Data Structures

Project Documentation

Department:

Computer Science

Group Members:

Shaif Imran (22i-1024)

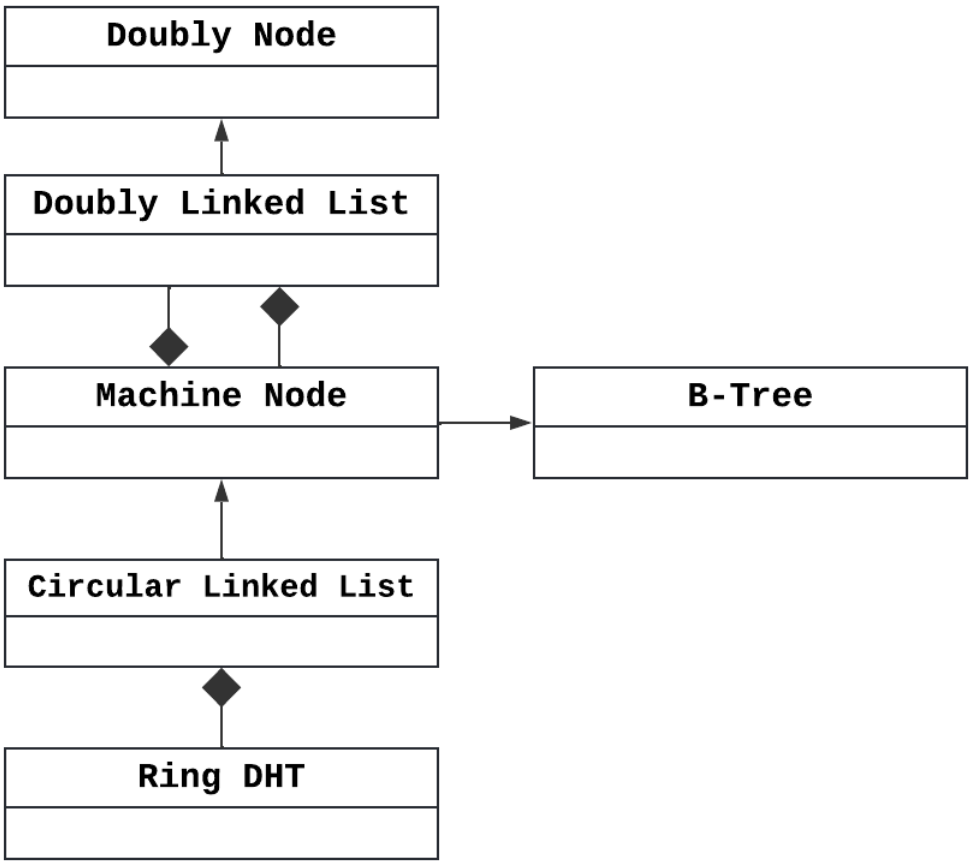
Adeel Mehmood Ansari (22i-0979)

Abdul Wasay (22i-1198)

Introduction:

This is a C++ implementation of an interplaneraty file system. IPFS file systems provide an efficient solution to file sharing. The project uses a distributed hash table to scatter files between different machines, instead of storing them on one machine. The functionalities implemented play a pivotal role in navigating the network, allowing seamless access to files regardless of their location on the network. Through this C++ implementation, we aim to showcase the versatility of IPFS file system.

Classes:



Doubly Node:

Attributes:

- **Data:** Pointer to the stored data.
- **Prev**: Pointer to the previous node.
- **Next**: Pointer to the next node.

Methods:

- **getData():** Retrieves the data stored in the node.
- **getNext():** Retrieves a pointer to the next node.
- **getPrev():** Retrieves a pointer to the previous node.
- **setData():** Sets the data in the node.
- **setNext():** Sets the next node.
- **setPrev():** Sets the previous node.

Doubly Linked List:

Attributes:

- **Head:** Pointer to the head node of the linked list.
- **Size:** Tracks the current size of the linked list.

Methods:

- **isEmpty():** Checks if the linked list is empty.
- **print():** Prints the elements stored in the linked list.
- **getHead():** Retrieves the head node.
- **setHead():** Sets the head node.
- **insertToHead(T* d):** Inserts a new node at the beginning of the list.
- **insert(T* d):** Inserts a new node at the end of the list.
- **search(int d):** Searches for a specific key in the list and returns a boolean.
- **search_node(T* d):** Searches for a node based on a key and returns the machine.
- **insertAtIndex(T* d, Bigint index):** Inserts a node at a specified index.
- **deleteData(T* d):** Deletes a node with specific data.

Circular Linked List:

Attributes:

- **Head:** Pointer to the head node of the circular linked list.

Methods:

- **CircularLinkedList():** Constructor initializing the circular linked list.
- **insert(Bigint data):** Inserts a new node in ascending order based on the provided data.
- **makeRoutingTables(int identifierBit, Bigint identifierSpace**): Creates routing tables for each node in the list based on identifier bit and space.

- **deleteRoutingTables():** Deletes all routing tables associated with nodes in the list.
- **printRoutingTable():** Prints the routing tables for each node in the list.
- **search(Bigint d):** Searches for a node with a given data value in the list and returns a boolean.
- **searchNode(Bigint d):** Searches for a node with a specified data value and returns the node.
- **update(Bigint d, Bigint newnumber):** Updates the data of a node with the given new data.
- **insertAtIndex(Bigint d, Bigint index):** Inserts a new node at a specified index in the list.
- **deleteData(Bigint d):** Deletes a node with a specified data value from the list.
- **isEmpty():** Checks if the circular linked list is empty.
- **getHead():** Retrieves the pointer to the head node of the circular linked list.
- **print():** Prints the content of the circular linked list.

Machine Node:

Attributes:

- **Key:** Holds the key for the machine.
- **B_Tree:** Represents the B-tree of file keys storing the file as a key generated through H(file Content)->key.
- **RoutingTable:** Doubly linked list storing routing table information.
- **Next:** Pointer to the next MachineNode.
- **Path:** Holds the path to the directory where machine-related data is stored.

Methods:

- **MachineNode():** Default constructor initializing key to 0 and next to NULL. Initializes the routingTable.
- **MachineNode(Bigint d, MachineNode* p**): Constructor setting the key to a specified value d and initializing the next pointer. Also initializes the routingTable.
- **getData():** Retrieves the key value.
- **getNext():** Retrieves the pointer to the next MachineNode.
- **setData(Bigint d):** Sets the key to a specified value of data.
- **setNext(MachineNode* node):** Sets the pointer to the next MachineNode.

B-Tree Node:

Attributes:

- **Keys:** Array to hold keys.
- **Children:** Array to hold pointers to child nodes.
- **num_keys:** Number of keys present in the node.
- **isLeaf:** Flag indicating if the node is a leaf node or not.

Methods:

- **BTreeNode(bool leaf):** Constructor initializing a node, setting isLeaf, num_keys, and allocating memory for keys and children.
- **insertNode(Bigint key):** Inserts a key into the B-tree node.
- **splitNodes(BTreeNode* toSplit, int i):** Splits a node into two when it's full.
- **searchNode(Bigint key):** Searches for a key within the node and its children.
- **traverse():** Traverses the B-tree.
- **removeFromLeaf(int idx):** Removes a key from a leaf node.
- **getPredecessor(int idx):** Retrieves the predecessor key of a given key index.
- **getSuccessor(int idx):** Retrieves the successor key of a given key index.
- **fill(int idx):** Handles filling a child node during deletion.
- **borrowFromPrev(int idx):** Borrow keys from the previous node during deletion.
- **borrowFromNext(int idx):** Borrow keys from the next node during deletion.
- **merge(int idx):** Merges nodes during deletion.
- **removeFromNonLeaf(int idx):** Handles deletion of a key from a non-leaf node.
- **findKey(Bigint k):** Finds the index of a given key.
- **deletion(Bigint k):** Deletes a key from the B-tree.

B-Tree:

Attributes:

- **Root:** Pointer to the root node of the B-tree.

Methods:

- **getRoot():** Retrieves the root node.
- **traverse(BTreeNode* node):** Traverses the B-tree nodes.
- **splitKeysHelperInsert(Bigint key):** Helper function to split keys for insertion.
- **splitKeysHelperDelete():** Helper function to split keys for deletion.
- **insert(Bigint key):** Inserts a key into the B-tree.
- **deletion(Bigint k):** Deletes a key from the B-tree.

Ring DHT:

Attributes:

- **m:** CircularLinkedList pointer to store machines in a circular manner.
- **numMachines:** Total number of machines in the ring.
- **identifierSpace:** Total identifier space for machines.
- **identifierBit:** Bit length for identifiers.
- **filePath:** Path where machine data is stored.

Methods:

- **ringDHT():** Constructor initializing default values and creating a directory at filePath.
- **userInput():** Takes user input for setting identifier space and number of machines, assigns unique identifiers to machines.
- **printRoutingTable():** Prints the routing table of the machines.
- **printMachines():** Prints the available machines.
- **deleteMachine():** Deletes a machine from the ring and adjusts routing tables accordingly.
- **addMachines():** Adds machines to the ring and updates routing tables.
- **splitKeysInsert(MachineNode* node):** Splits keys during insertion of a machine.
- **splitKeysDelete(MachineNode* node):** Splits keys during deletion of a machine.
- **printRoutingTableofSpecificMachine():** Prints the routing table of a specific machine.
- **addFile():** Adds a file to the ring and handles file storage and routing.
- **removeFile():** Removes a file from the ring.
- **menu():** method for handling a menu system.
- **~ringDHT():** Destructor removing all data stored at filePath.