CS514 Network Final Project

Title: Design and Implementation of a Chord-based Peer-to-Peer File-Sharing System

Fengyi Quan (fengyi.quan@duke.edu)
Zhangfan Li (zhangfan.li@duke.edu)
Yuxin Song (yuxin.song@duke.edu)

Abstract

The exponential growth of digital content has necessitated the development of robust, scalable, and efficient file-sharing systems. Traditional client-server architectures often face issues like central points of failure, scalability bottlenecks, and uneven bandwidth distribution. A peer-to-peer (P2P) file-sharing system addresses these challenges by decentralizing the distribution of files, enhancing reliability, and ensuring a more equitable use of resources.

Chord is a protocol and algorithm for a peer-to-peer distributed hash table (DHT) that facilitates the assignment of keys to nodes and the data location within a dynamic P2P network. For this project, we decided to implement a Chord-based P2P file-sharing system aimed to leverage the advantages of P2P networks with the robustness and efficiency of the Chord protocol for data location.

Introduction

In this section we will briefly introduce the Chord algorithm, which offers an elegant solution to the challenge of efficient key-based lookups in decentralized peer-to-peer networks. In this algorithm, each node within the network is assigned a unique identifier derived from the hash of its IP address. These identifiers are organized in a circular identifier space, forming a virtual ring that serves as the foundation for Chord's decentralized architecture. Simultaneously, data keys are hashed to produce identifiers and are placed on the same circle. Each node is then responsible for a specific range of keys, with responsibilities distributed in such a way that each node manages the keys falling between its own identifier and the identifier of its successor.

To optimize lookup operations, Chord introduces the concept of a finger table which is a data structure maintained by each node. This table contains entries pointing to other nodes in the Chord ring, facilitating quicker key lookups. When a node wishes to find the successor of a particular key, it consults its finger table. If the key falls within the current node's range of responsibility, the search concludes, and the node is identified as the successor. Otherwise, the node uses its finger table to forward the request to a node that has the largest identifier less than the target key. This process continues iteratively until the responsible node is found.

Chord accommodates dynamic changes in the network, such as nodes joining or leaving, through efficient mechanisms. When a new node joins, it updates existing nodes and adjusts finger tables, minimizing communication overhead. Conversely, when a node exits, its keys are transferred to the successor to maintain data integrity. The algorithm employs periodic stabilization routines to ensure that successor and predecessor information remains current, addressing the dynamic nature of large-scale distributed systems. In essence, Chord's design not only facilitates scalable and fault-tolerant distributed systems but also enables efficient and decentralized operations crucial for various applications.

Project Objective

The primary objective of this project is to design and implement a Chord-based P2P file-sharing system. The project will consist of two main components:

1. The Basic P2P File Transfer System: This foundational system will enable file sharing directly between peers without central coordination.
2. The Chord-based System: This advanced system will incorporate the Chord protocol to manage a distributed network of nodes, allowing for more efficient file lookup and retrieval.

Implementation

We decided to use gRPC to handle the communication between nodes in the Chord network as well as file transportation. We assume that the new node learns the existence of node n' by some external mechanism.

1. Node identifier: Each node in the Chord network is assigned a unique identifier based on the hash of its IP address and port number. These identifiers are typically represented as points on a circle, forming a circular identifier space. It is using m-bit identifier using SHA-1 for simplicity. For better performance, it can use universal hash functions and choose m large enough to avoid the hash collision. These parameters are customizable in the code and designed to change easily.

2. Node join: When a new node joins the network, it needs to inform the existing nodes and update the finger tables to maintain the integrity of the Chord ring. This process is generally efficient and involves minimal communication.
3. Lookup Operation: When a node in the Chord network wants to find the successor of a particular key, it starts by checking its own finger table. If the desired key falls within the range of the current node's responsibility, the search is complete, and the current node is the successor. If the key is not within the current node's responsibility, the node uses its finger table to forward the request to the node in the finger table that has the largest identifier less than the target key. This process is repeated until the responsible node is found.
4. Stabilization: The Chord ring is self-stabilizing, keeping nodes' successor pointers up to date, and telling the successor about itself, which is sufficient to guarantee the correctness of lookups. If a node leaves the ring, the current successor does not exist. It will not actively find a new successor, fixed fingers will tell the current node the correct successor. Fix fingers will periodically refresh finger table entries. The program starts two background processes to perform stabilization and fix fingers.
5. File sharing: Users can upload their content to the Chord ring, and it is stored on another node based on the calculation of the hash in the ring. This ensures that the content remains available even if the original owner leaves. Users can choose to host other content when they are online, contributing to the resilience and availability of shared files. Files also move to a new responsible node when the original responsible node joins or leaves the ring. Since file hash is calculated based on the file name, we assume file names are globally unique without loss of generality. The files are stored under resources/<port>.
6. File downloading: Each node, in addition to its responsibilities for key ranges, is equipped with the ability to manage and serve files associated with specific keys. When a node receives a file download request, it leverages the Chord algorithm to locate the node responsible for the corresponding key and initiates a file transfer.
7. Soft Exit: When a node leaves the ring, it will notify its successor to take over the files that the node is hosting. The successor will update its predecessor pointer to

the node's predecessor. It will also update all fingers that reference to self. This functionality is to update finger tables as soon as possible to reduce the work performed in stabilization.

8. gRPC server: gRPC server is running as a daemon thread by default. It handles all necessary operations in the chord algorithm as well as file sharing protocol. For more information and API calls, please refer to chord.proto file. All rpc calls related to files use steaming to avoid large files and divide them into pieces for fast transmission.

Conclusion

This Chord-based P2P file-sharing system project will lead to the development of a decentralized and efficient digital content distribution platform. It will cater to the growing need for scalable file-sharing solutions and foster research and development in distributed systems and networking.

The proposed budget and timeline ensure a streamlined development process, with adequate room for testing and refinement. This initiative will not only meet the immediate file-sharing needs but also contribute to the academic and practical understanding of P2P networks and DHTs.