CMU 18-746/15-746 **Storage Systems** **3 Mar 2015**

Spring 2015 **Exam 1**

Name: ______________________________

Instructions

There are three (3) questions on the exam. You may find questions that could have several answers and require an explanation or a justification. As we've said, many answers in storage systems are "It depends!". In these cases, we are more interested in your justification, so make sure you're clear. Good luck!

If you have several calculations leading to a single answer, please place a box around your answer.

Problem 1 : Short answer. [48 points]

(a) Indirect blocks allow a file system to have very large files without requiring that all data for a file be in contiguous LBNs or requiring the inode to include pointers to every discontiguous file block. But, most file systems use inodes that have some block pointers with differing levels of indirection (e.g., some direct, some single indirect, some double indirect, etc.). Why not have all block pointers in the inode be for the same level of indirection, to simplify the implementation? Explain your answer.

It would reduce small file performance (and space efficiency), limit the maximum file size, or both. If all pointers are deeply indirect, then even a one-byte file would use multiple indirect blocks, resulting in space overhead, write amplification, and multiple read I/Os for a cache miss. If all pointers are shallowly indirect (or direct), then either the largest file is not very big, or the inode is much bigger with lots and lots of pointers, leading to space waste for small files.

(b) Imagine that you work for a large Internet services company that has 100,000 disks in its data center. Assume that disks are organized into 10,000 10-disk arrays, using data striping and striped parity (i.e., RAID 5). To save money, the head of IT wants to use cheap disks that have an MTBF of 20 years, instead of the more reliable disks with an MTBF of 100 years. Harry argues that doing so will not reduce reliability if a small collection of extra disks is purchased and kept as spares, so as to reduce the repair time for a failed disk from 10 days (includes delivery of replacement disk) to 2 days. Do you agree with Harry? Explain your answer.

No, Harry is incorrect. MTTDL goes down linearly with MTTR but goes down quadratically with MTBF, because MTBF is part of both parts of the MTTDL equation with repair: the time until the first failure and the probability of a second failure before rebuild is completed.

(c) Janice likes to buy used storage devices, to save money. Before using a new storage device, Janice always reads every LBN in order to see if what types of data had been stored there previously. His most recent purchase is a traditional mechanical disk. What integrity benefit could be realized from Janice's curiosity? Explain your answer.

Detection of grown defects, such that the disk will remap their LBNs to spare sectors on the next write. (This action is essentially disk scrubbing.)

(d) You are a consultant for a company planning to replace the mechanical disks on its heavily utilized mail server with TLC SSDs. (The CTO has heard that his competitors are using first generation SLC SSDs and wants to leapfrog the competition's technology.) You are asked to highlight the biggest risk in making this change. What is this biggest risk and what do you advise to mitigate that risk?

TLC error tolerance is much lower, and the number of useful writes per cell is much lower. If TLC is used in place of SLC then the number of writes that can be done before wear out is lower by three orders of magnitude. Since a heavily used mail server is writing most the time, this solution is likely to see SSD wear out failures much more commonly than the competitors solution. Mitigate by using MLC SSDs (2 bits per cell) or SLC SSDs.

(e) What is the maximum number of file system blocks that could be modified in completing a single write() system call of 8KB, assuming ext2fs with a 8KB file system block size? Explain your answer.

15 + one or more copies of the superblock (plus group descriptor blocks, but lets set those aside ;)). The 8KB written could be part of the last file system block linked via the inode's double indirect block and the first block linked via the triple indirect block. All of the indirect blocks may need to be allocated (if the offset was set by a seek() call to more the file pointer to this location. So, updates blocks include two file system blocks, a triple indirect block, two double indirect blocks (one pointed to by the triple), two single indirect blocks, and the block containing the inode. All but the inode could need to be allocated, and fragmentation may result in each allocation modifying a distinct indirect block.

(f) File systems have traditionally used clever algorithms to group data and metadata for small files near each other in the LBN space. Francine argues that this is less important when using SSDs than when using mechanical disks. Do you agree? Explain your answer.

Yes. SSDs have no mechanical seek time, so they can read any single page at about the same speed, and they have parallelism so they can access two pages at the same time if they are not too close together.

The consequences with write are a little more nuanced because SSDs are going to remap the writes anyway to avoid bad write amplification, but the speed of two writes to adjacent versus different addresses is not different until background cleaning has to defrag. Then, defrag runs fastest if most runs are larger and more contiguous. But, this defrag speed is not on the critical path, so it is much less important than avoiding seeks on a mechanical disk.

Problem 2 : More short answer. [40 points]

(a) As a developer of a new on-disk file system, lets call it ext5, you have decided to optimize ext5 for modern SSDs. One design you are pursuing is to aggressively TRIM SSD LBNs whenever possible (i.e., issuing a TRIM to the SSD as soon as possible). Measurements indicate that each TRIM command takes almost half as long as as a single LBN write. Another developer, Bob, thinks that those TRIM commands are too slow and you should not issue any of them. Do you agree with Bob? Explain your answer.

Yes. (Explanation here is longer than you should write, because some folks were confused.)

Recall that TRIM is a command that tells the SSD it can mark an LBA range as will not be read before being written. This may or may not lead to cleaning any time soon, but it allows the SSD to know that it can clean some flash pages sooner than if the SSD has to wait for an eventual overwrite.

So, TRIM may help reduce future cleaning work, because the SSD has more choices of erase blocks with dead data. But, this is just a fractional improvement, while such expensive TRIM commands certainly slow down all work on the SSD. These commands are quite likely use more time recording dead pages than the reduced cleaning saves.

If TRIM commands are queued up and submitted at a time when the device is otherwise idle, the potentially small benefit of each TRIM command could still be an effective enhancement. But issuing a bunch of slow TRIM commands synchronously with every file delete is not a good idea.

(b) Your fsck program, like most, is intended to be run on a file system partition that is not currently in use. Why is it not safe to use it on a mounted partition in active use? Explain your answer.

An active mounted partition has a running file system that thinks it owns the disk, doesnt have to lock any of it, and that nothing can move unless the file system moves it (so its in-memory data is correct). A concurrent fsck opening and changing the raw disk is making mutations with out locking out the file system, so they will destroy the integrity of each others data structures.

(c) Fred argues that using synchronous writes for update ordering would be faster than write-ahead logging, if every metadata change had to be committed to disk right away, once we all switch to using Flash-based SSDs for our storage. Alice disagrees. Who do you agree with and why?

It depends. Agreeing with either could be correct, depending on the justification.

Alice is more likely correct. With write-ahead logging, multiple metadata updates can be concisely described in a single write I/O, and this is true whether using a mechanical disk or an SSD. The in-place updates would then be completed in the background, which avoids some delays and also allows more coalescing of updates in memory (e.g., multiple updates to the same inode, each recorded in the log but then written in place only after all done).

Fred could be right, for certain workload circumstances. For example, for a write-heavy workload in which the updates are almost all to distinct pieces of metadata. In this case, write-ahead logging involves more total writes (updates to the log plus the in-place updates). While the in-place updates are in the background, the additional writes can increase cleaning overheads and reduce steady-state throughput.

(d) Imagine a system with an I/O workload described by a closed arrival process with zero think time. If the number of users doubles, what happens to the average response time? Explain your answer.

The average response time also doubles, since the average request waits for one request to be completed for each user. That queue time, plus the service time for the request in question, represents the overall response time.

(e) Ted is a new employee at a company that creates file servers that include many disks, using parity to provide redundancy (in a RAID4 arrangement). Ted is excited, and he proposes that they should use that redundancy to detect errors in the data, not just to tolerate disk failures. Specifically, he proposes the following scheme be used for every read: read the entire stripe, check the parity, and use the parity to fix the stripe if incorrect data is discovered. Will Ted's proposed scheme work? Explain your answer.

No, it will not work. (And, it will also be unnecessarily slow.) Parity can be used to detect that there is an error, or to correct an erasure, but it is not sufficient to detect and correct an error.

Problem 3 : Bonus questions. [up to 2 bonus points]

(a) For what file system did you write fsck in lab 1?

ext2fs

(b) Which instructor delivered the most lectures in the first half of the semester?

Prof. Ganger

(c) Which instructor was part of the RAID project at University of California, Berkeley ?

Prof. Gibson

(d) List the names of two TAs.

Henggang Cui, Omkar Gawde, Kiryong Ha, Gaurav Jain, Rohan Sehgal

(e) Why were we unable to enroll everyone on the waitlist, on the first day of class?

Fire marshal rules require that people not be sitting on the floor or in the aisles to attend class. Only as many people as seats available could be enrolled.