Chord

A DHT lookup service

A little history!

The problem of lookup in DHT

- Where should node A store data item D?
- How do other nodes e.g., node B, discover the location of D?
- How can the distributed system be organized to ensure scalability and efficiency?

Conventional systems

- Central server based approach (Napster)
- Flooding search technique (Gnutella)

Comparative Analysis

	Memory	Lookup Latency	#Messages for a lookup
Napster	O(1) (O(N)@server)	O(1)	O(1)
Gnutella	O(N)	O(N)	O(N)

Why Chord?

Chord

Chord is a protocol and algorithm for a peer-to-peer distributed hash table. It is one of the four original distributed hash table protocols, along with CAN, Tapestry, and Pastry. It was introduced in 2001 by Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan, and was developed at MIT.

Chord uses the concept of consistent hashing for maintaining load balancing features.

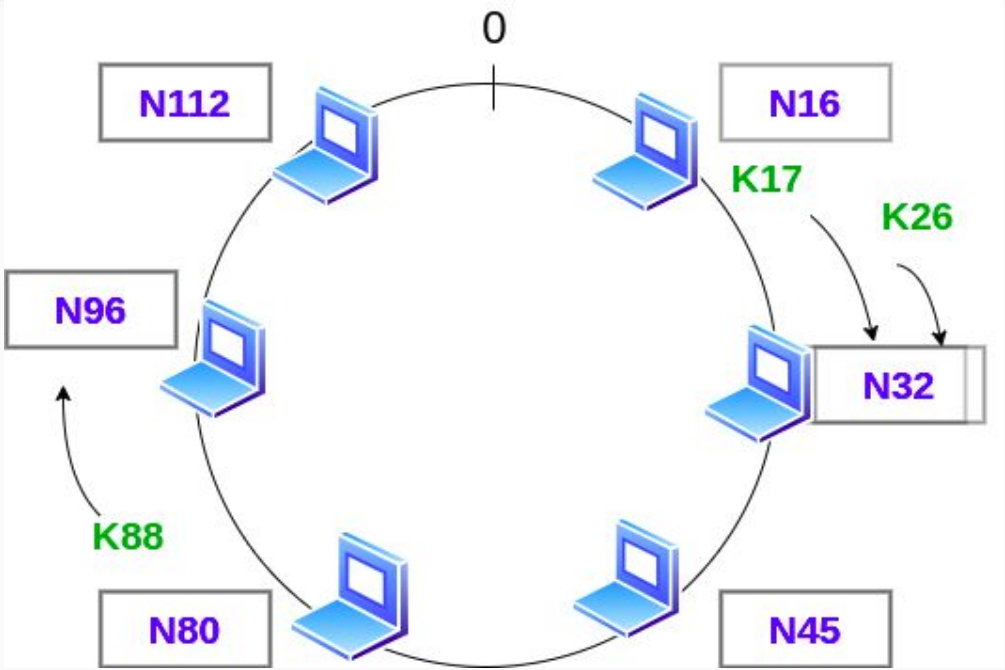
Consistent Hashing

- A distributed hashing scheme
- Independent of the number of nodes in the network
- Works by assigning a position to the nodes on an abstract circle or hash ring

Consistent Hashing

- Each node is assigned an ID.
- All the IDs are mapped to an imaginary circular space
- All the keys of the data are also mapped to the same circular space
- All the keys are assigned to the nearest node having ID greater or equal to the key ID.

Consistent Hashing



This example uses 128 modulo encoding to create the hash ring

Chord - Finger Table

Every node maintains a routing table knows as Finger Table.

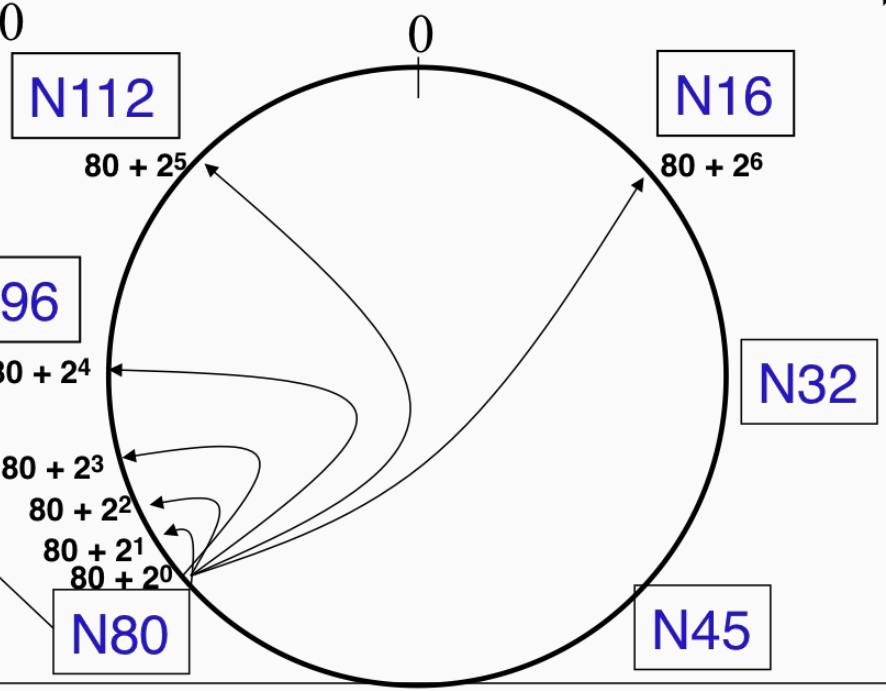
The **i**th entry in the finger table contains the identity of the first node that succeeds the current node by at least $\mathbf{2}^{(i-1)}$ on the identifier circle

- Each node stores information about only a small number of other nodes, and knows more about nodes closely following it on the identifier circle than about nodes farther away.
- A node's finger table generally does not contain enough information to directly determine the successor of an arbitrary key k.

Say $m=7$

Finger Table at N80

i	$ft[i]$
0	96
1	96
2	96
3	96
4	96
5	112
6	16



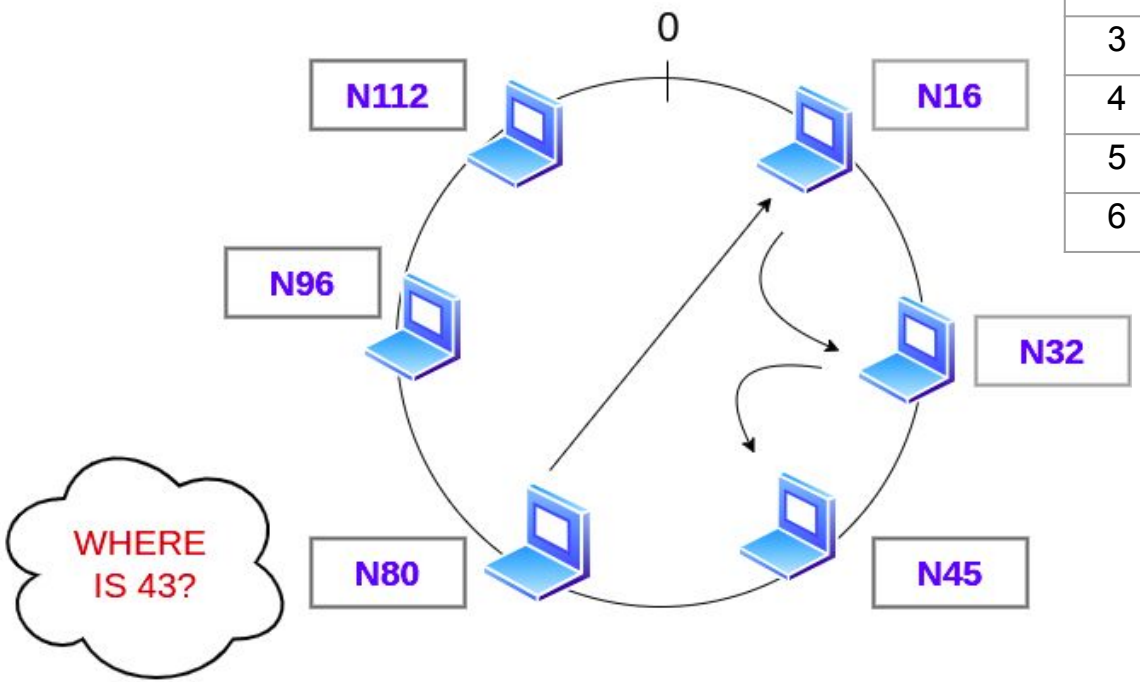
ith entry at peer with id n is first peer with id >= $n + 2^i \pmod{2^m}$

Operations - Search

1. Tries to find the successor the queried key.
2. The querying node searches in its finger table first.
3. If not found, the highest predecessor of the queried key is found in the finger table and the query is passed on to that node.
4. Step 2 is repeated until the query data is found or the whole circle is traversed. In that case, a miss is registered.

Example : Search

i	ft(i)
0	32
1	32
2	32
3	32
4	32
5	80
6	80



Challenges

- New Peers join
- Peers leave
- Peers fail

Operations - Join

1. Entry to a chord network is done via a known node of the network n’.
2. The new node asks n’ to find the immediate successor of the new node.

Operations - Stabilize

Every node performs **stabilize** periodically to learn about newly joined nodes. Each time node ***n*** performs stabilizations, it asks its successor for the successor's predecessor ***p***, and decides whether ***p*** should be ***n***'s successor instead. This would be the case if node ***p*** recently joined the system. In addition, **stabilize** notifies node **n**'s successor of **n**'s existence, giving the successor the chance to change its predecessor to **n**. The successor does this only if it knows of no closer predecessor than **n**.

Operations - Fix fingers

Each node periodically calls f**ix_fingers** to make sure its finger table entries are correct. This is how new nodes initialize their finger tables, and it is how existing nodes incorporate new nodes into their finger tables.
Each node also runs **check_predecessor** periodically, to clear the node's predecessor pointer if **n.predecessor** has failed; this allows it to accept a new predecessor in **notify**.

Properties of Chord

1. Autonomy and decentralization
2. Fault Tolerance
3. Load Balancing
4. Scalability

Comparative Analysis

	Memory	Lookup Latency	#Messages for a lookup
Napster	O(1) (O(N)@server)	O(1)	O(1)
Gnutella	O(N)	O(N)	O(N)
Chord	O(log(N))	O(log(N))	O(log(N))

THE END

Devesh Jalan	001710501071
Md Sahil	001710501029
Priyank Lohariwal	001710501055

Final point

A one-line description of it

Chord bawaal

“Sona de warna ,
Sab ka ghar jala denge”

- Devesh Jalan

*Arpan ka bhi

This is the most important takeaway that everyone has to remember.

Thanks!

Contact us:

The Black Hole
Google meet
DM for link

please_no@maam.com
www.give_us_s.com