Design Document for memFS

Sambhav Jain
24CSC60R39

Contents

1 Introduction

memFS is an in-memory file system designed for rapid data access in RAM. It supports essential file operations like creation, deletion, writing, reading, and listing. Due to its RAM-based nature, data in memFS is volatile, optimized for environments where data persistence is non-critical, such as temporary caching and testing.

2 Functional Requirements

memFS will support the following commands:

- **Create**: `create -n (number of files) filenames (in order)`
- **Write**: `write -n (number of files) <filename> "<text to write>"`
- **Delete**: `delete -n (number of files) filenames (in any order)`
- **Read**: `read filename`
- **List**: `ls` or `ls -l` for detailed listing
- **Exit**: `exit`

3 System Modes

memFS operates in two distinct modes:

- **User Mode**: This mode is for standard file operations (create, write, delete, read, list, exit) as per user input.
- **Benchmarking Mode**: This mode benchmarks the performance of memFS under various workloads and thread counts.

The program initially prompts the user to select a mode: User Mode or Benchmarking Mode.

4 System Architecture

4.1 Core Modules

- **File Management**: Manages file creation, deletion, and storage in RAM.
- **Command Interpreter**: Parses and interprets user commands in User Mode. In Benchmarking Mode, commands are directly invoked based on preset workloads.
- **Memory Management**: Monitors available memory for file storage, reports errors if insufficient memory.

4.2 Multithreading Architecture

Threads handle concurrent file operations, reducing latency. Synchronization mechanisms manage concurrent access to shared resources.

4.3 Error Handling

- Reports errors for duplicate file names during creation and missing files during write/read/delete operations.
- Continues processing remaining files if an error is encountered.

5 Module Design

5.1 User Mode

- Implements core commands for file manipulation and listing as per user input.
- **Commands Supported**: Create, Write, Delete, Read, List, and Exit.

5.2 Benchmarking Mode

- **Workloads**: Executes sets of 100, 1,000, and 10,000 commands (create, write, read, delete).
- **Metrics Collected**: Average latency, CPU utilization, and memory usage.

This mode bypasses the command interpreter, directly invoking the relevant functions with predefined parameters.

6 Memory Management

Manages allocation and deallocation of memory for files. Tracks usage to prevent memory over-allocation.

7 Benchmarking Strategy

7.1 Workloads

- 100, 1,000, and 10,000 operations (create, write, read, delete).
- Run against 1, 2, 4, 8, and 16 threads.

7.2 Metrics

- **Average Latency**: Time taken per command, averaged over operations.
- **CPU Utilization**: Measures CPU load across thread counts.
- **Memory Usage**: Monitors RAM usage under different workloads.

7.3 Expected Observations

- Latency should decrease with thread count to an optimal point, beyond which overhead may increase latency.
- CPU utilization and memory usage should grow with thread count.

8 Benchmarking Report

A benchmarking report has been prepared based on tests with 100, 1,000, and 10,000 commands, evaluating the performance of memFS. This report includes:

- Average Latency
- CPU Utilization
- Memory Usage

The results demonstrate the effects of varying thread counts on performance, revealing optimal configurations for minimizing latency and resource consumption. After you run the benchmarking, all the timing results will be printed on command prompt and the above items will be stored in a seperate .txt file.

9 Conclusion

memFS provides a lightweight, in-memory file system optimized for speed. Multithreading enhances performance by handling multiple file operations concurrently. Benchmarking validates the impact of thread count on system performance, ensuring optimization for real-world usage.