CS 537 Handout - Nov 12

- What is a crash?
    - Unexpected interruption of running OS
    - power loss
    - kernel panic
    - user hard reset
- Why is it a problem?
    - FS could be in the “middle” of an update to persistent state
    - Ex: creating a file
        - updates parent dir, data, inode bitmap, child inode
    - Ex: file append
        - data bitmap, inode, data block
        - what can happen if something partially goes to disk?
    - Consistency: all on-disk structures need to be in agreement about file-system state
- Approach #1: FSCK
    - Scan entire disk
    - Find and fix inconsistencies
    - do only if it was a crash (not on clean unmount)
- Approach #2 : Journaling
    - new on-disk structure called “journal”
    - write a “note” about the update to the journal
    - do the update
    - if interrupted by a crash, read the note, complete update (recovery)
    - once update is done, remove the note
    - goal: make multi-block update atomic
- Journal is located at the beginning/middle of the disk
    - could be on own device
- Transaction (tx) basics
    - tx begin
    - update blocks
    - tx commit
        - If the tx commit is found on disk, tx is assumed to be correct and complete
    - Protocol:
        - write [tx begin and update blocks], wait for completion
        - write tx commit. wait for completion
        - checkpoint
- Recovery
    - scan log (not entire disk)
    - find all valid txs
    - replay valid txs

    - replay in order! if tx1 is not complete on disk, tx2 should not be replayed
- Optimization to journaling
    - Problem: every data block is written twice
    - Solution: metadata journaling
    - Write only metadata to journal
    - Write data blocks before commit

Questions

1. What would happen if the journaling protocol just involved writing tx begin, the updates, and the tx commit all at the same time to the disk and then waiting?
2. In metadata journaling, what would happen if the data blocks were written *after* the tx commit block?
3. After a crash, is journal recovery faster than fsck? Why?
4. Could running journal recovery even when there is no crash ever lead to a problem? What about running fsck?
5. Journaling requires things be written to disk in a specific order. What is this order? (for both data and metadata journaling).
6. What would happen in journaling if the commit block was not there?
7. Is checkpointing something that is required for functional correctness of the file system? Would journal recovery in the case of a crash suffice?
8. Is putting the journal on a separate disk a good idea? Why or why not?
9. What should be inside the transaction begin block?
10. Can the transaction commit block be of any size? What happens if it spans multiple blocks?