CS425 Distributed Systems MP3 Report

Aditi Sadwelkar, Aryaman Ramchandran

a) Design

Replication Strategy

Files are mapped to ring positions using SHA-1 (secure hashing algorithm), and replicas are stored on the first 3 successor nodes on the ring to tolerate up to 2 simultaneous machine failures. The ring.py module implements the consistent hash ring where both nodes and files are hashed to positions. When a file is created or accessed, the system calls get_successors to determine the exact 3 nodes that should store the file. Node IDs follow the format hostname:port:timestamp to ensure uniqueness across node restarts.

Consistency Guarantees Implementation

Each client maintains a sequence counter that increments with each append. When blocks are merged, sorting by client_id, sequence_num, timestamp ensures that appends from the same client always appear in the order they were issued.

The merge operation collects blocks from all replicas, deduplicates them using block_id as the unique identifier, sorts them to establish a total order, and redistributes the merged result to all replicas.

The ConsistencyManager tracks pending writes per client_id, filename pair. When a client issues a GET request, the system checks has_pending_writes() and blocks the read if that client has outstanding appends to the file, ensuring the client always sees its own writes.

Re-Replication Implementation

The ReplicationManager runs a monitoring thread that checks every 5 seconds for under-replicated files. The check_and_rereplicate() method iterates through all stored files, queries each replica to determine actual replication count, and triggers re-replication if the count falls below 3. When a node failure is detected through MP2, the handle_node_failure() method is called, which removes the failed node from the ring and triggers the monitoring thread to fix under-replicated files.

Merge Algorithm

The merge operation is coordinated by the first replica in the successor list. The coordinator sends requests to all replicas, including itself, collects all blocks, and passes them to consistency.merge_blocks(). This method deduplicates blocks, sorts, and validates ordering using validate_block_ordering() to ensure no sequence gaps exist per client. The merged blocks are then sent to all replicas via messages with the replace=True flag, causing each replica to delete old blocks and write the new merged set atomically.

CS425 Distributed Systems MP3 Report

Aditi Sadwelkar, Aryaman Ramchandran

b) Past MP Use

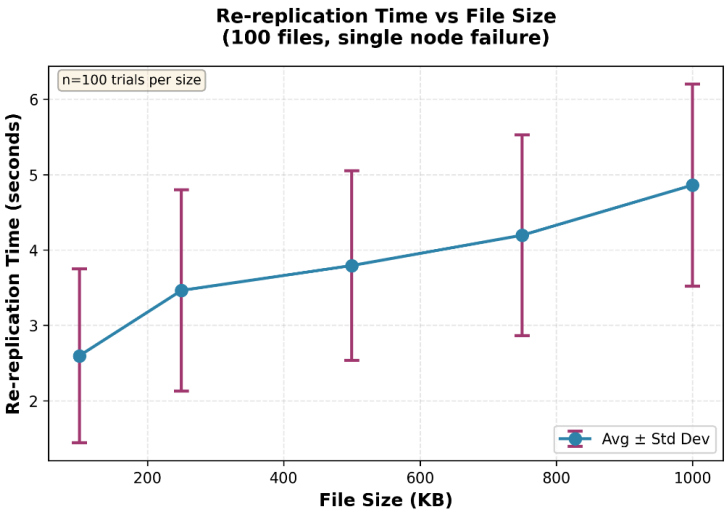
MP2's membership protocol is integrated directly into node.py through the MembershipService class and provides the foundation for our HyDFS's failure detection and node discovery. When nodes join or fail, MP2's PinkAck protocol detects the change, and HyDFS triggers rebalancing or re-replication.

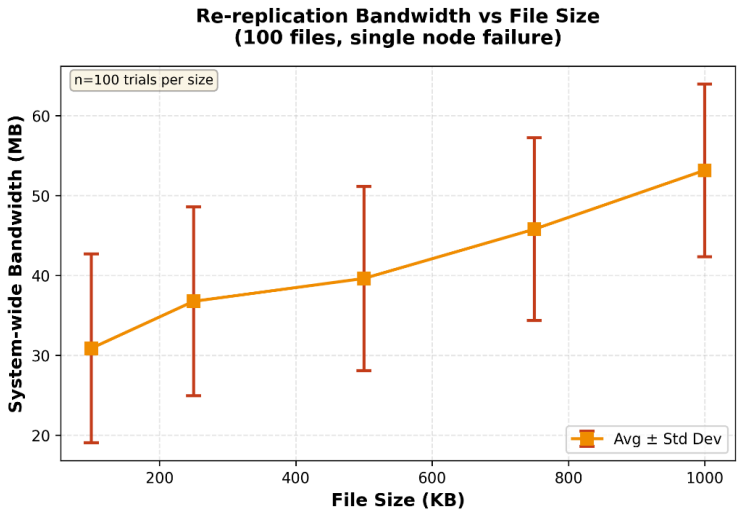
MP1's distributed grep was useful for debugging by allowing queries like grep "REPLICATION ERROR" -i across all VM logs to quickly identify issues with file placement, merge conflicts, or network errors during development and testing.

c) LLM Use

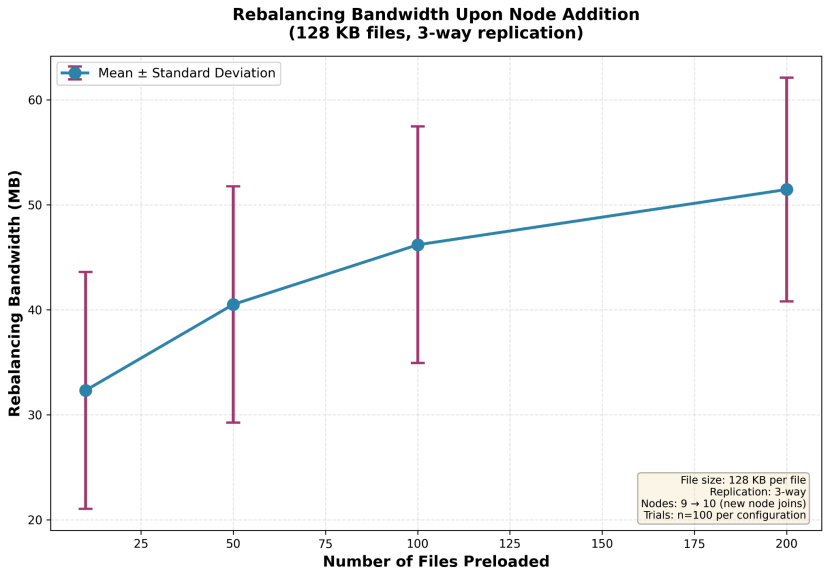
LLM's have been used to bounce back and forth ideas regarding system design and implementation. Conceptual struggles and weighing strategy options to help make clear and informed decisions on direction. The LLM logs are stored in <llm_logs> directory.

(i) (Overheads)





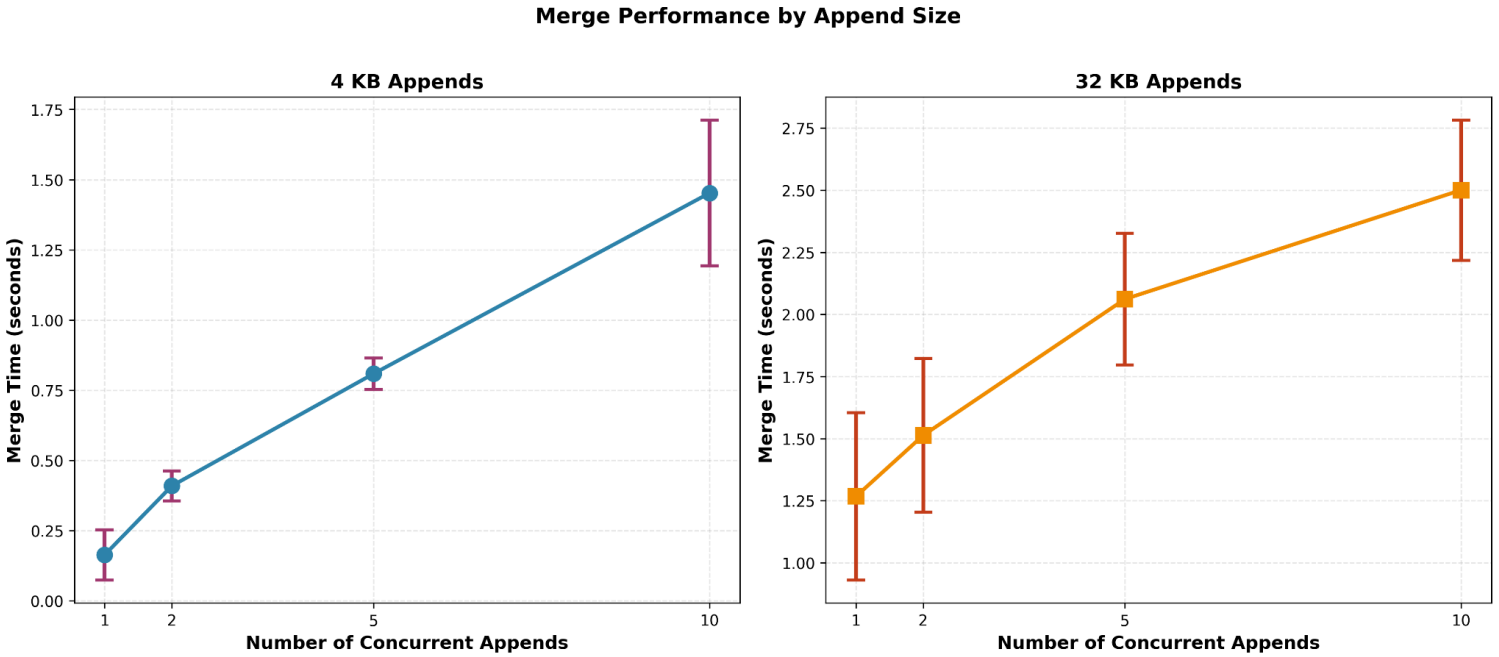
(ii) (Rebalancing Overheads)



CS425 Distributed Systems MP3 Report

Aditi Sadwelkar, Aryaman Ramchandran

(iii) (Merge Performance)



(iv) (Subsequent-merge Performance)