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

Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, Hari Balakrishnan
CS344G class presentation

February 4, 2015
The landscape

Goal
Distributed lookup table

Existing/competing implementations

- DNS
- Freenet storage system
- Plaxton protocol
- CAN
Features

• Load balance
• Decentralization
• Scalability
• Availability
• Flexible naming
Features

- Load balance
- Decentralization
- Scalability
- Availability
- Flexible naming

Potential applications

- Cooperative mirroring
- Timeshared storage
- Distributed indices
- Distributed computation
Chord protocol

Assume keys are mapped to a particular node

Components

- Finding location of keys
- Handling nodes joining/exiting system
- Handle state inconsistencies/suboptimality
Location of keys
Main idea
Consistent hashing
Location of keys

Main idea
Consistent hashing; requires global information
Location of keys

Idea
Consistent hashing with finger tables

Provides scalable key location with $O(\log N)$ entries in finger table
Node joining/leaving

Invariants

- Each node’s successor is correctly maintained
- For every $k$, $\text{successor}(k)$ is responsible for $k$
Node joining/leaving

Invariants

- Each node’s successor is correctly maintained
- For every $k$, $\text{successor}(k)$ is responsible for $k$

Theorem

Any node joining/leaving the network can use $O(\log^2 N)$ messages to establish the invariants and finger tables
Some other considerations

- Stabilization
- Concurrent joins/failures
Performance

Load is somewhat unbalanced without virtual nodes
Performance

Load balancing with virtual nodes
Performance

Path length of lookup query
Performance

Simultaneous node failure
Performance

Lookup failure due to state inconsistency