

CS 518

Chord and CFS: Internet-Scale Scalable Object Lookup

Randy Wang
Fall 2002
Princeton University

Outline

- Chord: scalable lookup
- CFS: file system on top of Chord
- Remarks

CS518

1

Randy Wang

Outline

- **Chord: scalable lookup**
- CFS: file system on top of Chord
- Remarks

CS518

2

Randy Wang

Object Lookup Abstractions

Obj. id.	machines	hash(Obj. id.)	machines
0000	bolle	00	bolle
0001	willy, tux	01	willy
...
ffff	opus	ff	opus

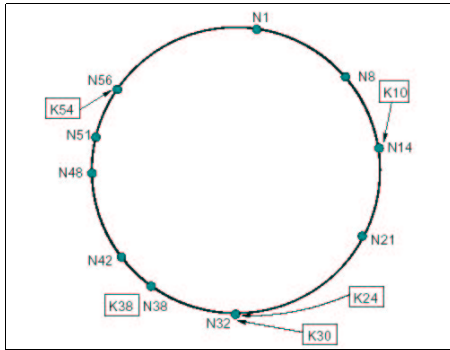
- Per-object lookup
 - Napster, AFS, Sprite: centralized
 - SVM, xFS: distributed (plus a level of indirection)
 - Object space big: big table
- Hash table lookup
 - Smaller table
 - Naive implementation still no good
- Key scalability issues
 - Per-machine state
 - Number of network hops per lookup
 - Amount of distributed state change
 - Load-balancing

CS518

3

Randy Wang

Chord ID Space



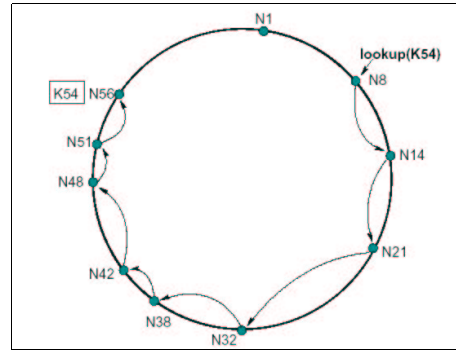
- Hash both machine IDs and data objects to the same object ID space
- Object is located at machine N if N is the immediate successor node in the ID space
- Random hashing should distribute the objects evenly across the machines
- Question: how do we find the successor machine?

CS518

4

Randy Wang

Naive Routing



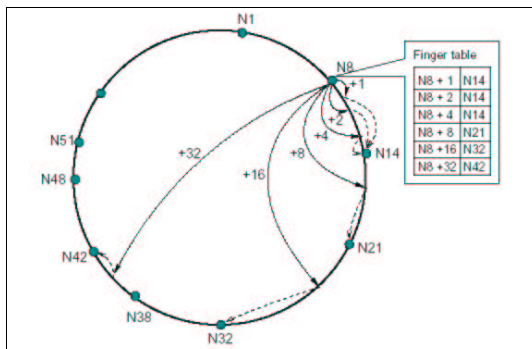
- Each machine points to its unique successor
- $O(1)$ routing state, $O(N)$ lookup hops

CS518

5

Randy Wang

Better Routing



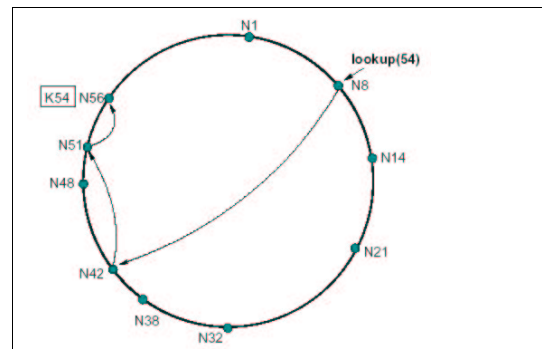
- Each machine n points to more machines around the ID ring in a "finger table"
- $\text{finger}[k] = \text{successor}(n + 2^{k-1})$

CS518

6

Randy Wang

Better Routing



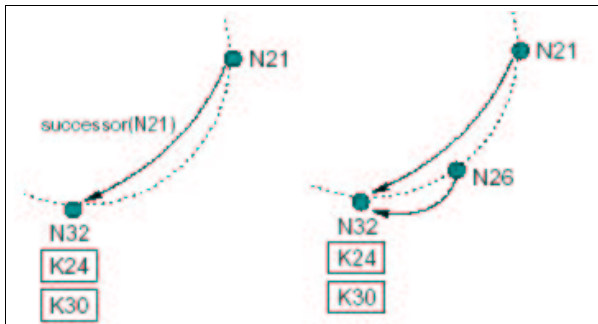
- Each machine n points to more machines around the ID ring in a "finger table"
- $\text{finger}[k] = \text{successor}(n + 2^{k-1})$
- Each machine routes to the finger table machine that is closest to the target
- Worst case: each hop cuts the distance by 1/2
- $O(\log N)$ routing state, $O(\log N)$ lookup hops

CS518

7

Randy Wang

Machine Joining



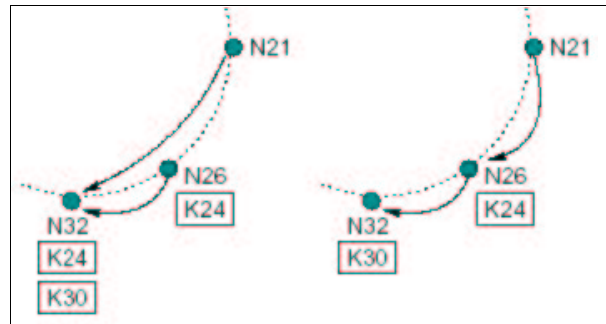
- Joining machine finds its successor machine and points to it

CS518

8

Randy Wang

Machine Joining (cont.)



- Joining machine finds its successor machine and points to it
- New machine copies the data objects it should keep from its successor
- New machine's predecessor points to the new machine
- Can still prove $O(\log N)$ lookups even in face of concurrent joins and departures

CS518

9

Randy Wang

Chord Properties

- Key scalability issues
 - Per-machine state: $O(\log N)$
 - Number of lookups: $O(\log N)$
 - Amount of distributed state change: localized
 - Load-balanced

CS518

10

Randy Wang

Outline

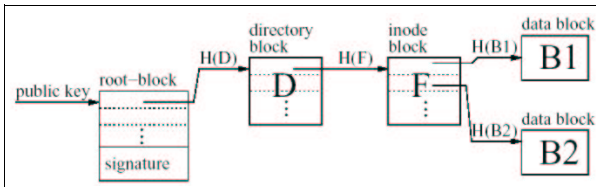
- ~~Chord: scalable lookup~~
- **CFS: file system on top of Chord**
- Remarks

CS518

11

Randy Wang

CFS: Making a File System Out of a Chord-Based Block Store



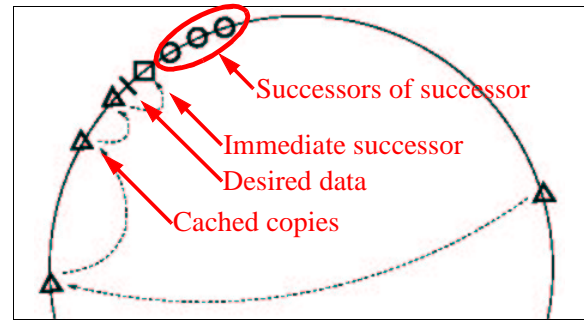
- Metadata and data blocks treated identically and stored at hash locations determined by Chord
- Write: publicize new versions of a chain of blocks all the way up to the root block
- Read: start chasing metadata blocks at the root to avoid accessing stale copies
- No deletion
- No concurrent writes: followup paper (Ivy)

CS518

12

Randy Wang

Replication and Caching



- Replica locations determined by hashing of destination
- Cache locations determined by access patterns of the requester (as well as hashing of destination)
- Improves
 - Availability
 - Latency/locality

CS518

13

Randy Wang

Outline

- ~~Chord: scalable lookup~~
- ~~CFS: file system on top of Chord~~
- Remarks

CS518

14

Randy Wang

Lookup Scalability Ain't Free

Obj. id.	machines	hash(Obj. id.)	machines
0000	bolle	00	bolle
0001	willy, tux	01	willy
...
ffff	opus	ff	opus

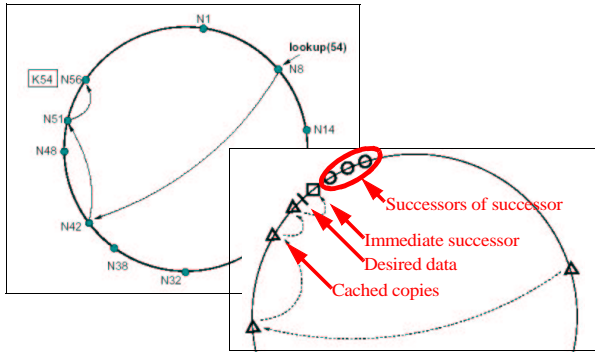
- Per-object lookup table vs. hash table lookup
 - Loses placement flexibility to the hash function
 - Loses ability to track cached copies
 - Node joining and leaving necessitate data redistribution
- You wouldn't want to use this unless you need extreme scalability of lookups

CS518

15

Randy Wang

Loss of Placement Flexibility



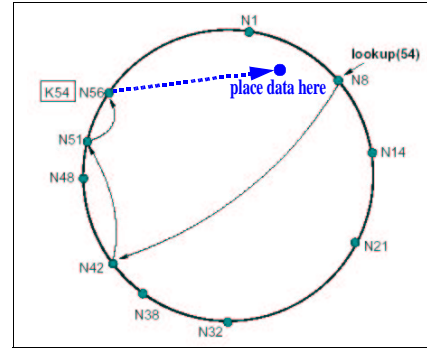
- Hashed placement = fixed random placement
 - Data with same hash values cannot be dealt with differently
 - log N hops (even the first hop might be expensive)
- Possible “fix” #1: Chord locality enhancements come at the cost of replication
 - The more choices for locality, the more replicas are needed, needs not just extra capacity but also b/w

CS518

16

Randy Wang

Loss of Placement Flexibility (cont.)



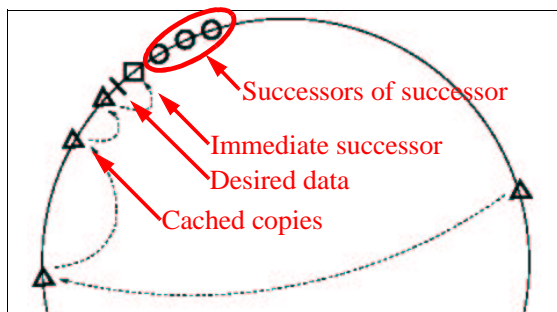
- Possible fix #2: introducing a level of indirection may improve large data transfer
 - It does not improve latency

CS518

17

Randy Wang

Loss of Ability to Track Cached Copies



- Why track cached copies?
 - Consistency: need to invalidate or update them if we were to deal with writes
- Cached copies are intrinsically at non-hashed locations
- Possible fix #1: disable caches? hurts performance
- Possible fix #2: “leases”:
 - Short lease: too much lease renewal traffic
 - Long lease: hurts invalidation/update latency

CS518

18

Randy Wang