Consistent Hashing

Background

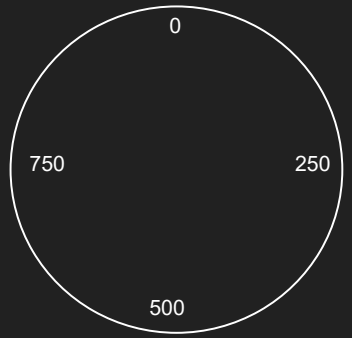
In situations where we are trying to uniformly and randomly distribute load over nodes in a cluster, consistent hashing is a way of doing so while also minimizing the amount of reshuffling previously hashed items when adding or removing nodes from the cluster.

Useful in:

- Sharding
  - Minimal rebalancing of partitions leads to reduced stress of the network
- Load balancing
  - Having the same requests to the same servers can allow for better cache performance

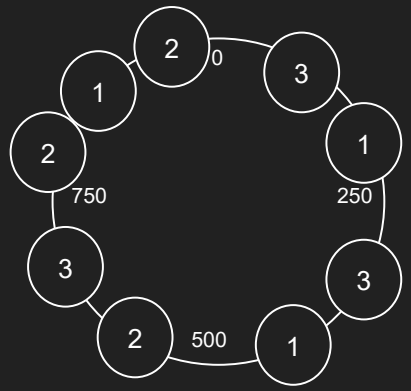
Consistent Hashing Overview

- Pick any good hash function, and imagine its range
  - For the purposes of this video let’s imagine the hash function only returns outputs from 0-999
  - Visualize the range as if it was a ring



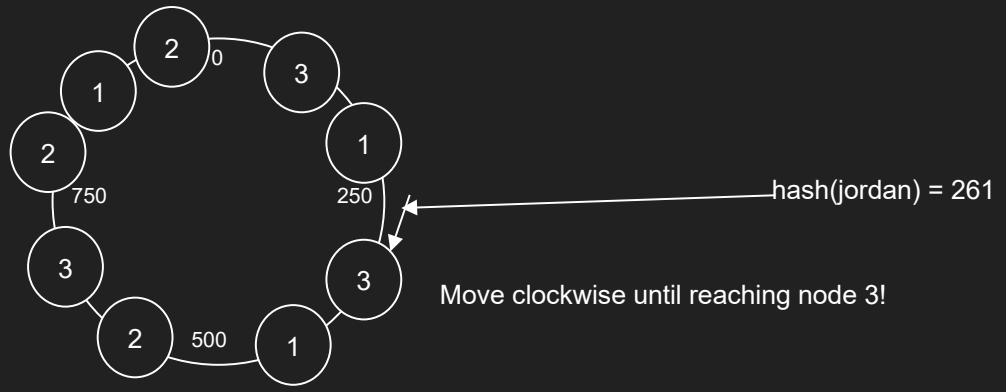
Consistent Hashing Overview

- To add nodes to a cluster, we add it to a random place in the ring (removing is the exact opposite)
  - In reality, we actually tend to add it to a few places in the ring
  - This helps us to make sure that new nodes are taking load relatively evenly from other nodes, as opposed to just one



Consistent Hashing Overview

- To determine where to place a given item
  - First take the hash of it
  - Then move around the ring clockwise until reaching one of the node circles, and send it to that node



Consistent Hashing Summary

Solid algorithm for distributing load evenly and trying to make sure that even on new membership or exits from the cluster, the majority of the items stay in the same place while still being evenly distributed.

Used in various load balancers, and partitioning methods (we see this in Cassandra which is why I’m talking about it now).