FILE INDEXING

GROUP : 19

PUDOTA HARSHITHA - CB.EN.U4CSE21248
THIMMISETTY SAILEELA - CB.EN.U4CSE21264

TABLE OF CONTENT

1. Introduction and Objective

Efficient File Indexing with Hashtrees is a data structure approach that allows for fast and secure file indexing. Hashtrees use a combination of hashing and tree structures to ensure that files can be located quickly and with a high degree of accuracy. This presentation will provide an overview of how hashtrees work and their benefits.

To provide a flexible and efficient solution for managing files and directories within a file system, with the objectives of effective data organization, practical application usage, and optimal performance in terms of time and space complexity.

Hashtrees are ideal for situations where fast and accurate file retrieval is critical. Some common use cases include search engines, file sharing networks, and cloud storage systems. Additionally, hashtrees can be used to ensure the integrity of data in fields such as digital forensics and financial auditing.

2. Hybrid Data Structure Implementation

1.Hash Map: The hash map is used as the primary data structure for storing key-value pairs. It provides fast access and retrieval of elements based on their hash codes. The hash map is implemented using an array, where each element of the array is a linked list or a balanced binary search tree (in case of collision). The hash map supports constant time complexity for insertion, deletion, and retrieval on average

2.Binary Search Tree: In cases where collisions occur in the hash map, a binary search tree is used to handle the collisions. The binary search tree provides efficient searching, insertion, and deletion operations with a time complexity of O(log n), where n is the number of elements in the tree. The binary search tree ensures that the elements are stored in a sorted order based on their keys.

2. Hybrid Data Structure Implementation

1.Integration of Data Structures:When a key-value pair is inserted into the hybrid data structure, it first computes the hash code of the key. The hash code determines the index of the corresponding element in the hash map array.If there is no collision, the key-value pair is directly inserted into the linked list or binary search tree at the computed index.If there is a collision, the binary search tree is used to handle it. The key-value pair is inserted into the binary search tree associated with the index, ensuring that the elements within the tree are sorted based on their keys.During a search or retrieval operation, the hash code of the key is computed, and the corresponding linked list or binary search tree is traversed to find the desired element.

2.Design Choices:The use of a hash map allows for fast access and retrieval of elements based on their hash codes, ensuring efficient operations.The binary search tree is employed to handle collisions, providing efficient search, insertion, and deletion operations in case of multiple elements with the same hash code.The hybrid data structure leverages the strengths of both the hash map and binary search tree, enabling efficient operations in scenarios with varying key distributions and potential collisions.

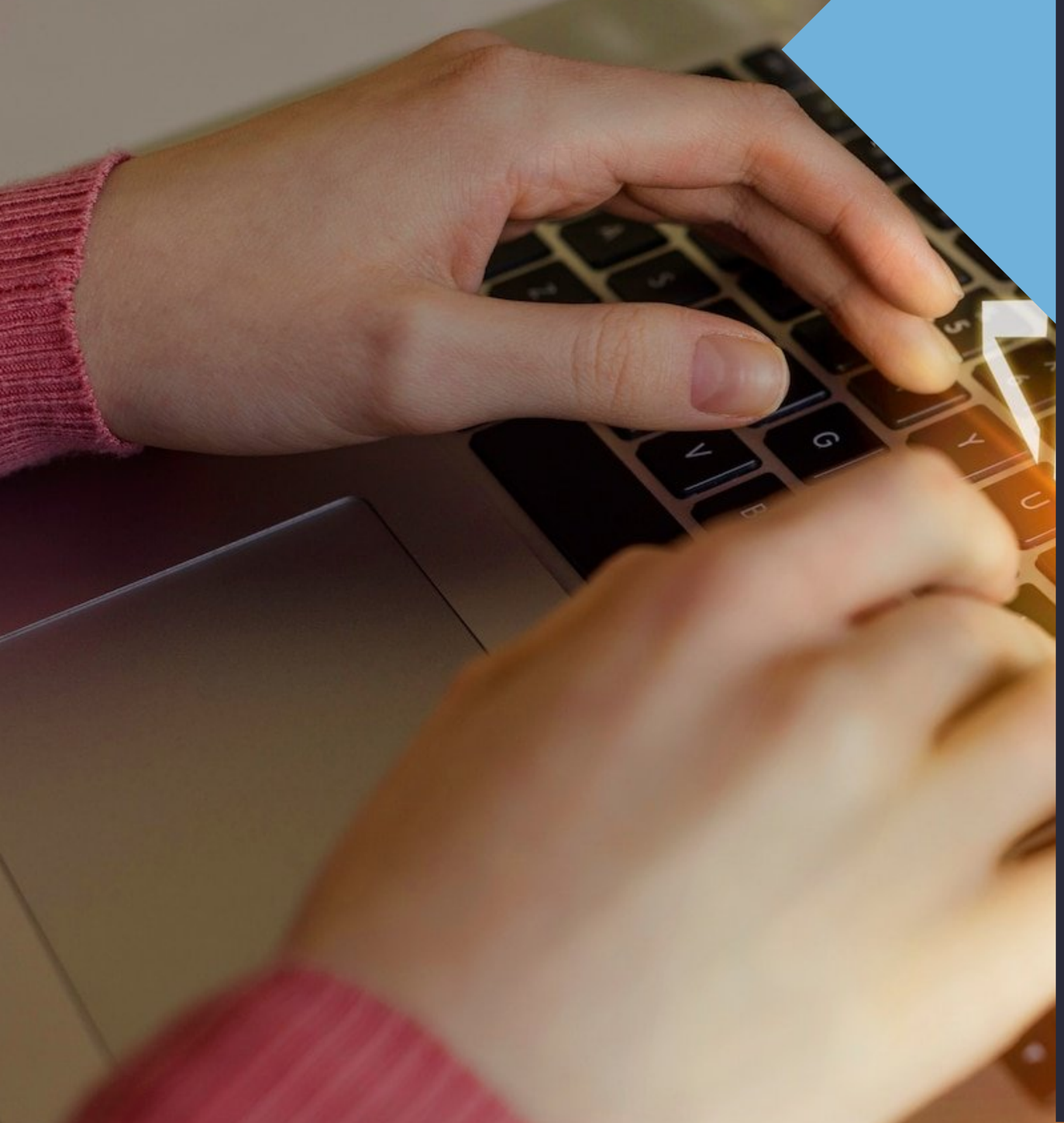
3. Practical Application

The functionalities to manipulate and manage files and directories within the file system tree. This can be useful in various applications, such as file management systems, operating systems, data backup utilities, and more.

1. File Organization
2. Efficient File Addition
3. Size and Hash Indexing etc.,

4. Time and Space complexity

Adding a File
TIME- O(length of path)
SPACE- O[1]
Creatin a New Directory
TIME- O(length of path)
SPACE- O[1]
Renaming a File
TIME- O(length of path)
SPACE- O[1]
Sorting Files by Last Modified
TIME- O(m log n) m-unique file sizes, n-length of path
SPACE- O[1]
Calculating Directory Size
TIME- O(number of nodes)
SPACE- O[1]
Deleting a File
TIME- O(length of path)
SPACE- O[1]
Moving a File
TIME- O(length of path)
SPACE- O[1]
Opening a Directory and Adding a File
TIME- O(length of path)
SPACE- O[1]
Listing Directory Contents
TIME- O(length of path)
SPACE- O[1]
Getting Files above a Certain size
TIME- O(length of path)



5. Experimental Evaluation and Result

Experimental Setup:

Hardware: Determine the hardware specifications of the system on which the experiments will be conducted, including the CPU, RAM, and storage capacity. These specifications will impact the performance results. Software Environment: Set up the required software environment, including the programming language, libraries, and frameworks used in the implementation. Data Set: Prepare a representative data set that simulates real-world scenarios. The data set should include a variety of files with different sizes and contents. This will allow for a comprehensive evaluation of the hash tree's performance across different scenarios.

5. Experimental Evaluation and Result

Methodology:

Define Metrics: Determine the performance metrics to measure, such as the time taken for various operations (e.g., file insertion, deletion, searching, and integrity verification) and memory utilization. Test Cases: Design a set of test cases that cover different scenarios, including varying file sizes, different numbers of files, and different types of operations. Ensure that the test cases are representative of the intended use cases for the hash tree. Execution: Run the experiments by performing the defined operations on the hash tree using the test cases. Measure the execution time for each operation and record the memory utilization. Repeat and Average: To ensure accurate results, repeat each experiment multiple times and calculate the average execution time. This helps to account for any variations caused by system factors or other external influences. Comparative Analysis: Compare the performance of the hash tree implementation with individual data structures (e.g., arrays or linked lists) for the same set of operations.

6. Discussion and conclisionsion

Hashtrees offer a unique and efficient approach to file indexing that provides fast and accurate file retrieval, easy detection of any changes or corruption in the file system, and secure data storage. By implementing hashtrees, organizations can improve their file management systems and ensure the integrity of their data.

Efficient lookup and traversal
Scalability and flexibility
Space efficiency
Hashing and integrity
Code simplicity

To address these limitations and enhance the practicality and effectiveness of the hybrid data structure, further enhancements could include implementing more advanced file system features, incorporating error handling and validation mechanisms, optimizing file system traversal algorithms, and considering stronger hash functions for better integrity verification.

The overall success of the above work lies in its ability to showcase the effectiveness of the hybrid data structure, provide insights into its performance characteristics, and offer practical applications in real-world scenarios. The work contributes to the broader understanding of data structures and their combinations, guiding future improvements and optimizations in this field.


```
1. List directory contents
2. Add file
3. Create new directory
4. Rename file
5. Sort files by last modified
6. Calculate directory size
7. Open directory
8. Get files above size
9. Delete file
10. Move file
11. Visualize file system tree
0. Exit
Enter your choice: 2
Press 'b' to go back
a
Enter the file path: /documents/files.txt
Enter the file size: 1024
```


```
Menu:
1. List directory contents
2. Add file
3. Create new directory
4. Rename file
5. Sort files by last modified
6. Calculate directory size
7. Open directory
8. Get files above size
9. Delete file
10. Move file
11. Visualize file system tree
0. Exit
Enter your choice: 1
Press 'b' to go back
a
Enter the directory path: /documents
Contents of directory: /documents
files.txt
```

```
Menu:
1. List directory contents
2. Add file
3. Create new directory
4. Rename file
5. Sort files by last modified
6. Calculate directory size
7. Open directory
8. Get files above size
9. Delete file
10. Move file
11. Visualize file system tree
0. Exit
Enter your choice: 3
Press 'b' to go back
a
Enter the parent directory path: /documents
Enter the new directory name: images
```

Thank you