Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications by Stoica , Morris, Karger, Kaashoek, Balakrishnan

CSC 724 Paper review - Vaibhav Singh, vsingh7

January 21, 2019

1 Summary

The paper presents Chord, a distributed data lookup protocol in which nodes need to maintain membership information about a small amount of other nodes(O(logN)). The consistency degrades gracefully if this condition is not maintained. The paper also talks about Chord's stabilization algorithm which ensures finger tables are continuously updated, thereby ensuring nodes have knowledge of the state of the system, and also addresses the issues of replication and load balancing.

2 Description

The base Chord protocol uses a primitive called consistent hashing which basically assigns each node and key an m-bit identifier taking the node's IP address as input, and using a base hashing function like SHA-1. The nodes are then imagined to be arranged in order along a circle of numbers from 0 to 2 pow(m) - 1 (m is a number large enough to ensure adequate load balancing). The nodes need to keep information about successor of n + 2pow(i)th node, where i goes from 1 to m. If a new node joins the system, some keys assigned to the new node's successor get assigned to the new node. When a node n leaves the network, all nodes assigned to that node get assigned to its successor. In both cases, O((log pow 2)N) messages are required for every node to re-establish updated finger table state.

The stabilization algorithm in Chord looks for its successor's (n) predecessor p, and decides whether p should be its successor instead (for when p recently joined the system). It also tells p of the presence of n to help initialize p's finger table.

When a new request comes to any node, the nodefirst checks if the requested data is present in that node, then the node compares the hash of the key with its own finger table, and sends the request to the closest node smaller than the hash in its finger table, which in turn does the same until the data is either found or the request fails.

3 Strong Points

Chord spreads keys evenly among nodes, thus providing automatic load balancing.
Chord lookup cost grows as log of number of nodes, thus providing scalability.
No node is more important than the other nodes, which provides Decentralization and improves fault tolerance
As nodes continuously run stabilization algorithms to update their finger tables to take into account nodes which joined or left recently, membership problem is handled adequately.

4 Weak Points

O(logN) lookup time might still be quite high.
Chord does not take into account the geographic location of nodes, so nodes which are geographically located close to each other may be placed far apart in the ring and vice versa.
Stabilization time O(Npow2) is quite high.

5 Improvement

Chord can be improved to detect and heal partitions in rings.
A malicious or buggy set of participants can present an incorrect view of the chord ring leading to denial of service.
Log(N) messages per lookup could be improved by creating finger tables with distance 1 + 1/d instead of using powers of 2. This will lead to increase in number of finger table entries by a factor of d though.