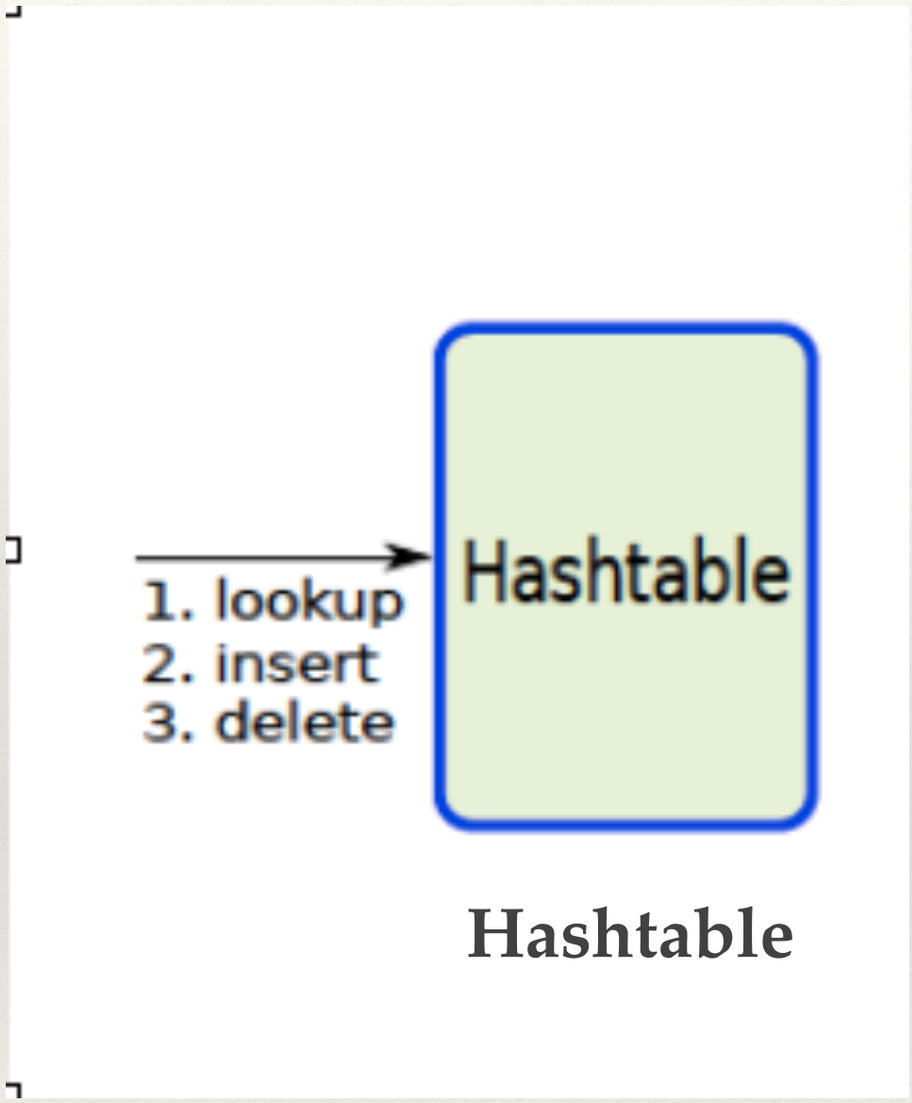
PASTRY : Distributed Hash Table

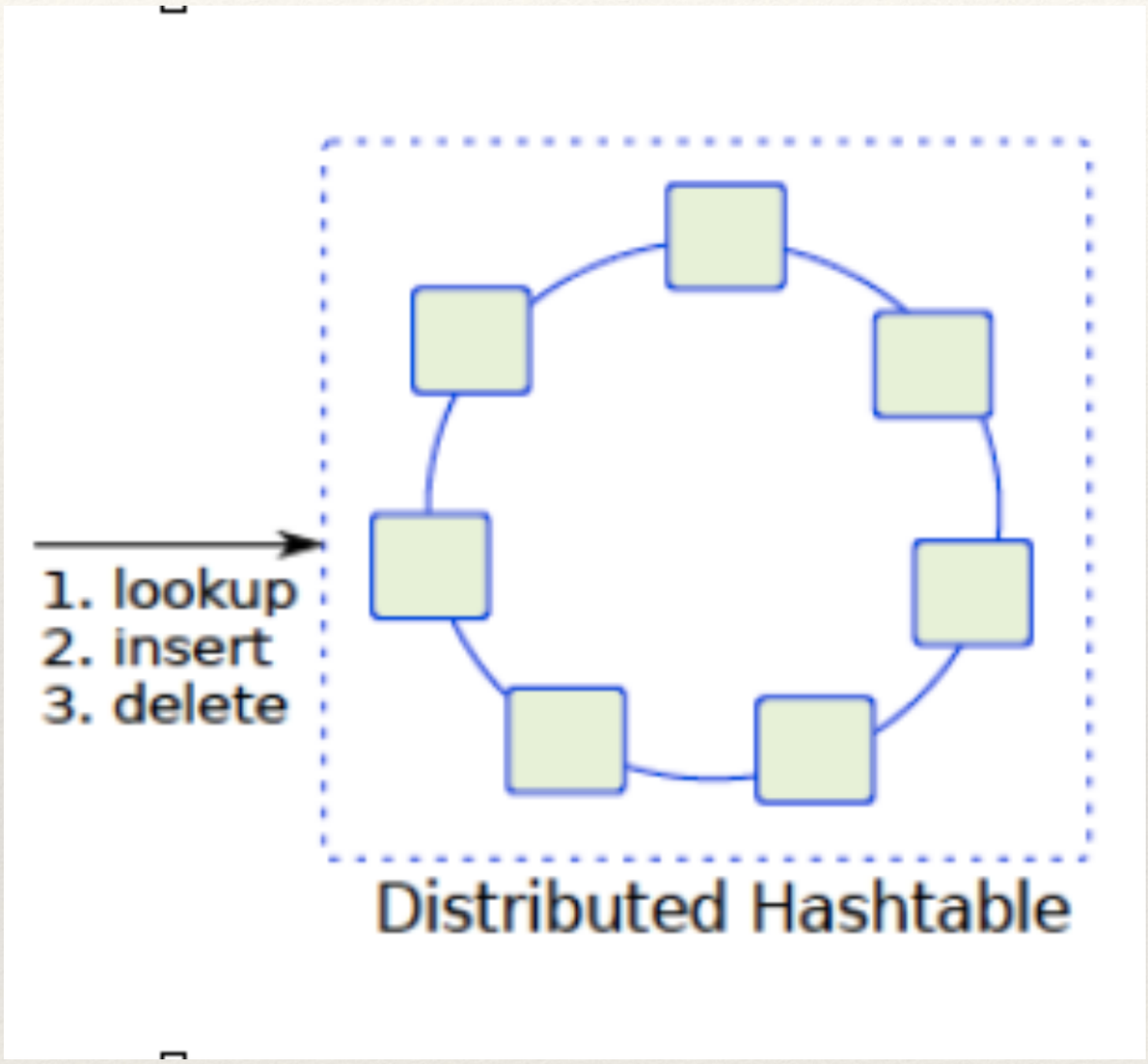
4 Nodes
Juhi Gupta
Kaushik LV
Rishabh Murarka
Krishna Dwypayan

Distributed Hash Table

- Class of **decentralised** distributed system that provides look up similar to a hash table.
- Stores a (key, value) pair, which any node can efficiently retrieve.
- DHTs can store more data than centralised databases.
- Scalable, both in terms of data and users. Ideal for web scale storage.
- Better load balancing.


1. lookup
2. insert
3. delete
Hashtable

Hashtable


1. lookup
2. insert
3. delete
Distributed Hashtable

Pastry : Introduction

- Pastry is a structured type of an overlay network
- The structure enables efficient resource search in large scale distributed storage networks.
- The structure imposed by Pastry is that of DHT.
- Structure to accommodate the participating nodes and their corresponding routing data is maintained.
- Also maintained is a structured format of join/leave mechanism.

Why Pastry?

- Its main goal is to create a completely decentralised and structured overlay network.
- In Overlay network, objects can be efficiently located and lookup queries efficiently routed.
- The Pastry routing is based on numeric closeness of IDs.
- When forwarding a message to a destination key, a node will choose the node in its routing table with the longest prefix match.

Design of Pastry Network

- Pastry network consists of nodes acting both as a server and a client.
- Every node has a unique 128 bit nodeId.
- The nodes are conceptually organised as a ring.
- Each node stores part of the Distributed Hash Table.

Design of Pastry Node

- Each node is assigned a 128 bit NodeID; generated by applying a cryptographic hash(MD5) on the node's IP address and port no.
- The structure of every pastry node consists of:
  - **Leaf Set** has entries based on logical proximity.
  - **Routing Set** has entries based on overlay network of Pastry.
  - **Neighbourhood Set** has entries based on physical proximity.

Routing Algorithm

- Admin sends the new node's join request to the proximally nearest node of existing Pastry network.
- Maximal prefix matching of the node ID with the requested node's node ID in the leaf set.

1. If found in leaf set
    A. Create an entry of new node.
    B. Send the join request to all the nodes present in the leaf set.
2. Else if found in Routing Table
    A. Search in routing table based on prefix.
    B. Create an entry of new node in matched row.
    C. Send the join request to all the corresponding nodes in the matched prefix row.
3. Else if not found anywhere
    A. Create an entry in neighbourhood set (assumption).
    B. Send the join request to every node present in neighbourhood set.

- Send the routing table to the neighbours.

Our Implementation

Data Structures

1. **unordered_map<string, string> distributedHashTable;**

   // structured database where the key_ID serves as a key and uniquely identifies the value across the network.

2. **struct routingTable {**

   vector< pair<string,int> > neighbour_set;

   // It is a vector of pairs that stores the node IP and the corresponding port number.

   vector< pair<string,int> > leaf_set;

   // It is a vector of pairs that stores the node IP and the corresponding port number.

   map<string,vector<pair<string,int> > >routing_set;

   // map that stores pair of node IP and port at the key of map which matches with prefix of nodeID.

   };

Base Assumptions:

1. 4 Byte node ID
2. 6 Byte key

Functionalities

1. **Port <X>**

   Creating a node which has its listening port as X.

2. **Join <Node ID> <IP> <Port>**

   Joining a new node to the existing network that has Node ID, IP and a listening Port.

3. **Put <key> <value>**

   Store the key value pair with matched prefix Node ID.

4. **Get <key>**

   Retrieve the key value pair with matched prefix Node ID.

Functionalities

5. lset

Prints the leafset of current node

6. rset

Prints the routing table of current node

7. nset

Prints the neighbourhood set of current node

https://github.com/rishabhmurarka7/Pastry
https://github.com/rishabhmurarka7/OS-Project

"Thankyou!"

– 4 Nodes