Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications

Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek and Hari Balakrishnan

Abhishek Srivastava
Student ID: 861307778

May 8, 2017
CS 204, Spring 2017

Review:

This paper implemented a scalable peer-to-peer efficient lookup algorithm which overcomes the existing lookup problems. The *Chord* protocol works by providing it a key and it maps the key to a particular node. The node mapped may store value which is associated by the given key. *Chord* protocol uses consistent hashing to assign keys to the available nodes. The reason for using consistent hashing is it balances load by assigning each node almost equivalent amount of keys and also it is robust to the nodes leaving and joining the system. *Chord* extended the shortcoming of consistent hashing by using "routing" information of few other nodes. In *Chord* protocol participating nodes communicate with each other nodes while performing lookup. Therefore in N-node system all lookups are resolved in $O(logN)$ and each node maintains information of about $O(logN)$ other nodes. *Chord* performance gets degraded by state routing information because nodes join and leave in arbitrary fashion.

The paper also discusses multiple other peer-to-peer systems and compares them how they are different to *Chord* protocols. Few systems they compared were $Freenet$, $Ohaha\ Systems$, $Globe\ Systems$, $Plaxton\ Protocol$, CAN, GLS, $Napster$ and $Gnutella$.

The design principle for *Chord* protocol was to solve the problem such as: **Load balance** by spreading keys evenly over all the nodes, **Decentralization** by making it distributed, **Scalability** by making it feasible for very large systems as well, **Availability** by adjusting internal tables so that it can be robust to the node failures and **Flexible naming** by putting no constraints on structure of the keys for look up. *Chord* behaves like a library which provide functionality like lookup function which takes input a key and return an IP address for the node mapped to that key and notifying application changes in the keys which it is responsible for. Application using *Chord* also provide authentication, caching, replication and renaming of data. *Chord* provide good foundation for the application such as: cooperative mirroring, time-shared storage, distributed indexes and large-scale combinatorial search. *Chord* protocol Specifies how to find the locations of keys, How new nodes join the system and How to recover from the failure or planned departure of existing nodes.

.

Comments:

- This Paper improved on lot of problem by peer-to-peer systems by proving lookups in $O(logN)$ nand robust to node failures. The part which i thought is pretty good is *Chord* protocol can be extended to File system and also can be used as DNS lookup.