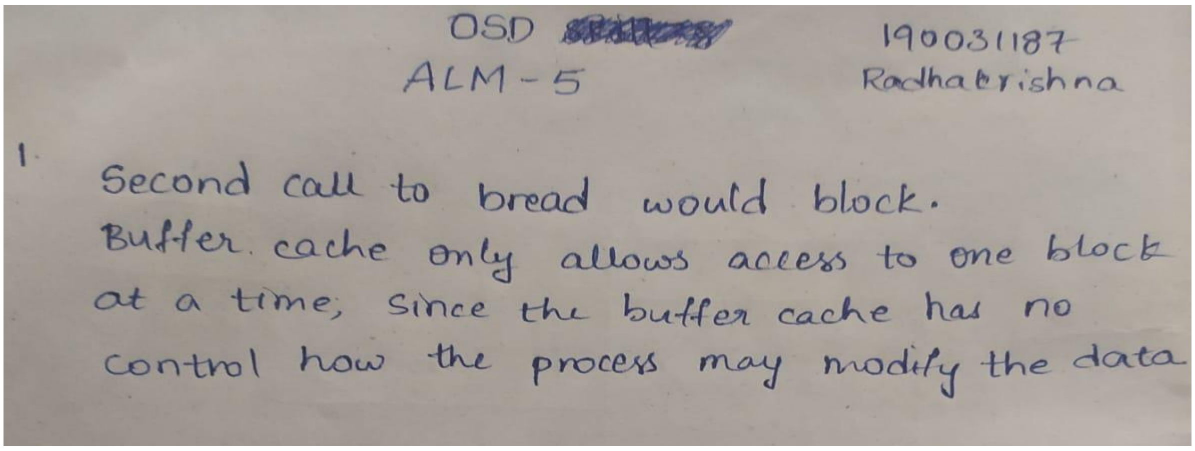
190031187 **NERELLA VENKATA RADHAKRISHNA**

Operating Systems Design 19CS2106S

Session – 5

ALM

1) Consider two processes in xv6 that both wish to read a particular disk block, i.e., either process does not intend to modify the data in the block. The first process obtains a pointer to the struct buf using the function “bread”, but never causes the buffer to become dirty. Now, if the second process calls “bread” on the same block before the first process calls ”brelse”, will this second call to “bread” return immediately, or would it block? Briefly describe what xv6 does in this case, and justify the design choice.



OSD [illegible]

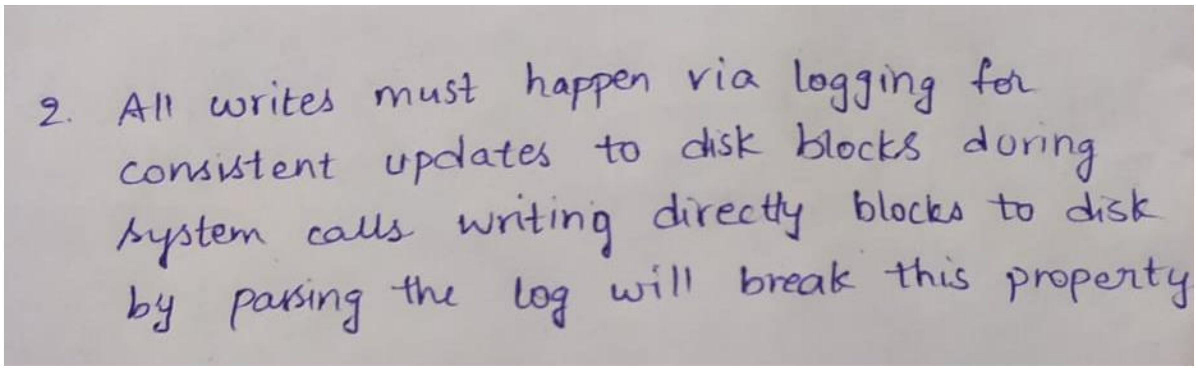
ALM - 5

190031187

Radhakrishna

1. Second call to bread would block.
Buffer cache only allows access to one block at a time, since the buffer cache has no control how the process may modify the data

2) When the buffer cache in xv6 runs out of slots in the cache in the bget function, it looks for a clean LRU block to evict, to make space for the new incoming block. What would break in xv6 if the buffer cache implementation also evicted dirty blocks (by directly writing them to their original location on disk using the bwrite function) to make space for new blocks?

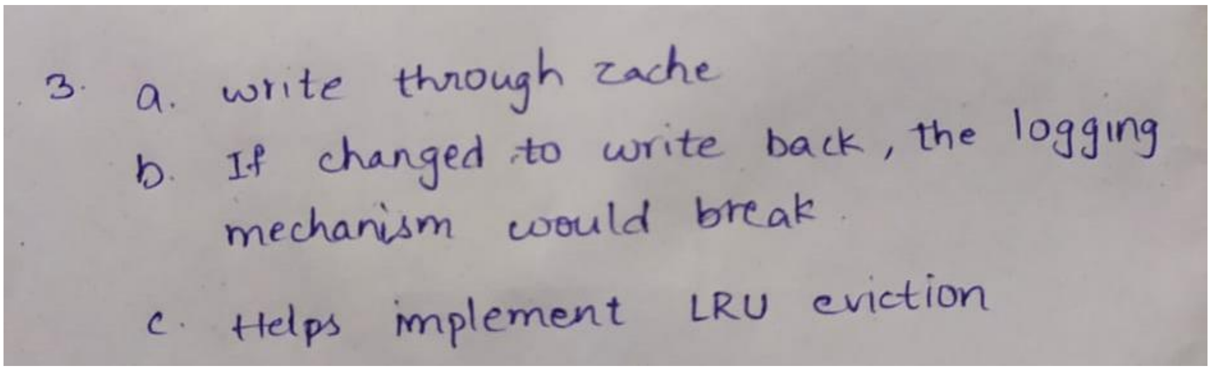


2. All writes must happen via logging for consistent updates to disk blocks during system calls writing directly blocks to disk by parsing the log will break this property

3) (a) Recall that buffer caches of operating systems come in two flavors when it comes to writing dirty blocks from the cache to the secondary storage disk: write through caches and write back caches. Consider the buffer cache implementation in xv6, specifically the bwrite function. Is this implementation an example of a write through cache or a write back cache? Explain your answer.

   (b) If the xv6 buffer cache implementation changed from one mode to the other, give an example of xv6 code that would break, and describe how you would fix it. In other words, if you answered “write through” to part (a) above, you must explain what would go wrong (and how you would fix it) if xv6 moved to a write back buffer cache implementation. And if you answered “write back” to part (a), explain what would need to change if the buffer cache was modified to be write through instead.

   (c) The buffer cache in xv6 maintains all the struct buf buffers in a fixed-size array. However, an additional linked list structure is imposed on these buffers. For example, each struct buf also has pointers struct buf *prev and struct buf *next. What additional functions do these pointers serve, given that the buffers can all be accessed via the array anyway?



3. a. write through zache
   b. If changed to write back, the logging mechanism would break.
   c. Helps implement LRU eviction