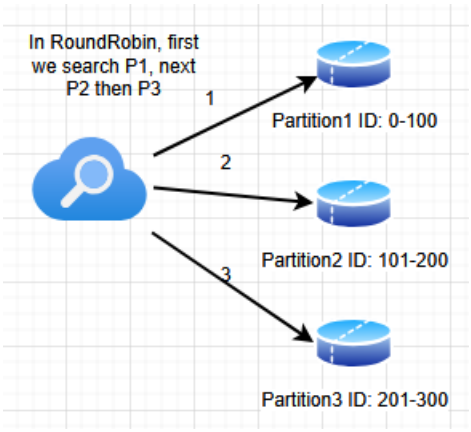
**Partitioning3:**

Re-balancing partitions:
We need to re-balance partitions in the following conditions:
- Add more machines
- Add more disks
- Replace machines
- Skewed + hotspot partitions

Request Routing:
- Reading mechanisms
Our problem statement is, we have to read from multiple partitions

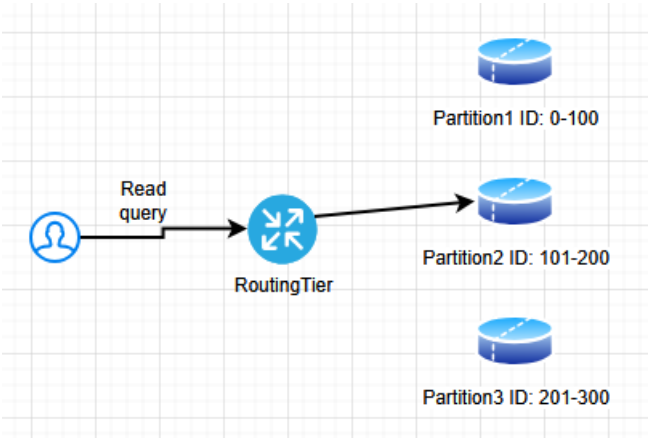
Approaches are Round Robin,

Round Robin:




We are checking each and every partition in round-robin manner and once we find the data we stop searching. Time complexity is O(n). It will wait after searching one partition before moving to the next partition.
Demerit is Time complexity is high, latency because we go into all partitions

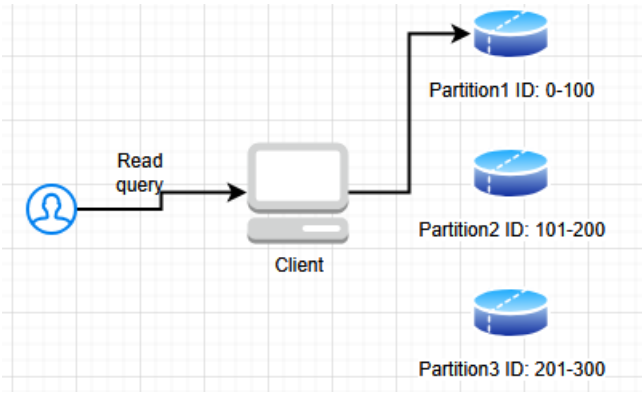
Routing Tier:
RoutingTier will have all the addresses, it's like a global index. It redirects user read query to the correct partition.



It’s better than RoundRobin because Time performance improves. Time complexity O(1)

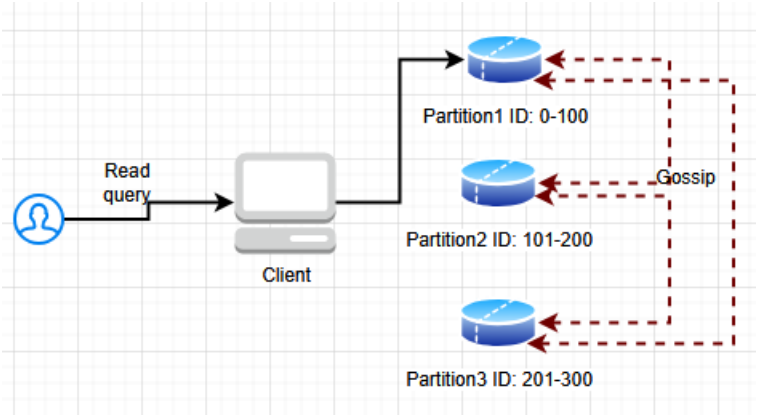
- RoutingTier determines which partition to look for
- RoutingTier contains logic for routing and maintains metadata for mapping

3. Clients are aware of partitioning and assignment of partitions in nodes:
We remove RoutingTier and client will be managing routing to partitions.




Client is aware that ID: 56 is in Partition1
- No RoutingTier
- All transactions will be handled by our client
- when a new partition arrives, client must be updated about it
Use coordination services: These services keep track of the partition. Examples where coordination services is part of the application are Zookeeper, HBase, Kafka

4. Gossip protocol: Cassandra
First lets say the Read query goes to P1 by default, if it doesn’t have the data then nodes/partitions gossip/interact among themselves to locate the data then the query will be redirected to that partition.




Homework:
CouchBase: it is a system from which we can handle data, replication, partitioning
How CouchBase handles routing? Look it up