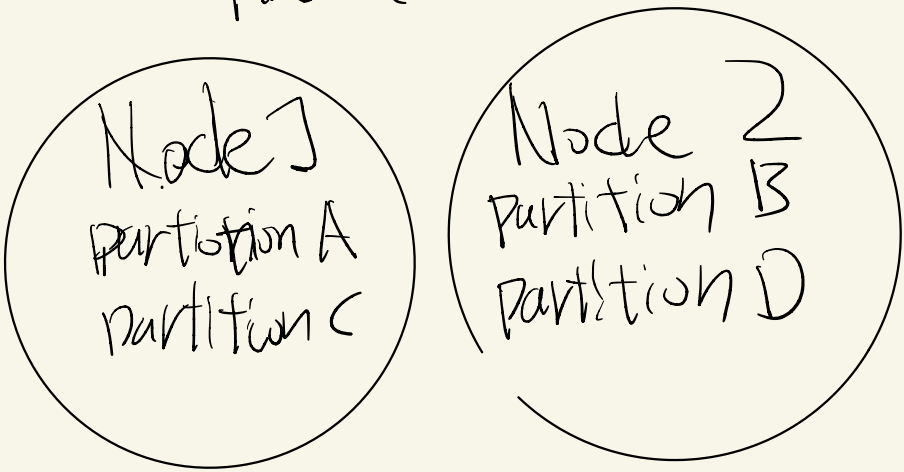
# CHAPTER 6 : Partitioning

# Chapter6: Partitioning (data sharding)

A partition: small database of its own
each piece of data belongs to one partition

Advantage of partioning: Scalability

different partitions can be placed on different nodes



skewed partition: some partitions have more data or queries than others

A hot spot: a partition with disapproportionately high load

## Partitioning of key-value data

- Partitioning by key Range
  - assign a continuous range of keys to each partition

Pros
- range scans are easy
  keys can be sorted in each partition

Cons
- hot spots may occur

## Partitioning by Hash of key
- assign a range of hashes for each partition

Pros
- keys can be distributed among the partitions

Cons
- range queries are inefficient

## ☐ Partitioning Secondary Indexes

### ✡ Partitioning by Document (aka local index)

— each secondary index is partitioned to the same one as the primary index

\\ may occur

scatter/gather

— must send a query to all partitions

— read queries are expensive

### ✡ Partitioning by Term (aka global index)

(term-partitioned)

— each secondary index is partitioned on the same node regardless of the node where its primary index is located

Pros

— range scan is efficient

Cons

— writes are expensive

↑ a write to a single document may affect multiple partitions of the index

# Rebalancing Partitions

rebalancing — the process of moving load from one node to another

## How to rebalance partitions

- Fixed number of partitions
  1. create many more partitions than there are nodes
  2. assign several partitions
  3. If a new node is added, it can steal a few partitions from every existing node until partitions are fairly distributed

  Pros — simple to implement

  Cons — hard to set the right size of initial number of partitions

- Dynamic Partitioning
  - split/merge partitions if a partitions reaches some thresholds

  Pros — Automatically adapt to the total data volume

  Cons — A single node has to deal with IO queries onset while other nodes are idle

# Request Routing

## 3 approaches to service discovery

In this case, how to find a node where a partition in question is located after rebalance

1. Allow clients to contact any node. If the first contacted node is the one in question, it handles the query. Otherwise, it forwards the request to another node.
2. All requests are sent to a routing tier first, then it forwards the requests to the appropriate node
3. Allow clients to directly send requests to the appropriate node. Clients need to be aware of rebalancing