Paper Review: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Sohail Ahmed Shaikh

January 21, 2019

1 Summary:

Through this paper the authors wish to provide a solution of locating a node that stores a data item. The result is a protocol called Chord which essentially provides a lookup through a key which is mapped on to a node. At the heart of the protocol is consistent hashing. The authors, through proofs and experimental results show the scalability of Chord.

2 Description:

A fundamental attribute in any peer to peer system is decentralized control and absence of hierarchy. The core of many operations in such a system is to locate data items effectively. The Chord protocol attempts to effectively address this in a dynamic environment where nodes frequently arrive and depart the system. So in essence the function of Chord protocol is quite simple: given a key, give the IP address of the node this key is mapped to.

In it's core, Chord uses consistent hashing as it has properties that act as a natural load balancer and each node receives roughly the same number of keys. The idea behind consistent hashing is determining the successor of a key. This leads to an important advantage that when a node enters or leaves the system only a small fraction of keys (or only the keys directly involved) are affected (with high probability).
Chord improves upon the scalability limitation of consistent hashing by altering it so that a Chord node knows the "routing" information of only a few other nodes. The routing table maintained by the system is itself distributed and routing is done by communicating with other nodes either iteratively or recursively. This leads to a system in which each node mantains information of up to logN other nodes if there are N nodes in the system.
Chord uses SHA-1 hashing and cracking it is an NP hard problem. The authors use this to justify security guarantees when keys can be randomly selected.

To reduce the number of hops to reach a node, each node maintains a routing state consisting of predecessor pointer and finger table. Finger table is essentially a list of successors of the current node and maps this to the IP address of the node.
The authors prove that this network model, with high probability provide a guarantee that the number of nodes that must be contacted to find a successor in an N node network is O(logN).
The authors also describe the working of the system when nodes are joined and when they depart and how the routing states are maintained. The authors prove that this takes on an average $O(log^2 N)$ time.

A stabilization protocol is also implemented in this system which guarantees that the finger table of

all nodes are maintained and the lookups are fast as well as correct when nodes are added to Chord. This is run periodically so that the network is stable and the system has eventual consistency and correctness.

The authors further elaborate upon failure scenarios and how the performance degrades gracefully when the information is out of date.

3 Strong points:

1) Chord protocol provides a good base for implementing many distributed application which would leverage key-value stores.
2) Authentication, caching and replication are easy to implement by applications on top of Chord because of it has a flat key space.
3) Simplicity: The routing protocol and hashing mechanism is quite simple and intuitive to implement.
4) Provable performance and correctness: The authors have given substantial proofs for the performance and correctness of Chord as well as results from exprements.

4 Weak points:

1) It does not consider the physical locality of nodes. Hence delays could be substantial. Lookup can be very unoptimized in this case. Also, network delays are not accounted for. This is a causal effect of not considering locality of nodes.
2) The implementation of Chord is done in an iterative style. However, the choice of this style over the recursive implementation is not elaborated and justified.
3) Stabilization algorithm used in Chord doesn't handle a system that has multiple disjoint cycles or network partitions.
4) The authors decribe periodic sampling of the ring topology as a solution to handle network partitions and stability. This may cause a lot of overhead and may not scale well. Also, it's performance is not described and evaluated
6) Performance is affected by the frequency of node joins and leaves. If there are frequent joins and leaves the network will be unstable. As a solution if frequency of the stabilize protocol is increased, then there will be a lot of overhead in stabilizing the system and scalability will be adversely affected. A problem with commodity hardware would be if nodes continue joining and leaving constantly, the eventual consistency guarantee may no longer be true.
7) Malicious nodes can affect routing and disrupt a global view of the system.

5 Improvements:

1) The cause of multiple disjoint cycles or partitions in this system must be evaluated. Effective mechanisms for dealing with this must be evaluated.
2) As another improvement, instead of randomly assigning keys to nodes, some determinstic method that accounts for physical locality of nodes can be evaluated, for example instead of completely moving away from a hierarchical system like DNS some of the hierarchy could be incorporated to have some locality.
3) To handle the problem of malicious nodes multiple routes to node can be considered to verify consistency.