Peer-to-Peer Hash Table

Leyang Li, Zanxiang Yin

High-Level Description

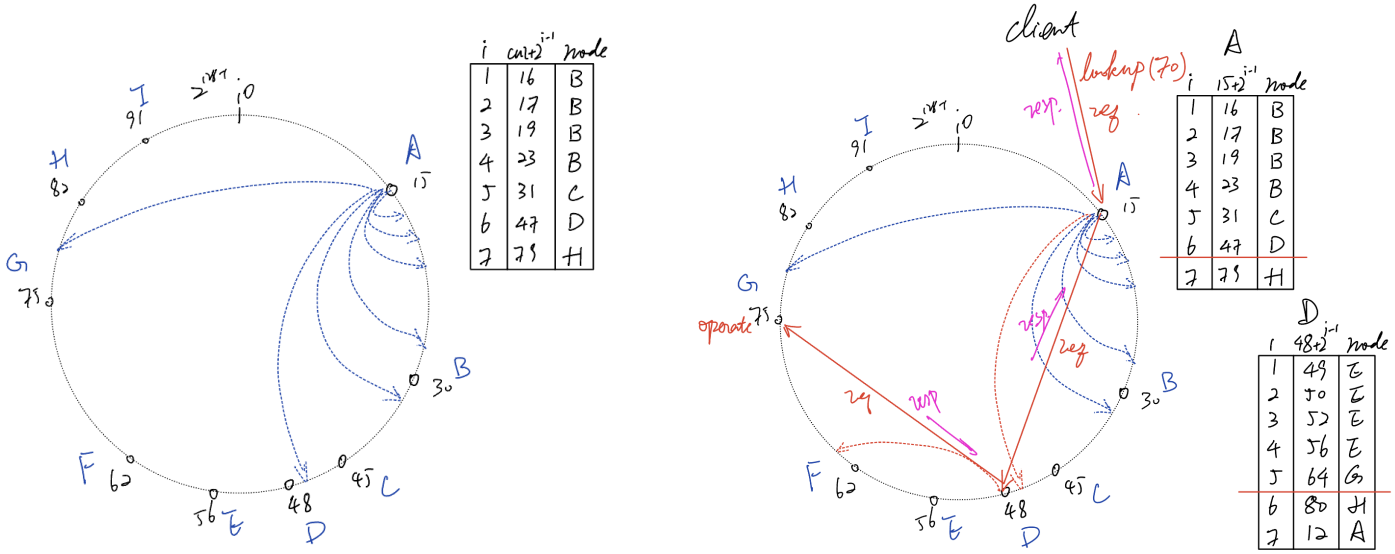
The goal of this project is to expand on the existing distributed spreadsheet assignment (with operations: insert, lookup, remove, size, and query) to scale up to a large size.

This is expected to be achieved by designing and implementing a distributed hash table (DHT) following the peer-to-peer (P2P) architecture based on the Chord protocol.

The system should support:

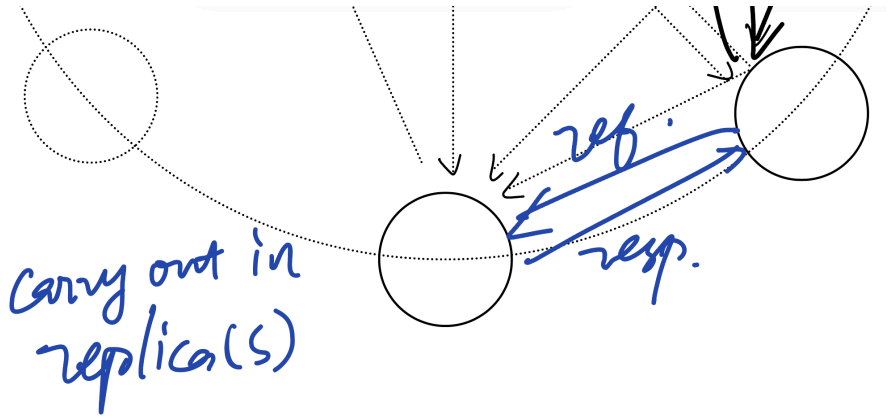
1. **O(logN) time Lookup**: The Chord protocol ensures O(logN) time key lookups. *N* is the number of nodes.
2. **Node Joins and Departures**: Updating finger tables and the predecessor link.
3. **Data Preservation**: Implementing data replication mechanisms to handle node failure.

Structure of the System



Chord Protocol: using finger tables to enable O(logN) time key lookups

Replication Mechanism: During each operation, the node sends the copy of the operation to node(s) where the data's replica resides. The replica node(s) then carry out the same operation for consistency.



Key Distributed Systems Problem

- Ensuring **low-latency** lookups for a large number of servers.
- Ensuring data availability and system consistency of nodes dynamically **joining and leaving**.
- Ensuring data persistence by **replications**, log and checkpointing.

Technologies, Languages, and Resources

- **Languages**: Using **Python** for core networking and distributed system functionality.
- **Networking Libraries**: **Sockets** will be used for peer-to-peer communication.
- **Distributed Hashing**: Implement **consistent hashing** to ensure efficient data distribution.
- **Threading/Concurrency**: Use Python's **threading library** for handling node operations.

Plan for Evaluating the System

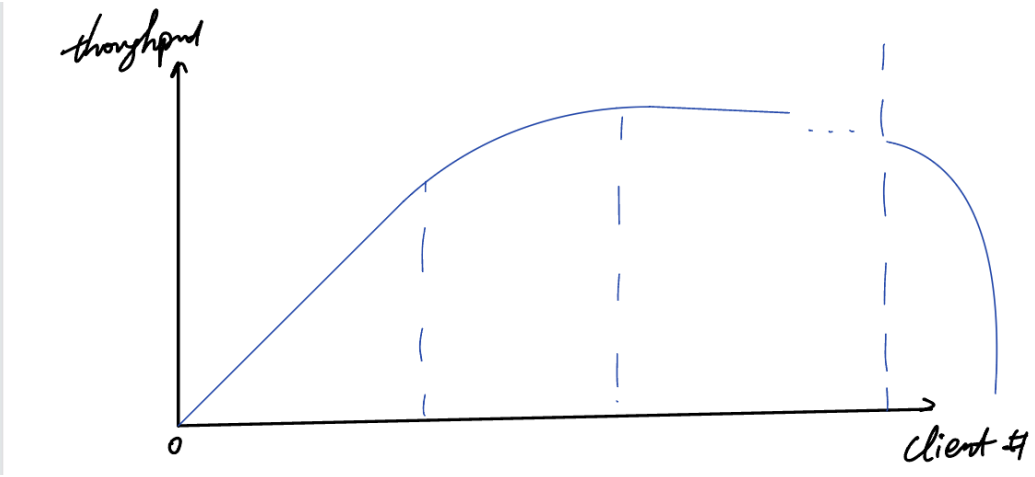
The system will be evaluated based on:

1. **Throughput**: Calculating the number of operations of lookups, insertions, and removals it can handle per second.
2. **Latency**: Time taken to perform a lookup, insertion or removals.
3. **Scalability**: Evaluate system performance as the number of nodes grows.
4. **Fault Tolerance**: Test how operation time scales with increasing data load.

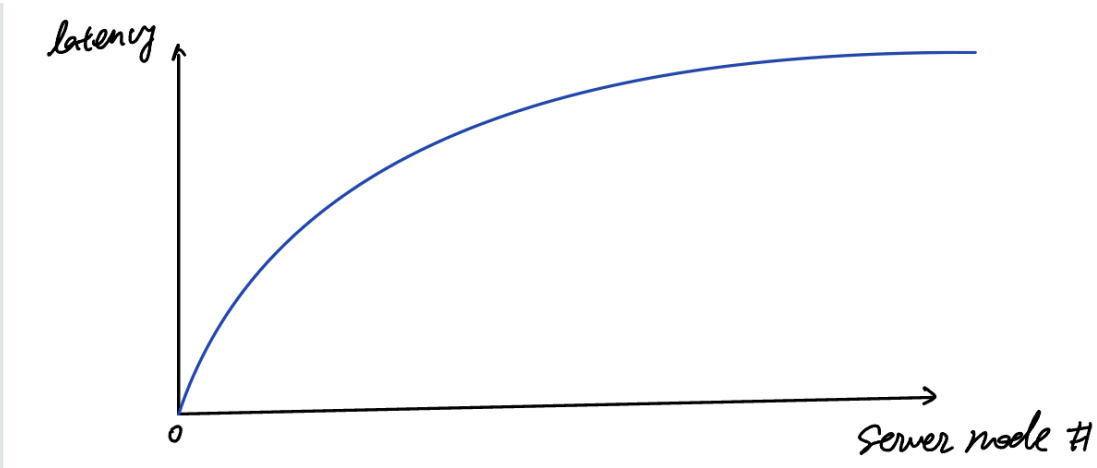
Notional Metrics and Expected Results

- **Throughput**: Expected to increase with system size (up to a certain point) as more nodes handle the load, but diminishing returns will occur due to overhead.

- Notional graph: Initially shows linear growth, then flattens as overhead increases.



- **Latency**: The system is expected to scale logarithmically (O(log N)) as nodes increase, with minor increases during node failures or departures.
    - Notional graph: O(logN) growth.



Conclusion

This project will provide a deep dive into designing a fault-tolerant, efficient, and scalable peer-to-peer hash table using distributed systems principles. By implementing and evaluating the system's key features such as lookup efficiency, fault tolerance, and scalability, we will gain insights into the challenges of decentralized data storage and management.