Chapter 6. Partitioning

▼ Author: Pylon, Peng

Chapter 6 of this book discusses the concept of data partitioning. The chapter explains how data partitioning can be used to improve the scalability and performance of distributed systems. The author describes various partitioning techniques and their advantages and disadvantages. The chapter also covers the challenges that arise when partitioning data, such as data skew and hotspots, and explains how to mitigate these issues. Overall, Chapter 6 provides a comprehensive overview of the partitioning strategies that can be used in modern distributed systems.

Date: @March 21, 2023 9:09 PM

Topic: Chapter 6 Partitioning

Recall	**Notes**
What is data partition?	Normally, partitions are defined in such a way that each piece of data (each record, row, or document) belongs to exactly one partition.
Why partition?	Scalability: query load vs time.
How to?	A large dataset can be distributed across many disks, and the query load can be distributed across many processors.
Query across multiple partitions?	Do query first then join in the application layer?

📌 **SUMMARY:**
Partitioning is the process of dividing a large dataset into smaller, more manageable parts called partitions. We need partitioning to improve the scalability and performance of distributed systems. By partitioning data, we can distribute the workload and storage across multiple nodes, which allows us to handle larger datasets and more concurrent requests. Additionally, partitioning can help us avoid hotspots and improve fault tolerance.

Date: @March 21, 2023 9:26 PM

Topic: Partitioning & Replication

Recall

Notes

Combination of partitioning and single leader-based replication?

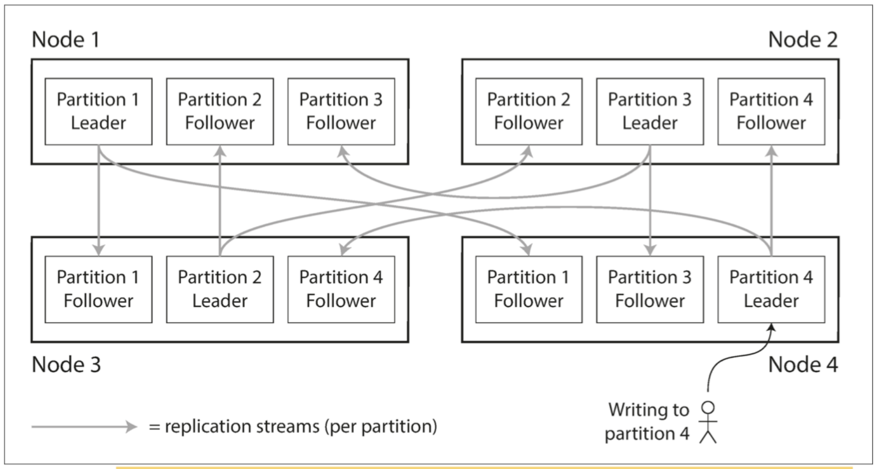


Figure 6-1. Combining replication and partitioning: each node acts as leader for some partitions and follower for other partitions.

If I have to split data into m partitions and have to set up n followers, how many nodes should I use?

📌 **SUMMARY:**
Partitioning is usually combined with replication so that copies of each partition are stored on multiple nodes. This means that, even though each record belongs to exactly one partition, it may still be stored on several different nodes for fault tolerance.

Date: @March 21, 2023 9:32 PM

Topic: Partitioning of Key-Value Data

Recall

Notes

What is skewed & hot spot?

Skewed: Query load is not evenly across nodes.

Hot spot: The worst case of skewed which all data was partitioned into one node, and other nodes are idle.

How do we avoid hot spots?

We need a better distribution algorithm.

#1 Partitioning by Key Range

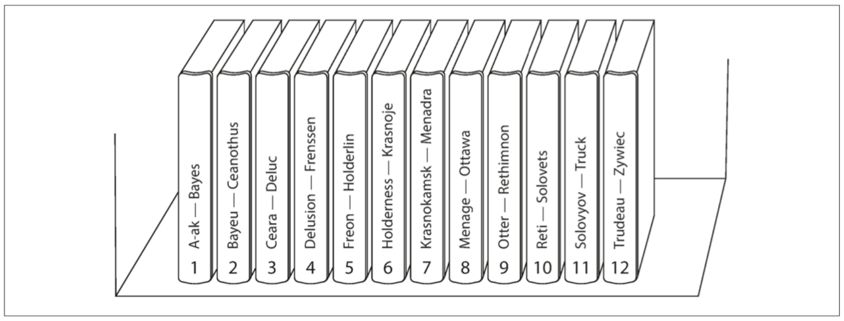


Figure 6-2. A print encyclopedia is partitioned by key range.

#2 Partitioning by Hash of keys

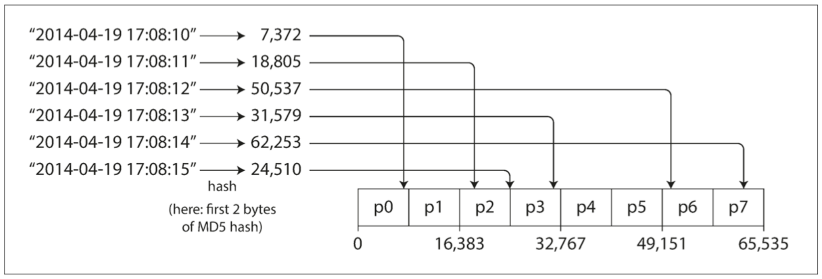


Figure 6-3. Partitioning by hash of key.

Pros and Cons for #1

Pros:

- Easy to implement
- Good for range queries

Cons:

- Hot spots can occur if keys are not evenly distributed
- Adding or removing nodes can require redistribution of data

Pros and Cons for #2

Pros:

- Keys are evenly distributed
- Adding or removing nodes does not require redistribution of data

Cons:

- Not suitable for range queries
- Requires a hash function that maps keys evenly to partitions

Skewed Workloads and Relieving Hot Spots

Add a random number to hotkeys for random hashing.

bookkeeping: track split keys

query needs additional work: as they have to read the data from all 100 keys and combine it

Redis problems to be considered?

Cache Hotspot Invalidation: one hot spot data used by multiple clients and the cache expired just now.

Solution:

Cache Avalanche: refers to a situation where many cache keys expire at the same time, causing a large number of requests to query the database simultaneously. This can overwhelm the database and cause a significant increase in load.

Solution: a) avoidance - set a random expiration time for hot spot data. b) handling - resilient solution, such as circuit breaker, rate limiter, etc.

Cache Penetration: quey a nonexisting date in db anyway.

Solution: a) cache nonexisting data b) bloom filter c) clients block invalid requests upfront

📌 **SUMMARY:**
Described the challenges that arise when partitioning key-value data, such as data skew and hotspots, and proposes two partitioning techniques: partitioning by key range and partitioning by hash of keys. This section highlights the pros and cons of each technique and emphasizes the importance of choosing a suitable distribution algorithm to avoid hotspots and ensure even workload distribution.

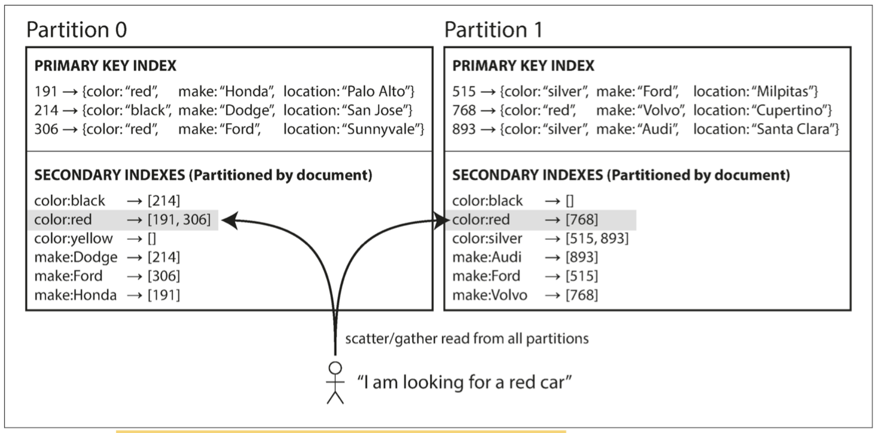
Date: @March 22, 2023 8:52 PM

Topic: Partitioning and Secondary Indexes

Recall

Notes

Partitioning Secondary Indexes by Document



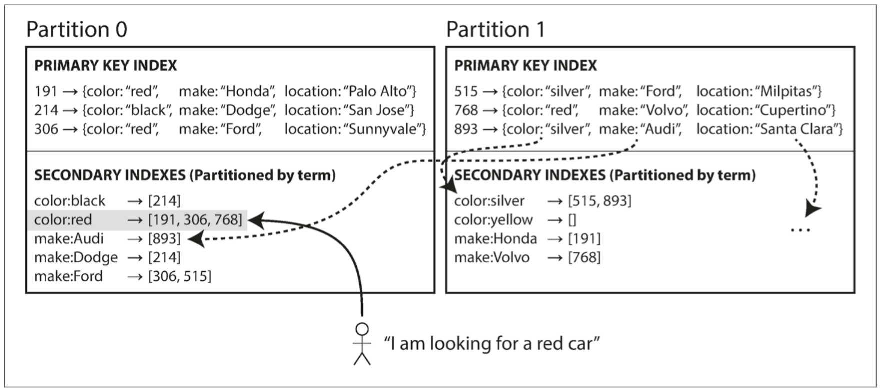
Partition 0
PRIMARY KEY INDEX
191 → {color: "red", make: "Honda", location: "Palo Alto"}
214 → {color: "black", make: "Dodge", location: "San Jose"}
306 → {color: "red", make: "Ford", location: "Sunnyvale"}
SECONDARY INDEXES (Partitioned by document)
color:black → [214]
color:red → [191, 306]
color:yellow → []
make:Dodge → [214]
make:Ford → [306]
make:Honda → [191]
Partition 1
PRIMARY KEY INDEX
515 → {color: "silver", make: "Ford", location: "Milpitas"}
768 → {color: "red", make: "Volvo", location: "Cupertino"}
893 → {color: "silver", make: "Audi", location: "Santa Clara"}
SECONDARY INDEXES (Partitioned by document)
color:black → []
color:red → [768]
color:silver → [515, 893]
make:Audi → [893]
make:Ford → [515]
make:Volvo → [768]
scatter/gather read from all partitions
"I am looking for a red car"

Figure 6-4. Partitioning secondary indexes by document.

Partitioning Secondary Indexes by Term

Pros: no longer scatter/gather

Cons: slower write



Partition 0
PRIMARY KEY INDEX
191 → {color: "red", make: "Honda", location: "Palo Alto"}
214 → {color: "black", make: "Dodge", location: "San Jose"}
306 → {color: "red", make: "Ford", location: "Sunnyvale"}
SECONDARY INDEXES (Partitioned by term)
color:black → [214]
color:red → [191, 306, 768]
make:Audi → [893]
make:Dodge → [214]
make:Ford → [306, 515]
Partition 1
PRIMARY KEY INDEX
515 → {color: "silver", make: "Ford", location: "Milpitas"}
768 → {color: "red", make: "Volvo", location: "Cupertino"}
893 → {color: "silver", make: "Audi", location: "Santa Clara"}
SECONDARY INDEXES (Partitioned by term)
color:silver → [515, 893]
color:yellow → []
make:Honda → [191]
make:Volvo → [768]
...
"I am looking for a red car"

Figure 6-5. Partitioning secondary indexes by term.

📌 **SUMMARY:**
The problem with secondary indexes is that they don’t map neatly to partitions. There are two main approaches to partitioning a database with secondary indexes: document-based partitioning and term-based partitioning.
Local index: document-partitioned index
Scatter/Gather: query data from all partitions ⇒ tail latency amplification.
Rather than each partition having its own secondary index (a *local index*), we can
construct a *global index* that covers data in all partitions. ⇒ Term-partitioned.

Date: @March 22, 2023 9:15 PM

Topic: Rebalancing Partitions

Recall	Notes
Rebalancing requirements	1. after rebalancing, the load should be shared fairly 2. No downtime when rebalancing 3. only move necessary data
Strategies for Rebalancing	#1 How not to do it: has mod N as N changes the hashing is broken #2 Fixed number of partitions. Image n node with always M partitions. if n ⇒ n + 1, then each node holds: M/(n+1)

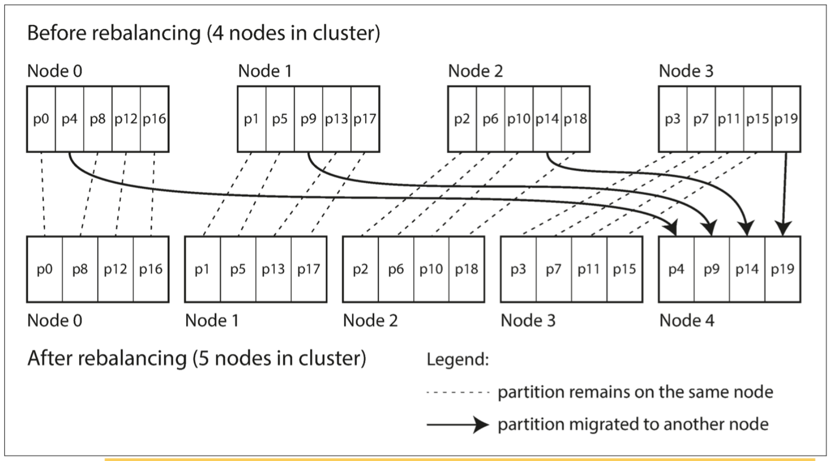


Figure 6-6. Adding a new node to a database cluster with multiple partitions per node.

#3 Dynamic partitioning

#4 Partitioning proportionally to nodes

📌 **SUMMARY:**
The process of moving **load** from one node in the cluster to another is called **rebalancing.**

Date: @March 22, 2023 9:38 PM

Topic: Request Routing

Recall

When a client wants to make a request, how does it know which node to connect to?

Notes

How do service discovery?

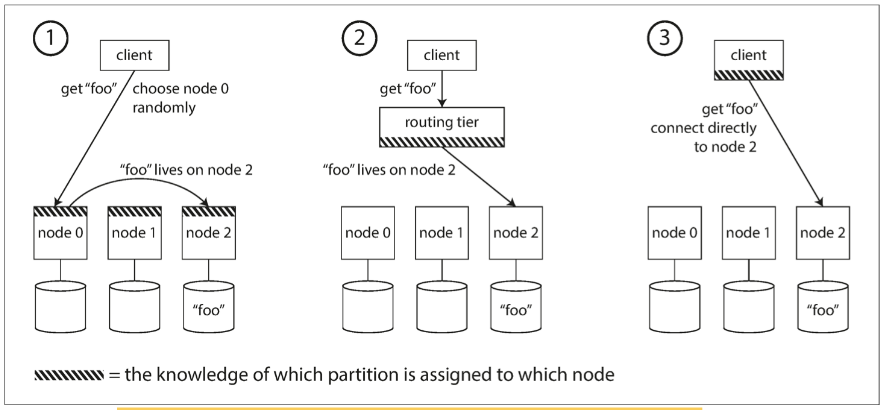
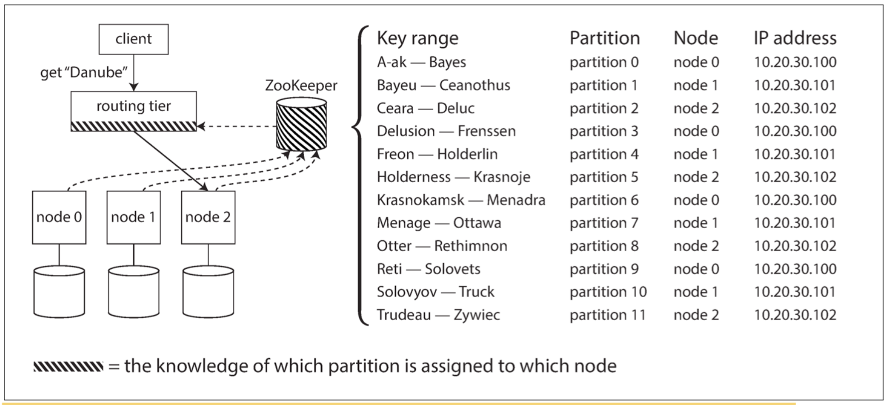


Figure 6-7. Three different ways of routing a request to the right node.

Many distributed data systems rely on a separate coordination service such as ZooKeeper to keep track of this cluster metadata. Each node registers itself in ZooKeeper, and ZooKeeper maintains the authoritative mapping of partitions to nodes.


client
get "Danube"
routing tier
ZooKeeper
node 0
node 1
node 2
▨ = the knowledge of which partition is assigned to which node

Key range	Partition	Node	IP address
A-ak — Bayes	partition 0	node 0	10.20.30.100
Bayeu — Ceanothus	partition 1	node 1	10.20.30.101
Ceara — Deluc	partition 2	node 2	10.20.30.102
Delusion — Frenssen	partition 3	node 0	10.20.30.100
Freon — Holderlin	partition 4	node 1	10.20.30.101
Holderness — Krasnoje	partition 5	node 2	10.20.30.102
Krasnokamsk — Menadra	partition 6	node 0	10.20.30.100
Menage — Ottawa	partition 7	node 1	10.20.30.101
Otter — Rethimnon	partition 8	node 2	10.20.30.102
Reti — Solovets	partition 9	node 0	10.20.30.100
Solovyov — Truck	partition 10	node 1	10.20.30.101
Trudeau — Zywiec	partition 11	node 2	10.20.30.102

Figure 6-8. Using ZooKeeper to keep track of assignment of partitions to nodes.

📌 **SUMMARY:**
Service discovery: refers to the automatic detection and identification of network services available in a distributed system. It enables applications and services to find each other and communicate efficiently without the need for hard-coded network addresses or configurations.
gossip protocol: to disseminate any change in cluster state used by Cassandra and Riak.
moxi: a routing tier used by Couchbase.
