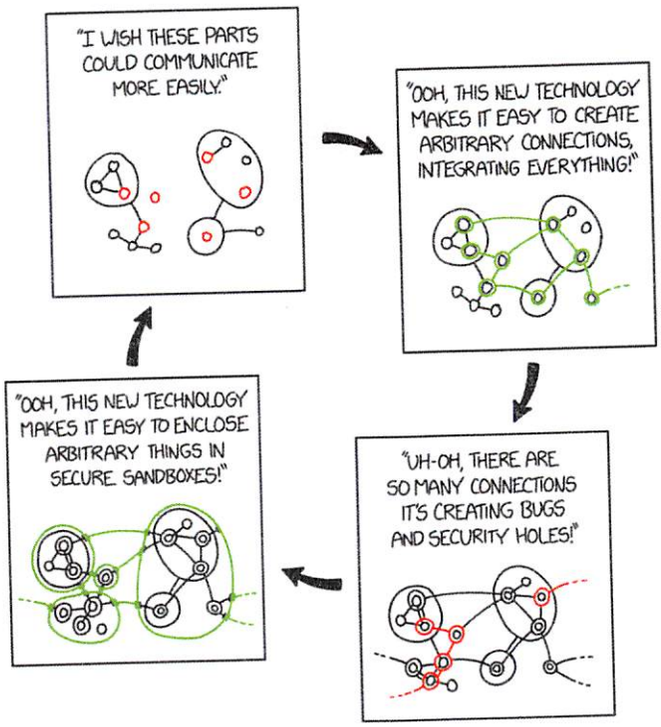
Question 4: Resilience in MapReduce

a. How does MapReduce behave when there are worker failures? What about master failures?
b. What are stragglers and why do they effect the performance of MapReduce jobs?
c. What are some of the causes of stragglers and how does MapReduce overcome them? What are some examples of stragglers that *cannot* be overcome using techniques described in the original MapReduce paper?
d. If you were to run MapReduce on a cluster that had no persistent disks how would the fault tolerance mechanism change for worker failures?

Question 5: Optimizations for LRPC

Lightweight RPC optimized a communication stack to minimize the latency of RPCs between processes on the same system. LRPC takes advantage of communication patterns to create fast paths through RPC stacks.

a. Explain two communication patterns that LRPC leverages.
b. Explain how LRPC leverages these two patterns to reduce RPC latency.
c. LRPC allocates a pool of E-stacks for threads to use when executing RPC server code. Why are these needed, and why can't they be shared?
d. Some modern networks provide the ability to directly read or write data on a remote machine (like a cross-machine memcpy()), and to optionally trigger an interrupt at the receiver. The RPC client and server can communicate before RPC calls to establish memory regions for the other to read and write. Which LRPC optimizations **will not work** if you use this communication mechanism rather than having both client and server run on the same OS kernel, and why?



(source:**https://xkcd.com/2044/**)

Question 6: Kernel structure

Most operating systems in use (IOS, MacOS, Linux, Windows) have a large, monolithic kernel. The XKCD comic above discusses the tension between isolation and flexibility, which has long been an issue in operating system design. **For each system,** explain how it addresses the tension between wanting the flexibility of letting different pieces of code communicate (flexibility) and also wanting reliability and security (isolation).

a. Nucleus
b. Unix
c. Hydra
d. Pilot
e. Exokernel
f. In your opinion, which OS structure best achieves this balance, and why?