NAME:SIDDARTH SAKTHI M

REG.NO:230701314

ASSIGNMENT4

MongoDB Sharding

Sharding is a method used in MongoDB to distribute data across multiple servers or clusters. This technique is crucial for managing large datasets and high-throughput applications, allowing for horizontal scaling, improved performance, and increased availability.

Key Concepts of Sharding

- **Shard**: A shard is a single instance of a MongoDB database that holds a subset of the sharded data. Each shard can be a standalone MongoDB server or a replica set.
- **Sharded Cluster**: A sharded cluster consists of multiple shards, along with config servers and query routers (mongos). Config servers store metadata and configuration settings for the cluster, while query routers direct client requests to the appropriate shard.
- **Shard Key**: The shard key is a field (or fields) used to distribute the data across shards. The choice of shard key is crucial as it determines how evenly the data will be distributed and how efficiently queries will be routed to the appropriate shard.
- **Chunks**: Data in a sharded cluster is divided into chunks based on the shard key. Each chunk contains a range of shard key values. MongoDB automatically manages the distribution of chunks across shards.
- **Balancing**: MongoDB includes a balancer process that automatically redistributes chunks among shards to ensure an even distribution of data and workload.

How Sharding Works

- **Data Distribution**: When a document is inserted into a sharded collection, MongoDB uses the shard key to determine which shard the document belongs to. The data is then distributed across the shards based on the ranges of the shard key.
- **Query Routing**: When a client queries a sharded collection, the query router (mongos) determines which shard(s) contain the relevant data based on the shard key and directs the query to those shards.
- **Scalability**: As data grows, additional shards can be added to the cluster. The balancer will redistribute the chunks to ensure that the new shard receives its fair share of data.

Applications of MongoDB Sharding

- **Large-Scale Applications**: Sharding is ideal for applications with large datasets that exceed the storage capacity of a single server. For example, social media platforms, e-commerce websites, and online gaming applications often generate massive amounts of data.
- **High Throughput**: Applications that require high read and write throughput can benefit from sharding, as it allows for distributing the load across multiple servers. This is particularly useful for real-time analytics and logging systems.
- **Global Applications**: Applications with a global user base can use sharding to distribute data geographically. By placing shards in different regions, latency can be reduced, and data can be served from the nearest shard.
- **Multi-Tenant Systems**: In multi-tenant applications, sharding can be used to isolate data for different tenants. Each tenant can be assigned a specific range of the shard key, ensuring that their data is stored together and can be managed independently.
- **Content Management Systems**: Content-heavy applications, such as video streaming platforms or document management systems, can leverage sharding to store and manage large volumes of media files or documents efficiently.
- **IoT Applications**: Internet of Things (IoT) applications often generate massive streams of data from numerous devices. Sharding can help manage this data efficiently, allowing for real-time processing and analysis.