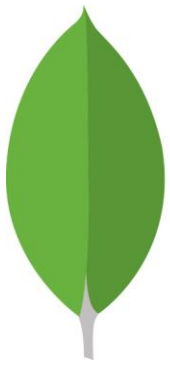


mongoDB

Sharding

Prof. P. M. Jadav
Associate Professor
Computer Engineering Department
Faculty of Technology
D. D. University, Nadiad

Introduction

- **Sharding** is the process of **splitting data across multiple machines**, known as **shards**, to:
  - Handle large datasets
  - Ensure high throughput (read/write performance)
  - Scale horizontally (add more servers to distribute load)

Understanding Partitioning

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Vertical Partitions

VP1

CUSTOMER ID	FIRST NAME	LAST NAME
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDA	BAĞCAN
4	JIM	PEPPER

VP2

CUSTOMER ID	FAVORITE COLOR
1	BLUE
2	GREEN
3	PURPLE
4	AUBERGINE

Horizontal Partitions

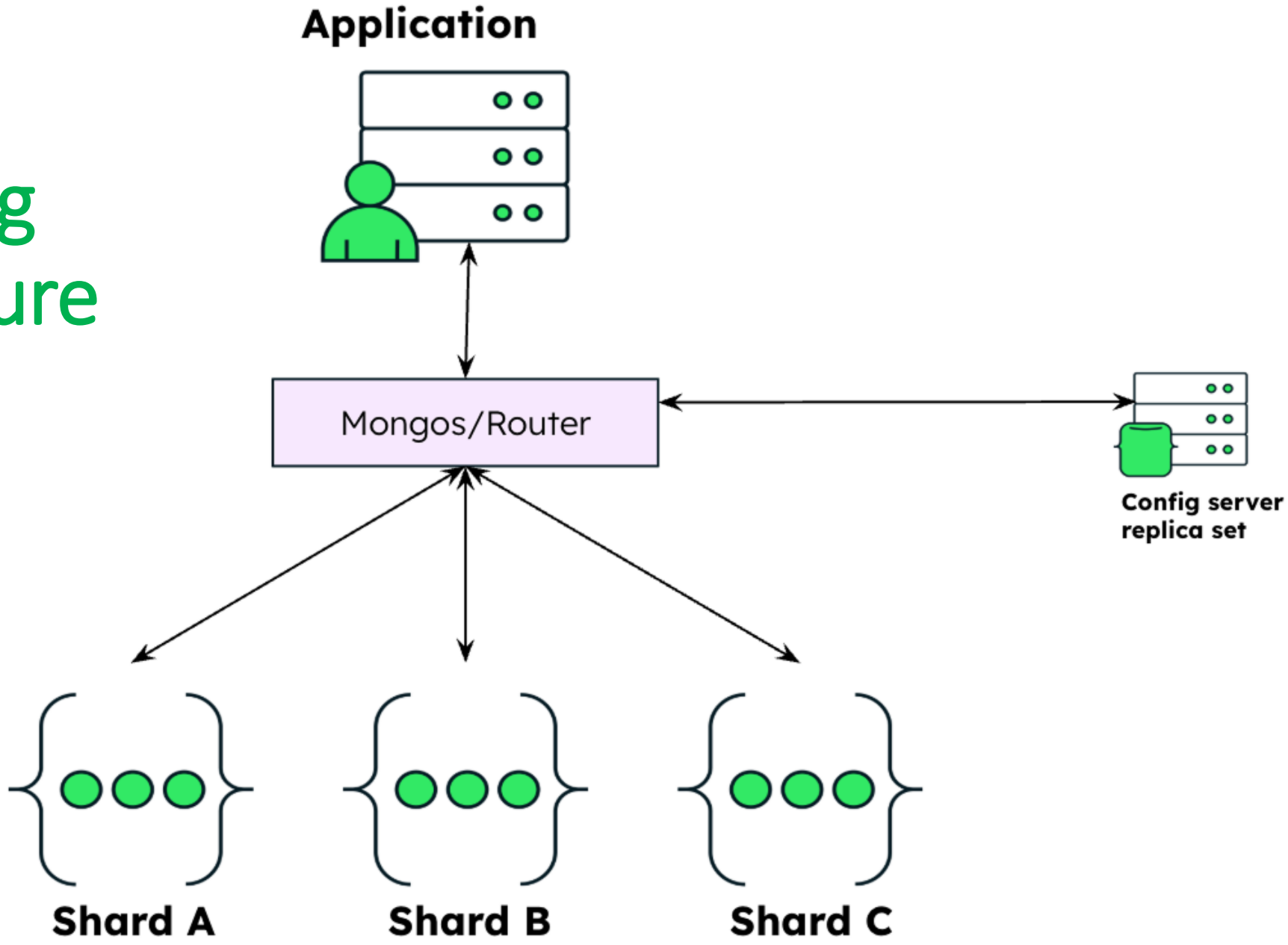
HP1

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN

HP2

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Sharding Architecture



Sharding Components

1. Shard

- Each **shard** holds a portion of the data
- It contains a subset of the cluster's data
- Each shard has a primary and two secondary nodes by default
- The data is **partitioned** among shards based on a **shard key**

Sharding Components

2. Shard Key

- A field(s) used to determine the distribution of data across shards
- Must be chosen carefully as it impacts **performance**, **query distribution**, and **data balancing**
- Types:
  - **Ranged Sharding** – based on value ranges of the shard key
  - **Hashed Sharding** – uses a hash of the shard key for uniform distribution

Sharding Components

3. Config Server Replica Set (CSRS)

- Store the **metadata** about the cluster and data distribution
- Minimum **3 config servers** are required in production for fault tolerance
- They maintain the mapping between shard key ranges and shards

Sharding Components

4. Query Routers (mongos)

- Mongos instances route queries from clients to the appropriate shard(s)
- The client application connects to mongos, not directly to the shards
- mongos consults config servers to locate the relevant data

Advantages

1) Increased read/write throughput — By distributing the dataset across multiple shards, read/write operation capacity is increased as long as read and write operations are confined to a single shard
2) Increased storage capacity — Similarly, by increasing the number of shards, you can also increase overall total storage capacity
3) High availability — Each shard is a replica set, every piece of data is replicated. Even if an entire shard becomes unavailable since the data is distributed, the database as a whole still remains partially functional, with part of the schema on different shards

Disadvantages

1) Query overhead

- Each sharded database must have a separate machine or service which understands how to route a querying operation to the appropriate shard
- This introduces additional latency on every operation
- If the data required for the query is horizontally partitioned across multiple shards, the router must then query each shard and merge the result together
- This can make an otherwise simple operation quite expensive and slow down response times

Disadvantages

2) Complexity of administration

- With every sharded database, on top of managing the shards themselves, there are additional service nodes to maintain
- In cases where replication is being used, any data updates must be mirrored across each replicated node
- Overall, a sharded database is a more complex system which requires more administration

Disadvantages

3) Increased infrastructure costs

- Sharding by its nature requires additional machines and compute power over a single database server
- While this allows your database to grow beyond the limits of a single machine, each additional shard comes with higher costs
- The cost of a distributed database system, especially if it is missing the proper optimization, can be significant

Sharding Types

1) Range/Dynamic Sharding
2) Algorithmic/Hashed Sharding
3) Entity-Relationship based Sharding
4) Geography based Sharding

Range/ Dynamic Sharding

PRODUCT	PRICE
WIDGET	$118
GIZMO	$88
TRINKET	$37
THINGAMAJIG	$18
DOODAD	$60
TCHOTCHKE	$999

($0-$49.99)

PRODUCT	PRICE
TRINKET	$37
THINGAMAJIG	$18

($50-$99.99)

PRODUCT	PRICE
GIZMO	$88
DOODAD	$60

($100+)

PRODUCT	PRICE
WIDGET	$118
TCHOTCHKE	$999

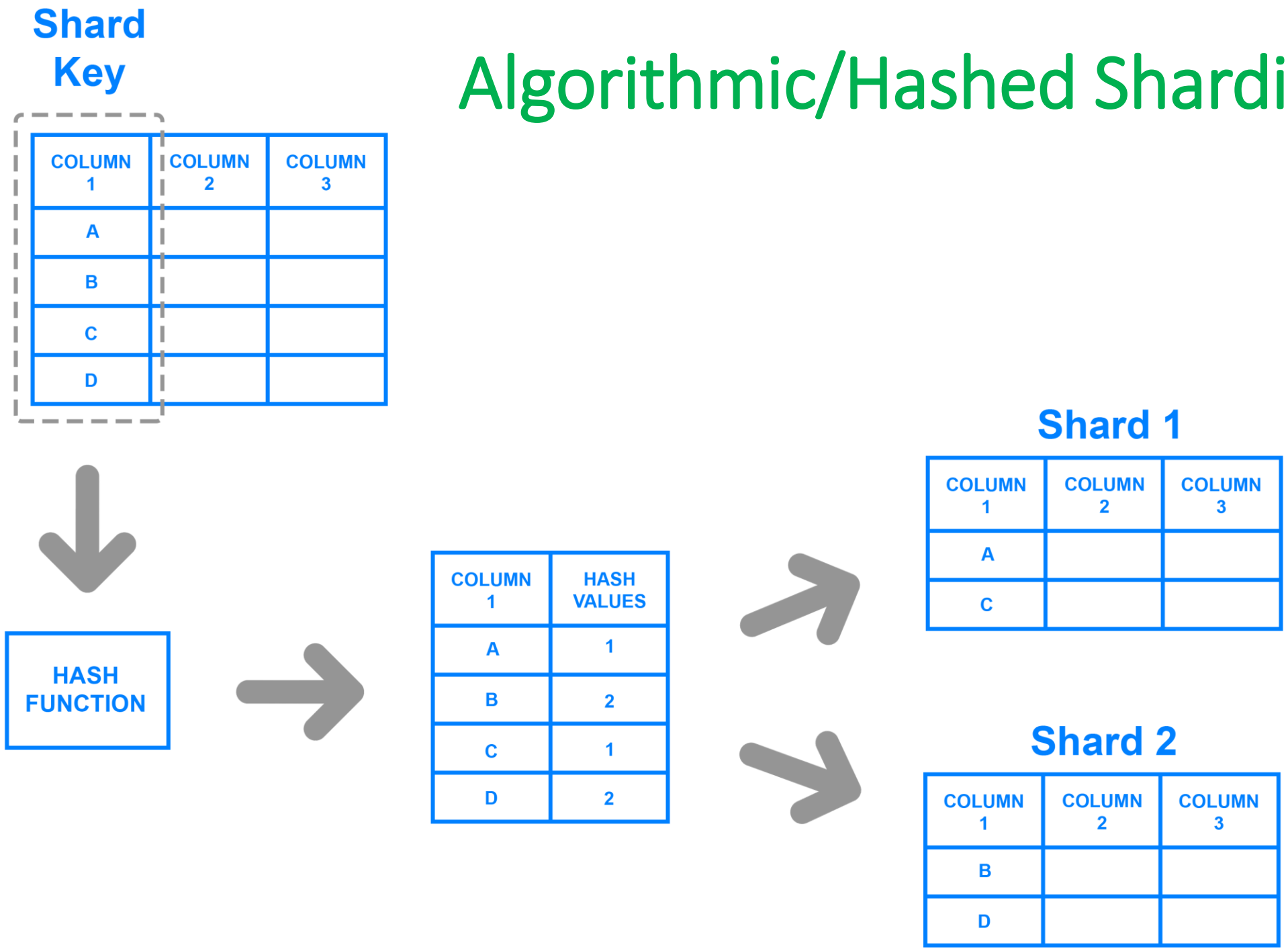
Range/Dynamic Sharding

- It takes a field on the record as an input and, based on a predefined range, allocates that record to the appropriate shard
- The field on which the range is based is also known as the shard key
- A poor choice of shard key will lead to unbalanced shards, which leads to decreased performance
- An effective shard key will allow for queries to be targeted to a minimum number of shards

Algorithmic/Hashed Sharding

- It takes a record as an input and applies a hash function or algorithm to it which generates an output or hash value
- This output is then used to allocate each record to the appropriate shard
- Hashed sharding typically disregards the meaning of the data

Algorithmic/Hashed Sharding



Algorithmic/Hashed Sharding

- Advantages:
  - Hashing the inputs allows more even distribution across shards even when there is not a suitable shard key
  - No lookup table needs to be maintained
- Disadvantages:
  - Query operations for multiple records are more likely to get distributed across multiple shards
  - Resharding can be expensive
  - Any update to the number of shards likely requires rebalancing all shards to moving around records

Entity-Relationship based Sharding

- It keeps related data together on a single physical shard
- In a relational database related data is often spread across several different tables
- e.g. Consider the case of a shopping database with users and payment methods
  - Each user has a set of payment methods that is tied tightly with that user
  - Keeping related data together on the same shard can reduce the need for broadcast operations, increasing performance

Geography based Sharding

- It keeps related data together on a single shard
- The data is related by geography
- This is essentially ranged sharding where the shard key contains geographic information and the shards themselves are geo-located
- For example, consider a dataset where each record contains a “country” field
- In this case, we can both increase overall performance and decrease system latency by creating a shard for each country or region, and storing the appropriate data on that shard

References

- https://www.mongodb.com/docs/manual/sharding/
- https://www.mongodb.com/resources/products/capabilities/database-sharding-explained
- https://www.mongodb.com/resources/products/capabilities/sharding
- https://www.digitalocean.com/community/tutorials/understanding-database-sharding