MongoDB
Replication & Sharding

Team Emertxe

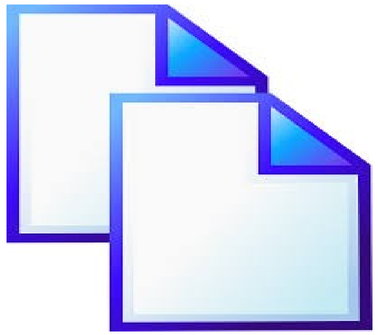
ΣMERTXE

Replication

Replication is the process of making replica (a copy) of something i.e the same data is available on more than one MongoDB server.

An Replica set in MongoDB is a group of MongoDB process that maintain the same data set.

Replication provide data redundancy and increases data availability.



Replica Set Member

The member of replica set are :

- The Primary Replica is the main server which interacts with the clients and perform all the read/write operations .
- The Secondary Replica maintain a copy of the data of the primary.

When a Primary Replica fails , the Replica set automatically switches over to the secondary and becomes the primary server.

Primary

MongoDB applies write operations on the primary’s oplog.

Secondary members replicate this log and apply the operations to their data sets.

All members of replica sets can accept read operations.

Primary

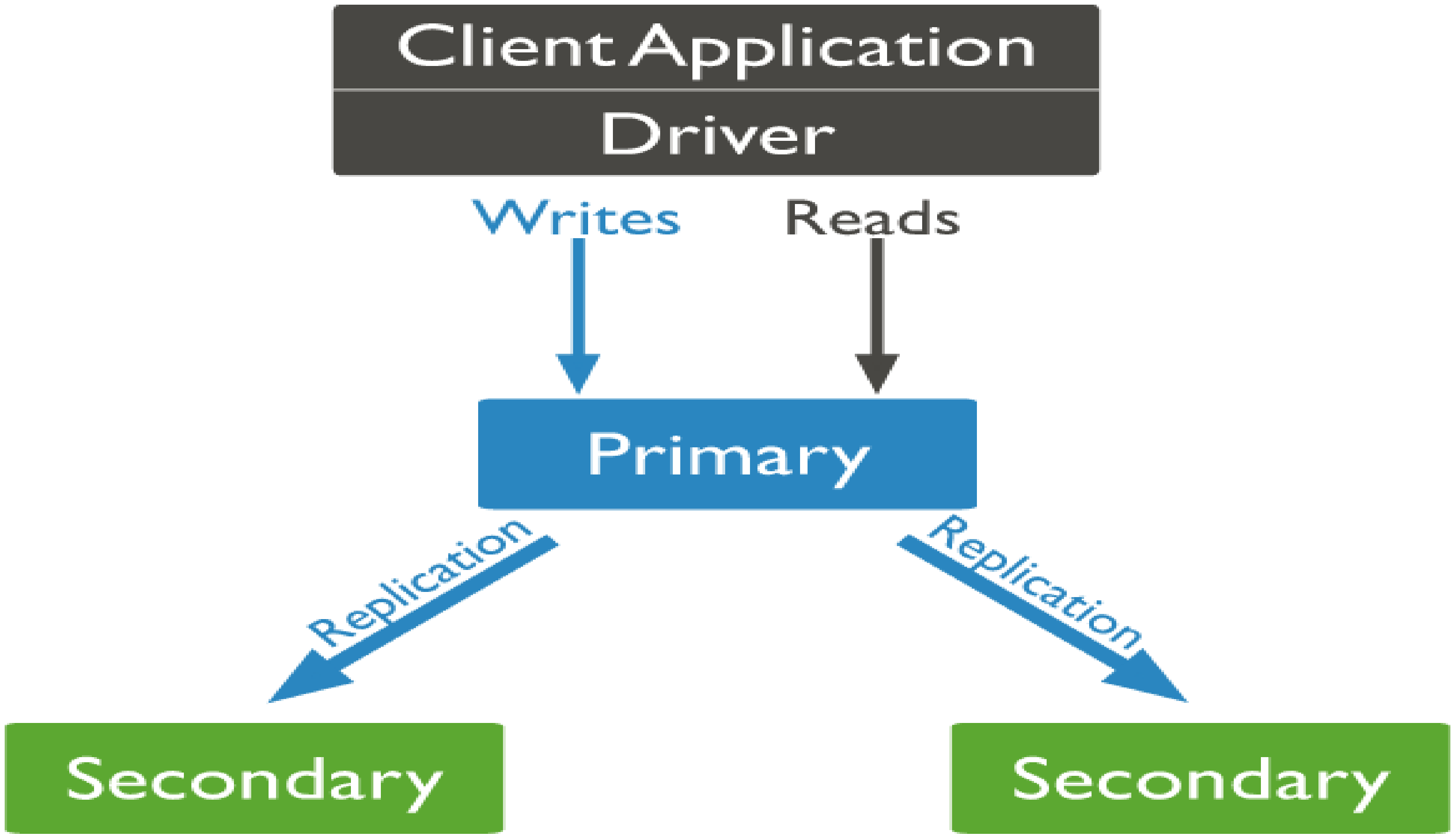


Image source : https://docs.mongodb.com/manual/core/replica-set-members/

Secondaries

A secondary maintains the copy of the primary’s data set.

A replica set can have one or more secondaries .

Clients can only read data from secondaries , not write data to secondaries.

A Secondary can become a primary if the current primary is not available.

The replica set holds an election to choose which of the secondaries becomes new primary.

HeartBeats

Replica set members send heartbeats (pings) to each other every two seconds.

If a heartbeat does not return within 10 seconds , the other member mark the member as inaccessible.

Secondaries

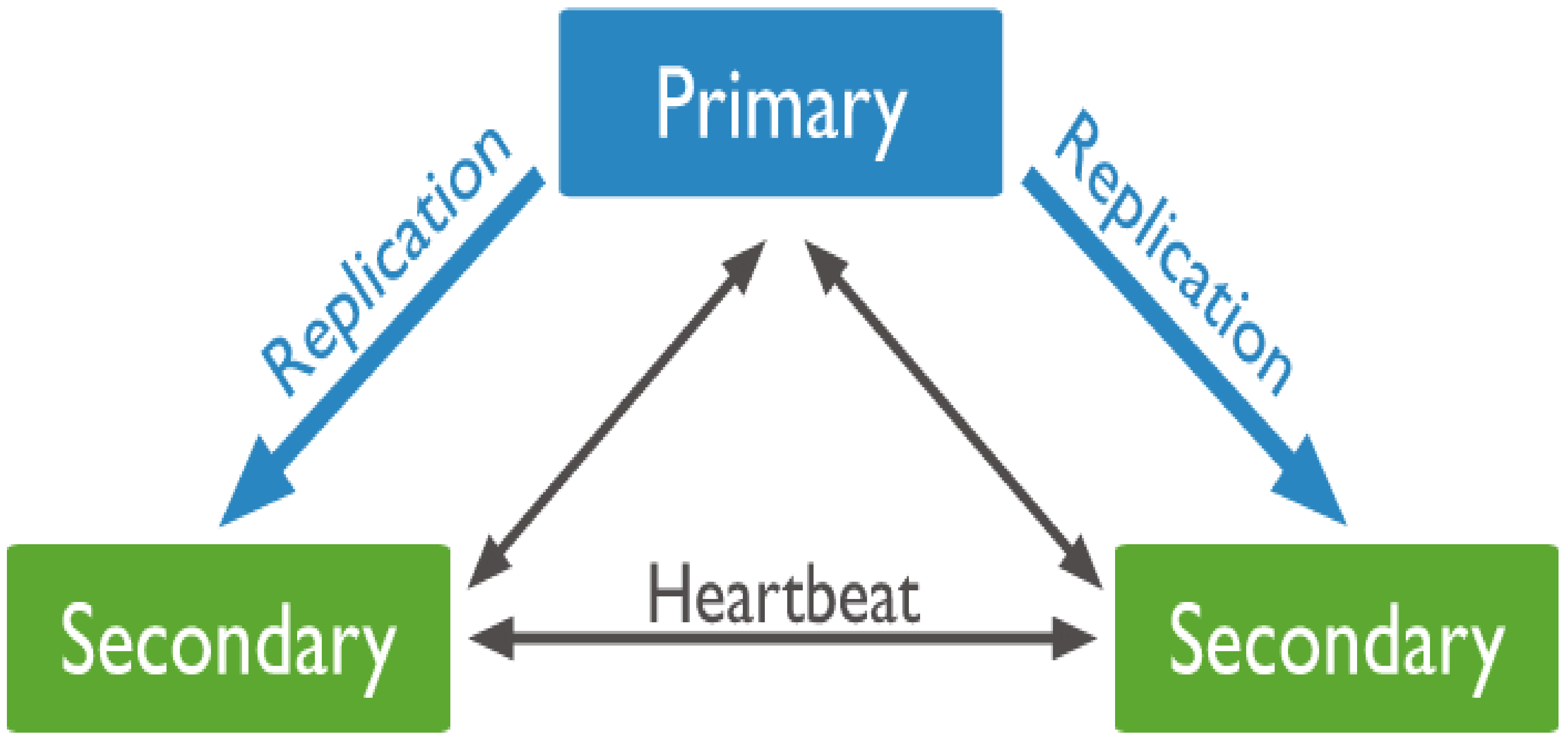


Image source : https://docs.mongodb.com/manual/core/replica-set-members/

Arbiter

The arbiter only votes in elections for primary.

Arbiter have exactly one election vote.

Replica sets use elections to determine which set member will become primary.

If a primary is not available , one of the remaining secondaries holds an election to elect itself as a new primary key.

Arbiter

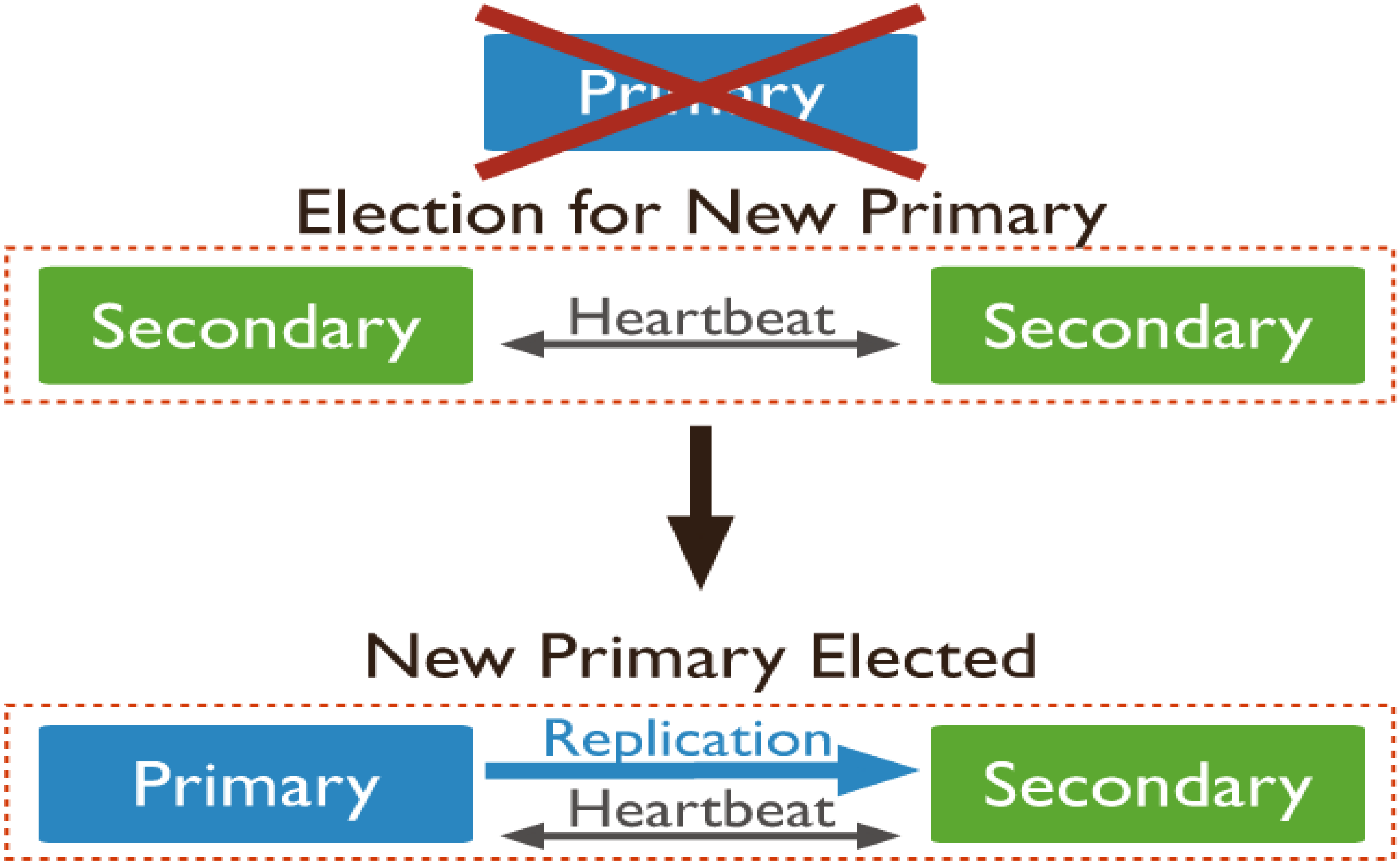


Image source : https://docs.mongodb.com/manual/core/replica-set-members/

Replication Features

Replica Set Oplog

The Oplog (Operation log) keeps a rolling record of all operations that modify data stored in the database.

MongoDB applies database operation on Primary and then records the operations on the Primary’s Oplog.

The Secondary operations then copy these operations.

All replica set member contain a copy of the Oplog in the local.oplog.rs collection to maintain current state of the database.

Replica Set
Data Synchronization

MongoDB uses two form of data synchronization :

- Initial Sync
- Replication

Initial Sync : Initial Sync copies all the data from one member of the replica set to another.

Replication : Secondary Member replicate data continuously after the initial sync.

Replica Set
High Availability

Replica Set provide high availability using automatic failover.

Failover allows a secondary member to become primary if the current primary not available.

In some situations the failover process may undertake a rollback.

Replication Methods

Replication Methods

Method Name	Description
rs.add()	Adds a number to replica set.
rs.addArb()	Adds an Arbiter to a replica set.
rs.help()	Returns basic help text for replica set function.
rs.conf()	Returns the replica set configuration document
rs.remove()	Remove a Member from a replica set.

Add Member to an existing Replica Set

The rs.add() method adds a member to replica set.

To run this method connect to the primary of replica set.

Add Member to Replica Set

Parameter	Type	Description
Host	String or document	Specify the host name & optionally port number
arbiterOnly	Boolean	Optional , Applies only if the <host> value is string , if true the added host is an arbiter.

Add Member to Replica Set (Example)

>rd.add('docs.mongodb.com : 8080');

Default port : 8080

if 'docs.mongodb.com' is an arbiter following form is used:

>rd.add('docs.mongodb.com : 8080',true);

Sharding

Sharding

Sharding is a method for distributing data across multiple machines.

MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

MongoDB supports horzontal scaling through sharding.

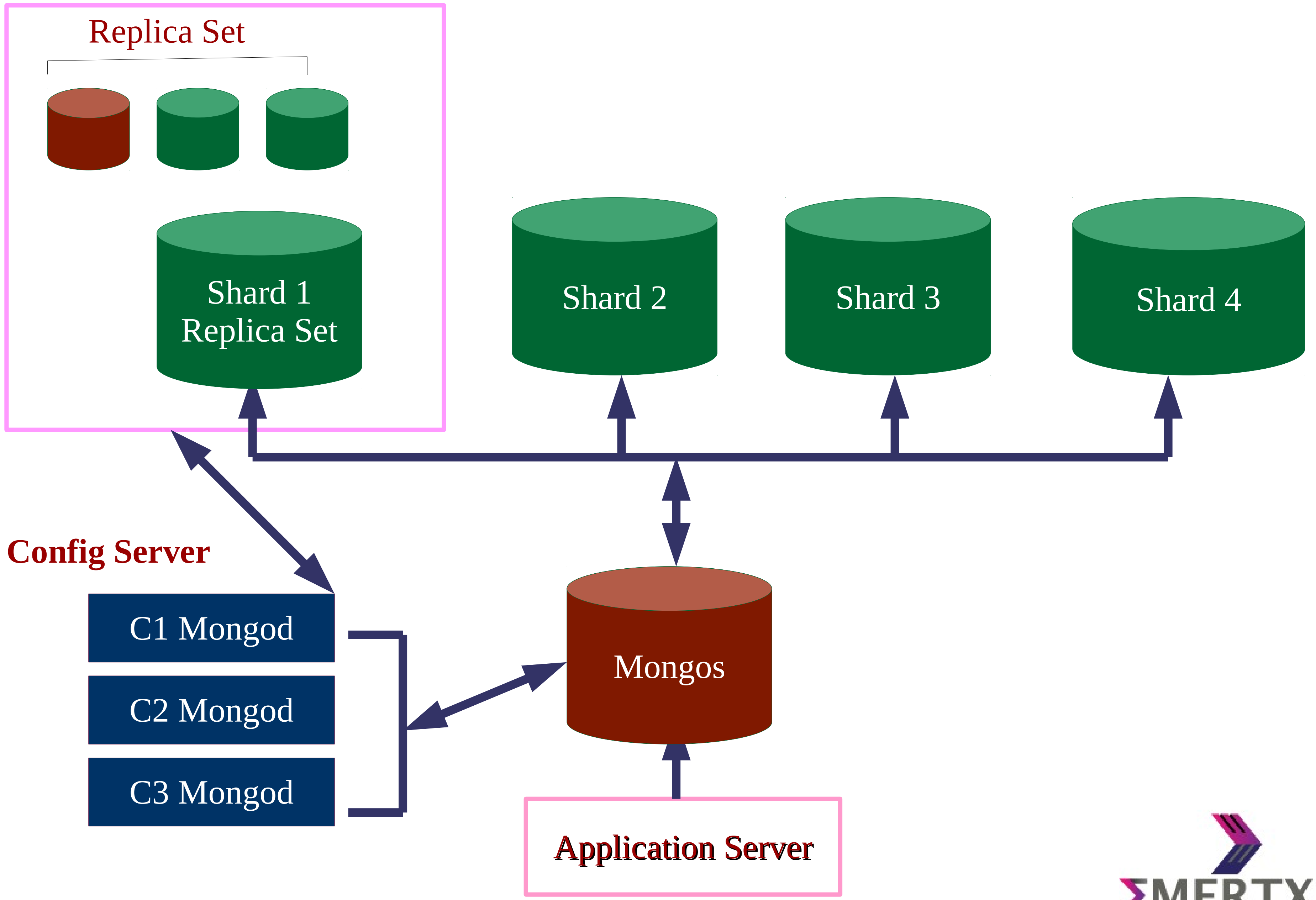
Why Sharding?

- All the data will be written to the Master node in replication.
- Space in local disk may not be huge enough to handle the data growth.
- Vertical scaling is too expensive.
- Simple replica set has limitation of 12 nodes.

Sharding Component

- Shards are used to store actual data.
- Config server stores the mapping of the cluster's data set to the shards.
- Query Router provides an interface between client application and sharded cluster. The Mongos act as a query router.

Sharding Architecture



ΣMERTXE

Shard Keys

The Shard key determines the distribution of the collection’s documents among the cluster’s shard.

The Shard keys is either an indexed field or indexed compound fields that exist in every document in the collection.

MongoDB partitions the collection to distribute the documents in a collection using shard key.

A sharded collection can have only one shard key.

Shard Key Specification

To Specify Shard key

`sh.shardCollection(namespace, key)`

- Namespace consists of a string <database>.<collection>
- The key parameter consist of document containing a field and the index for that field.

Shard key Indexes

All Sharded collections must have an index that supports the shard key.

- If the collection is empty , sh.shardCollection() creates the index on the shard key.
- If the collection is not empty , create the index before using sh.shardCollection()

Shard Key Limitations

- A shard key cannot exceed 512 bytes.
- A shard key index can an ascending index on the shard key.
- A shard key index cannot be an index that specifies a multikey index , a text index or a geospatial index on the shard key fields.
- Shard key is Immutable.

Chunks

MongoDB partitions sharded data into chunks .

Each chunk has an inclusive lower and exclusive upper range based on the shard key.

The default chunk size in MongoDB is 64 megabytes.

We can increase or reduce the chunk size.

Data Partitioning with chunks

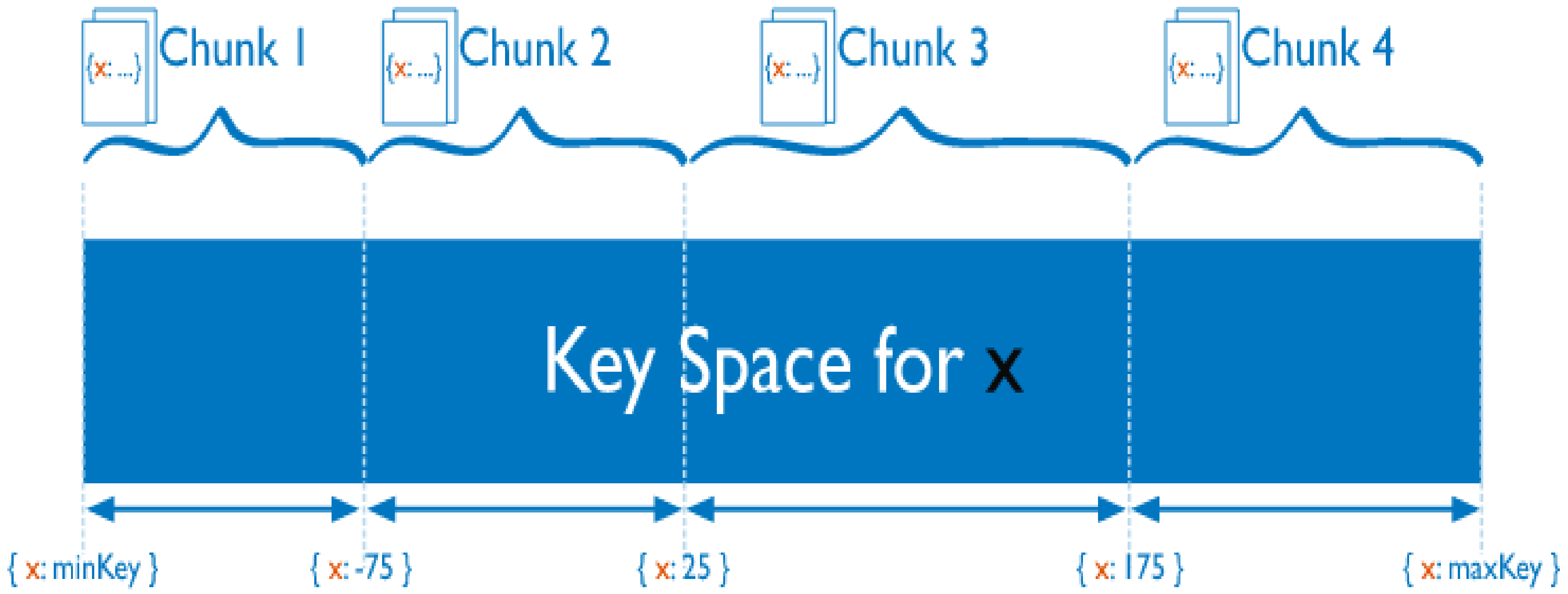


Image Source : https://docs.mongodb.com/manual/core/sharding-data-partitioning/

Data Partitioning with chunks

The mongos routes writes to the appropriate chunk based on the shard key value.

The smallest range chunk can represent a single unique shard key value.

MongoDB splits chunks when they grow beyond the configured chunk size.

The smallest range a chunk can represent a single unique shard key value cannot be split.

Advantages of Sharding

- Read/Writes

  MongoDB distributes the read and write workload across the shards in the sharded cluster. Both read and write workloads can be scaled horizontally.

- Storage Capacity

  Sharding distributes data across the shards in the cluster allowing each shard to contain a subset of the total cluster data.

Advantages of Sharding

- High Availability

  A sharded cluster can continue to perform partial read/write operations even if one or more shards are unavailable.

Hashed Sharding

Hashed Sharding uses a hashed index of a single field as the shard key to partition data across the sharded cluster.

Hashed Sharding provides more even data distribution across the sharded cluster

Hashed Sharding

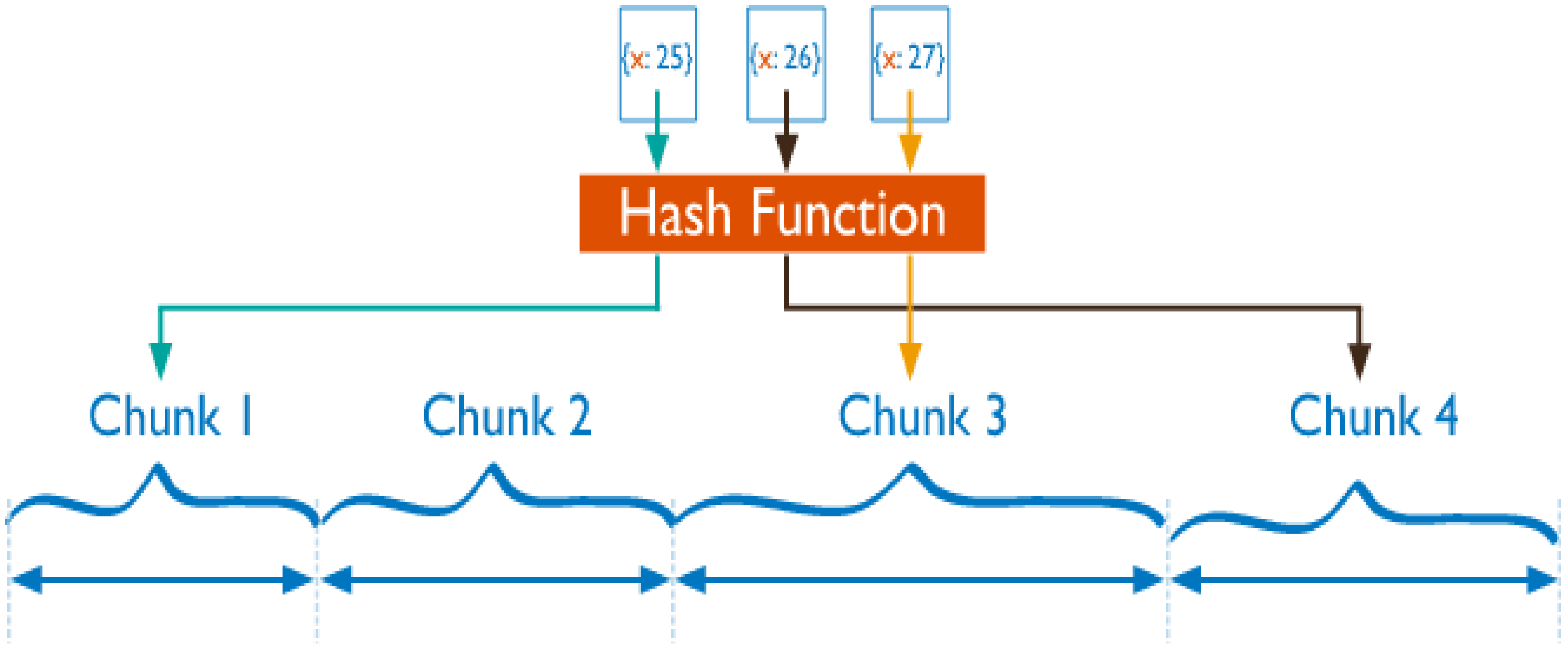


Image Source : https://docs.mongodb.com/manual/core/hashed-sharding/

Ranged Sharding

Range based Sharding involves dividing data into contiguous ranges determined by the shard key values.

This allows for efficient queries where read target documents within a contiguous range.

Range based Sharding is the default Sharding methodology.

Ranged Sharding

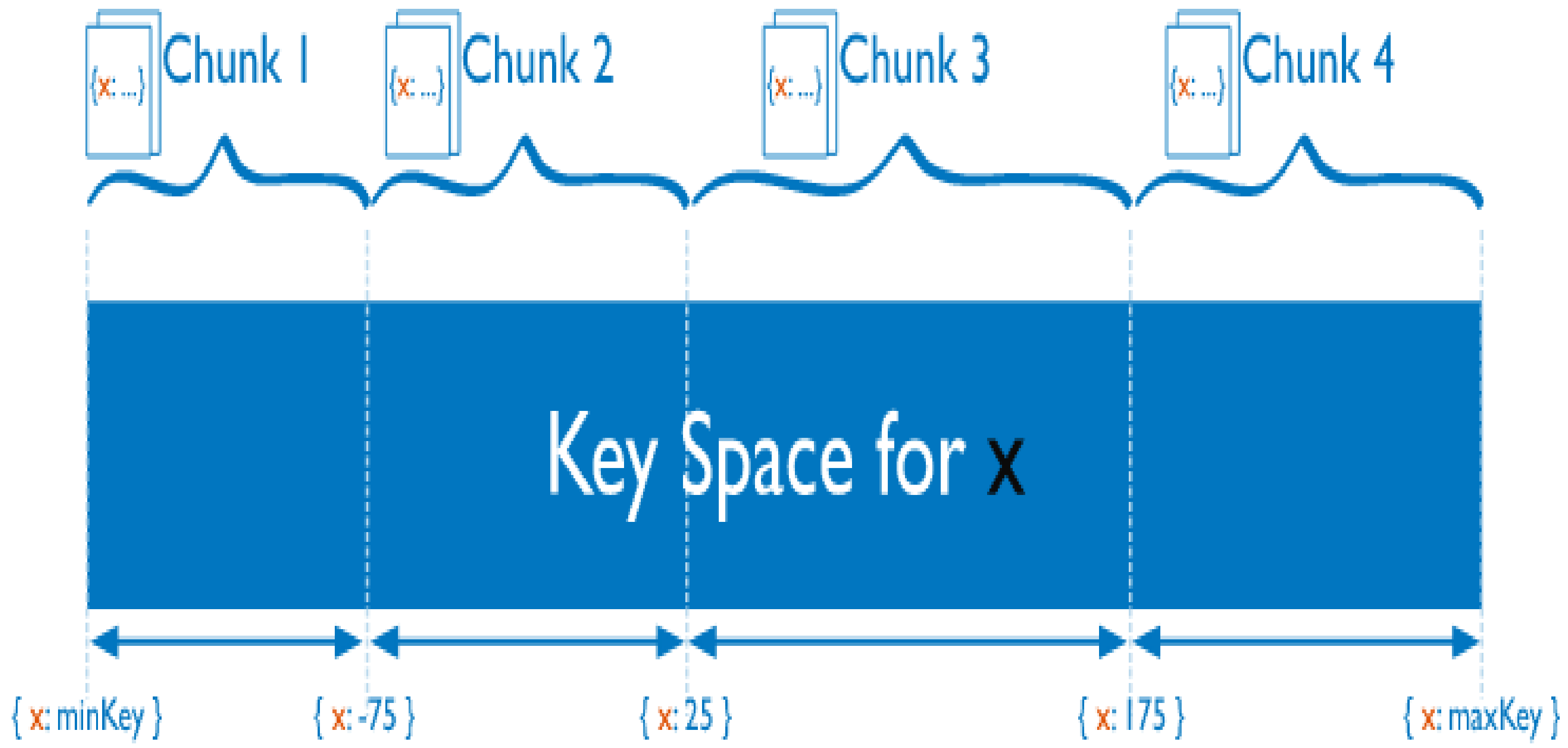


Image Source : https://docs.mongodb.com/manual/core/ranged-sharding/

References

- https://www.wikimedia.org/
- https://docs.mongodb.com/manual/

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe

https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

Thank You