7 Concurrent and Distributed Systems (RNW)

SimplisticFS is an in-memory filesystem, implemented as a directed and (mostly) acyclic graph in which nodes represent *directories* (which contain names and pointers to other nodes, including a special entry named ``..'' that points back to its parent node) or *files* (leaf nodes that contain data). The in-memory structure, with the exception of ``..'' entries in directories, is therefore a tree. SimplisticFS does not support hard links, and the ``..'' of the root node points back to itself. Path lookups start at the root node, and pathnames with multiple segments, separated by ``/'', are implemented as lookup operations on successive nodes. For example, opening ``/foo/bar'' will look up ``foo'' in the root, and then ``bar'' relative to the `foo` node.

Fine-grained locking adds a read-write lock to each node to ensure safe concurrent access. Operations for reading a directory entry (e.g., to list the contents or look up a child), or for reading file data will: acquire the node lock for read; perform the operation; and then release it. Operations for mutating a directory entry, (e.g., to add or remove a child node of a directory), or for modifying file data will: acquire its lock for write; perform the operation; and then release it. The lock implementation permits read recursion, write recursion, and race-free lock upgrades from read to write by threads.

(*a*) For (*i*) files and (*ii*) directories, explain, giving examples, how using a read-write lock may improve performance compared to mutual exclusion. [4 marks]

(*b*) The developers discover that compound operations, such as recursive path lookup, suffer from race conditions. They decide to adopt *strict two-phase locking* across compound operations to resolve this problem.

(*i*) Define *strict 2-phase locking* and describe how to apply it. [4 marks]

(*ii*) Explain why deadlock cannot occur prior to this change. [2 marks]

(*iii*) The new strategy suffers deadlocks when files are removed under high load. Given that removing a file requires a write lock on its parent directory, give an example of how a deadlock might occur. [4 marks]

(*c*) Moving from a "giant lock" to this finer-grained model focused on files and directories improves performance for some but not all workloads.

(*i*) Describe and explain an example of a workload in which file-granularity locking is unlikely to eliminate lock contention. [2 marks]

(*ii*) Propose a more granular locking strategy to improve parallelism with respect to (*c*)(*i*). Describe the potential performance benefit with respect to that workload, and additional overhead that might be incurred. [4 marks]