Title: Implementation of a Distributed File System

# 22i – 1184 – Hassan Zafar

# 22i – 1304 – Mesam E Tamaar Khan

**CircularLinkedList**

**Node**

* Data: string
* keyList: KeyNodeList\*
* nextNode: Node\*
* prevNodeL: Node\*
* parent: Node\*

**BTree**

* root:BtreeNode\*
* BTree(int \_t)
* Traverse()
* Extraction()
* Insertion()
* deletion()

**DadNode**

* Node: Node\*
* Id: BigInt
* Next: DadNOde\*
* Prev: DadNode\*
* DadNode(BigINt Node\*)

**keyNode**

**KEYNODE**

* Key: BigInt
* nextKeyNode: KeyNode\*
* prevKeyNode: KeyNode\*

--------------------------------------------------

* KeyNode(BigInt a)
* insert(BigInt x)
* isEmpty(): bool
* print()
* search(BigInt x)

**KeyNodeList**

* Head: KeyNode\*
* KeyNodeList()
* insert(BigInt x)
* isEmpty(): bool
* print()
* search(BigInt x)

**BTreeNode**

* T: int
* C: BTreeNode[]
* Keys: keyNode[]
* n: int
* leaf: bool
* BTreeNode(int \_t,bool \_leaf)
* Traverse()
* Extraction()
* Insertion()
* deletion()
* mergeRecursive()

# Class 1: KeyNode

Attributes

key: BigInt - Represents a unique key.

nextKeyNode: Pointer to KeyNode - Points to the next node in the list.

prevKeyNode: Pointer to KeyNode - Points to the previous node in the list.

## Methods:

KeyNode(BigInt a = -1): Constructor - Initializes a KeyNode with a default key value.

insert(BigInt x): Inserts a new key into the list.

isEmpty(): Checks if the list is empty.

print(): Prints the keys in the list.

search(BigInt x): Searches for a key in the list.

# Class 2: KeyNodeList

## Attributes:

head: Pointer to KeyNode - Points to the head of the key node list.

## Methods:

KeyNodeList(): Constructor - Initializes an empty key node list.

insert(BigInt x): Inserts a new key into the list.

isEmpty(): Checks if the list is empty.

print(): Prints the keys in the list.

search(BigInt x): Searches for a key in the list.

# Class 3: BTree

## Attributes:

root: Pointer to BTreeNode - Points to the root of the B-tree.

t: Integer - Represents the minimum degree of the B-tree.

## Methods:

BTree(int \_t = 3): Constructor - Initializes an empty B-tree with a specified minimum degree.

traverse(): Traverses the B-tree and prints its keys.

extract(KeyNodeList\* l): Extracts a new B-tree from the specified key node list.

insertion(string k, string data, BigInt machineKey): Inserts a new key, data, and machine key into the B-tree.

deletion(string k): Deletes a key from the B-tree.

mergeRecursive(BTreeNode\* current, BTreeNode\* other): Merges the keys of two B-tree nodes.

# Class 4: BTreeNode

## Attributes:

keys: Array of keyNode - Represents the keys in the node.

t: Integer - Represents the minimum degree of the B-tree.

C: Array of BTreeNode\* - Represents the child nodes.

n: Integer - Represents the number of keys in the node.

leaf: Boolean - Indicates whether the node is a leaf or not.

## Methods:

BTreeNode(int \_t, bool \_leaf): Constructor - Initializes a B-tree node with a specified minimum degree and leaf status.

traverse(): Traverses the B-tree node and prints its keys.

extractRecursive(KeyNodeList\* l, BTree\*& newTree): Extracts keys from the node recursively based on the specified key node list.

findKey(string k): Finds the index of a key in the node.

insertNonFull(string k, string d, BigInt m): Inserts a new key, data, and machine key into a non-full node.

splitChild(int i, BTreeNode\* y): Splits a child node.

deletion(string k): Deletes a key from the node.

removeFromLeaf(int idx): Removes a key from a leaf node.

removeFromNonLeaf(int idx): Removes a key from a non-leaf node.

getPredecessor(int idx): Gets the predecessor key of a key in the node.

getSuccessor(int idx): Gets the successor key of a key in the node.

fill(int idx): Fills the child that has less than the minimum number of keys.

borrowFromPrev(int idx): Borrows a key from the previous sibling.

borrowFromNext(int idx): Borrows a key from the next sibling.

merge(int idx): Merges the current node with its sibling.

mergeRecursive(BTree\* Tree): Recursively merges the keys of the B-tree node.

# Class 5: Dadnode

## Attributes:

id: BigInt - Represents the unique identifier.

node: Pointer to Node - Points to a node.

next: Pointer to Dadnode - Points to the next dad node.

prev: Pointer to Dadnode - Points to the previous dad node.

## Methods:

Dadnode(BigInt x = 0, Node\* p = nullptr): Constructor - Initializes a Dadnode with a default identifier and node.

# Class 6: Node

## Attributes:

data: String - Represents the data stored in the node.

keyList: Pointer to KeyNodeList - Points to a list of keys.

nextNode: Pointer to Node - Points to the next node in the linked list.

prevNode: Pointer to Node - Points to the previous node in the linked list.

## Methods:

Node(string d = ""): Constructor - Initializes a node with specified data.

insert(BigInt x): Inserts a new key into the key list.

remove(BigInt x): Removes a key from the key list.

search(BigInt x): Searches for a key in the key list.

getFirstKey(): Returns the first key in the key list.

merge(Node\* nextNode): Merges the node with the next node.

split(): Splits the node into two nodes.

getParent(): Returns the parent node.

setParent(Node\* parent): Sets the parent node.

findSuccessor(): Finds the successor node in the linked list.

findPredecessor(): Finds the predecessor node in the linked list

# Contribution:

* Big int, Btree – Hassan Zafar
* LinkedList, FT – Mesam e Tamaar Khan