



#hashtag

# Introduction to Sharding

Speaker Name

*Software Engineer, 10gen*

# Agenda

- Scaling Data
- MongoDB's Approach
- Architecture
- Configuration
- Mechanics

# Scaling Data



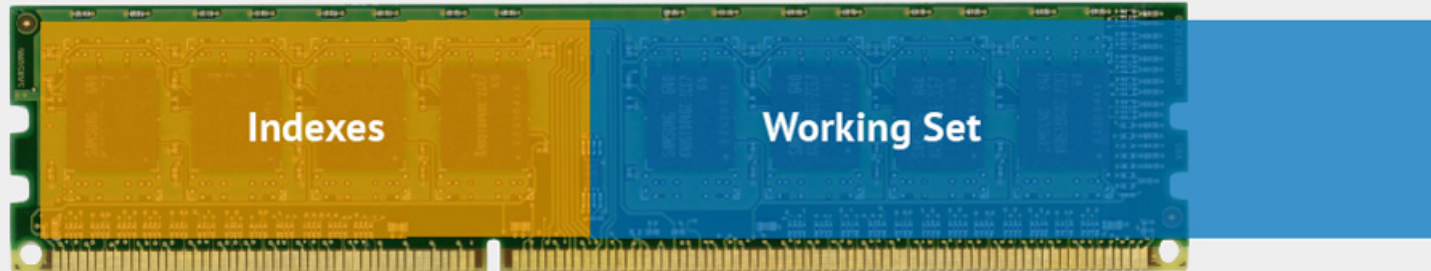
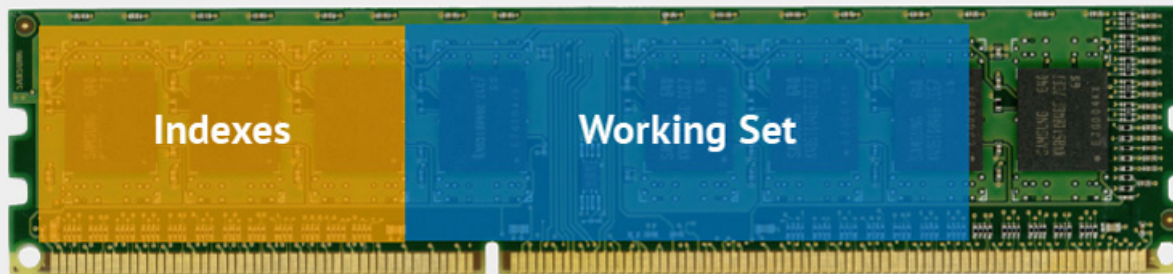
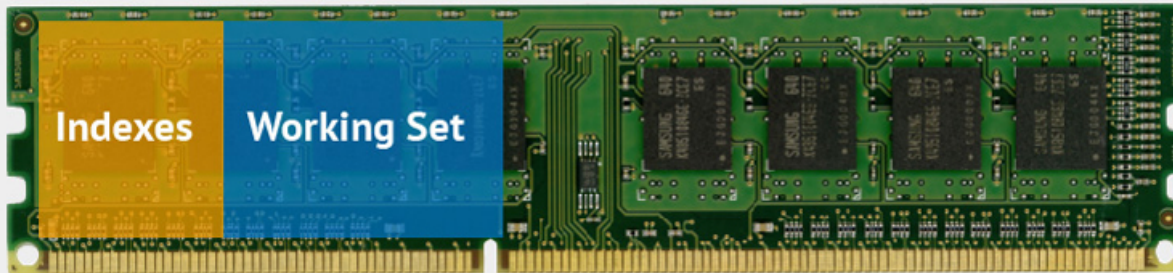
# Examining Growth

- User Growth
  - 1995: 0.4% of the world's population
  - Today: 30% of the world is online (~2.2B)
  - Emerging Markets & Mobile
- Data Set Growth
  - Facebook's data set is around 100 petabytes
  - 4 billion photos taken in the last year (4x a decade ago)

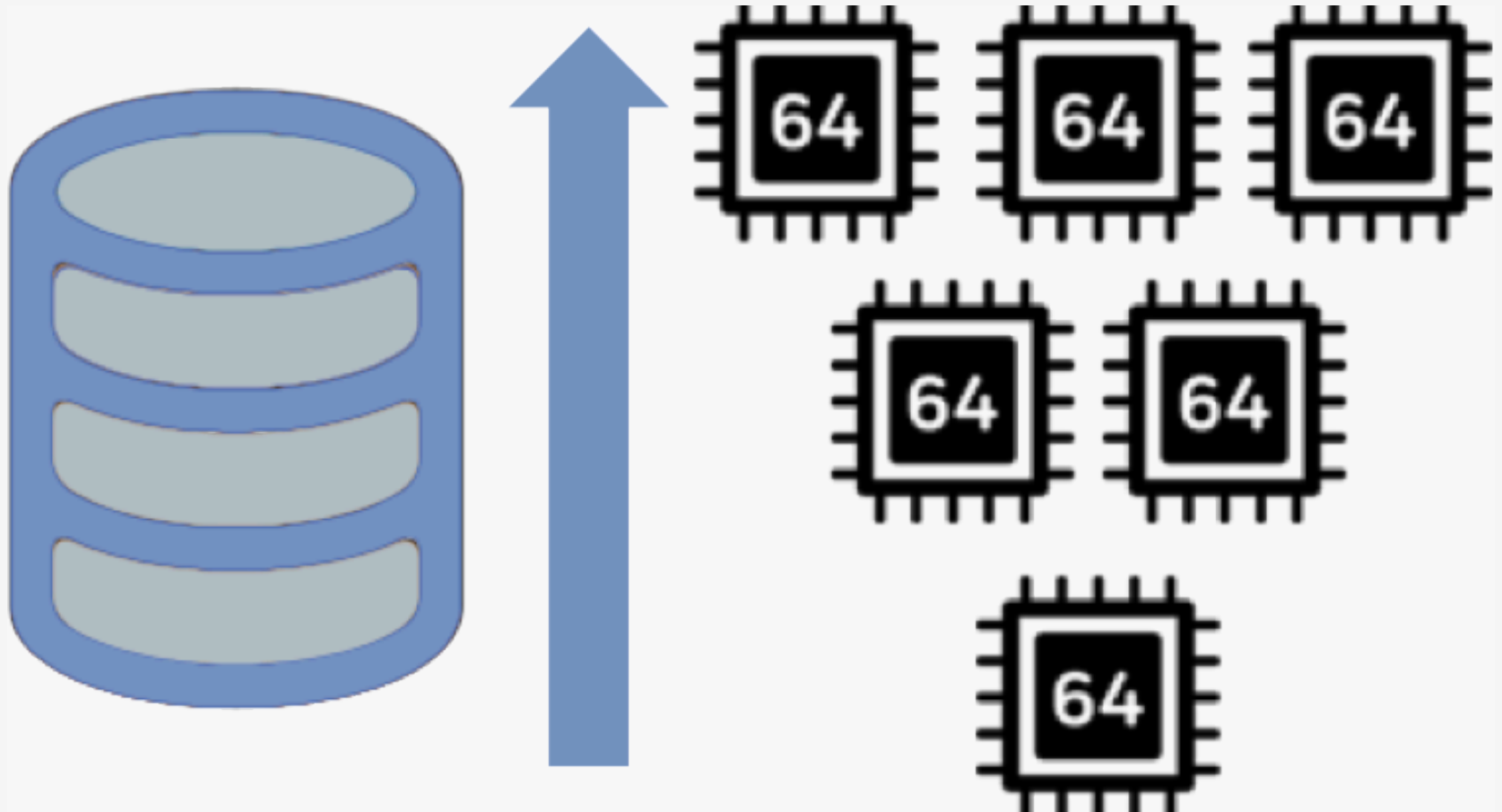
# Read/Write Throughput Exceeds I/O



# Working Set Exceeds Physical Memory



# Vertical Scalability (Scale Up)



# Horizontal Scalability (Scale Out)





# Data Store Scalability

- Custom Hardware
  - Oracle
- Custom Software
  - Facebook + MySQL
  - Google

# Data Store Scalability Today

- MongoDB Auto-Sharding
- A data store that is
  - Free
  - Publicly available
  - Open Source (<https://github.com/mongodb/mongo>)
  - Horizontally scalable
  - Application independent

# MongoDB's Approach to Sharding

# Partitioning

- User defines shard key
- Shard key defines range of data
- Key space is like points on a line
- Range is a segment of that line

$-\infty$

Key Space

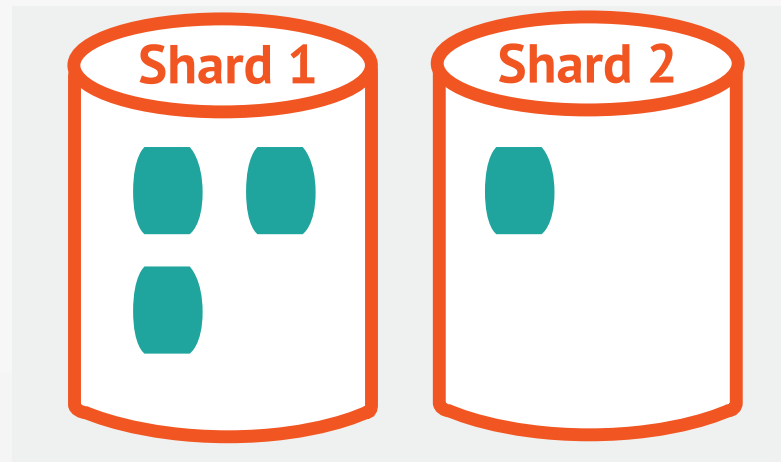
$+\infty$

# Data Distribution

Initially 1 chunk

Default max chunk size: 64mb

MongoDB automatically splits & migrates chunks when max reached

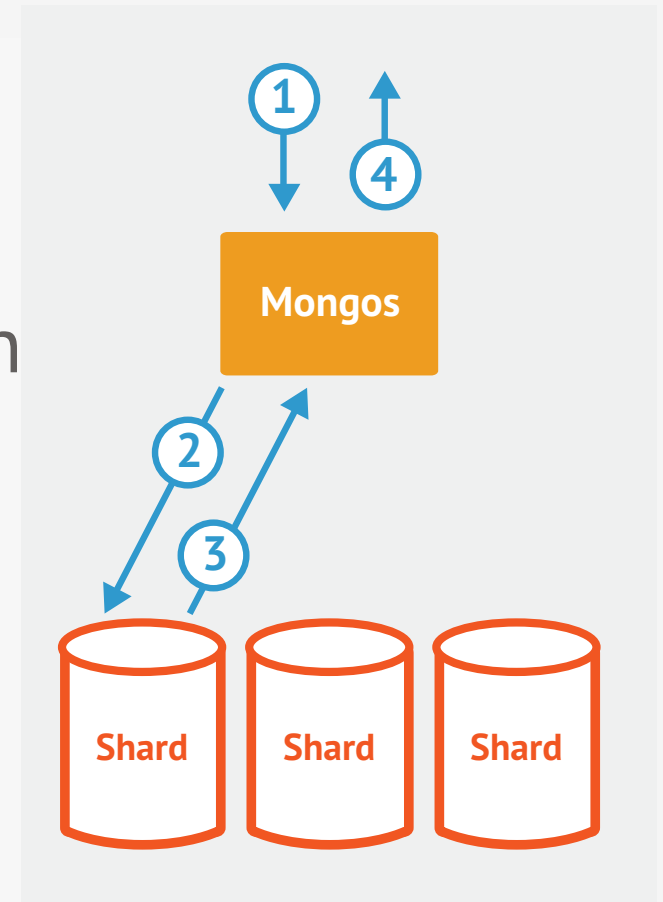


# Routing and Balancing

Queries routed to specific shards

MongoDB balances cluster

MongoDB migrates data to new n



# MongoDB Auto-Sharding

- Minimal effort required
  - Same interface as single mongod
- Two steps
  - Enable Sharding for a database
  - Shard collection within database

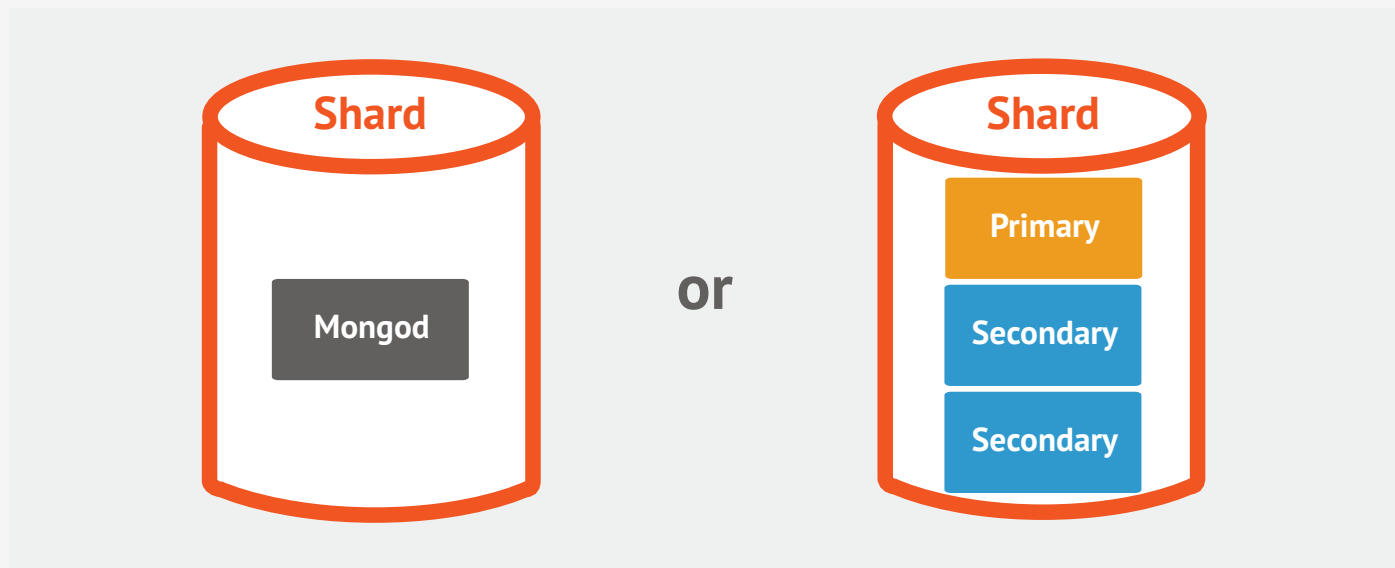
# Architecture





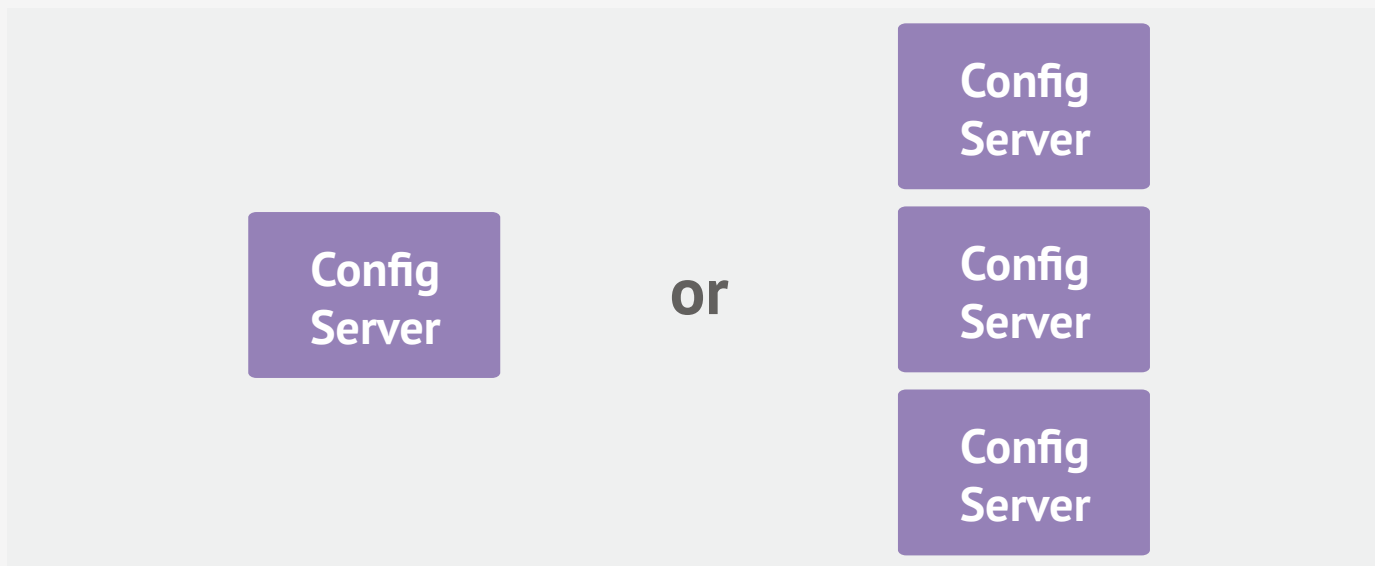
# What is a Shard?

- Shard is a node of the cluster
- Shard can be a single mongod or a replica set



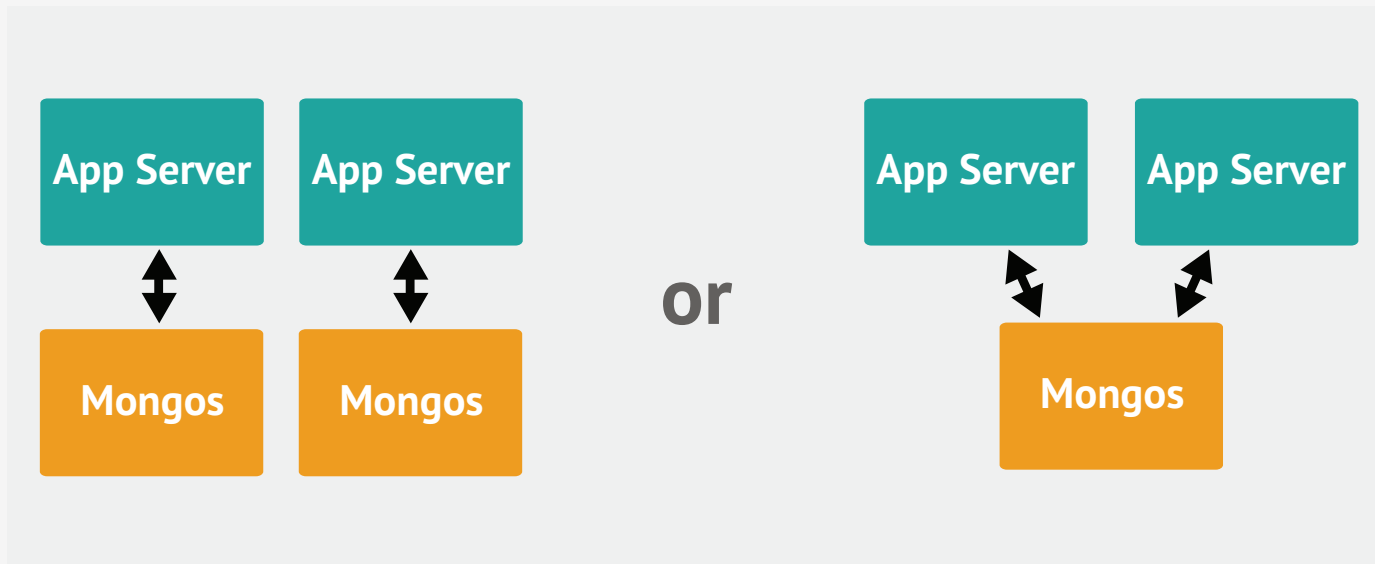
# Meta Data Storage

- Config Server
  - Stores cluster chunk ranges and locations
  - Can have only 1 or 3 (production must have 3)
  - Not a replica set

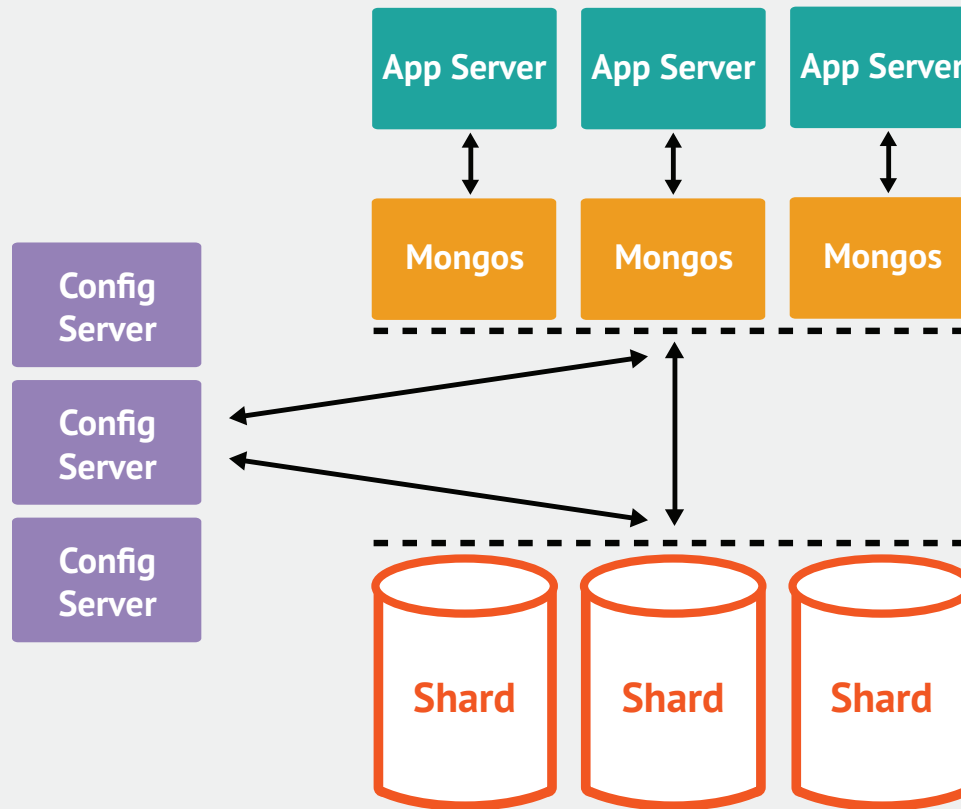


# Routing and Managing Data

- Mongos
  - Acts as a router / balancer
  - No local data (persists to config database)
  - Can have 1 or many



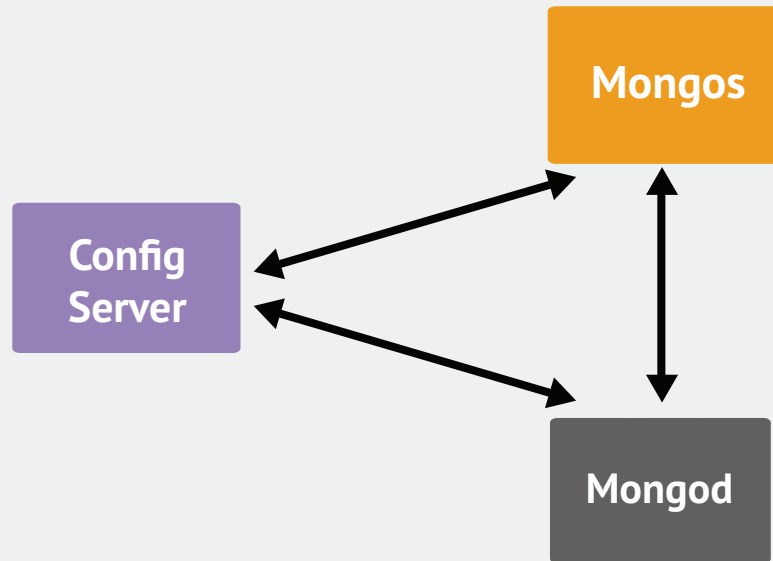
# Sharding infrastructure



# Configuration



# Example Cluster



# Starting the Configuration Server

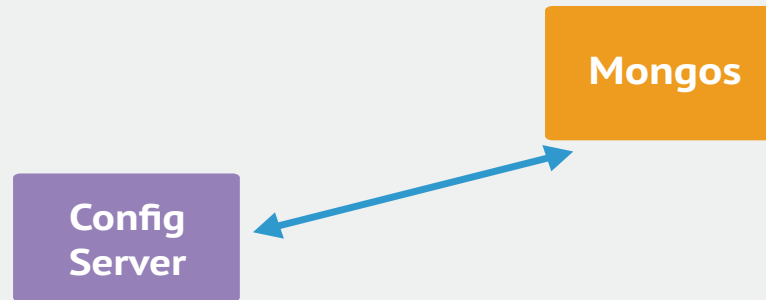


Config  
Server

```
mongod --configsvr
```

Starts a configuration server on the default port (27019)

# Start the mongos Router



```
mongos --configdb <hostname>:27019
```

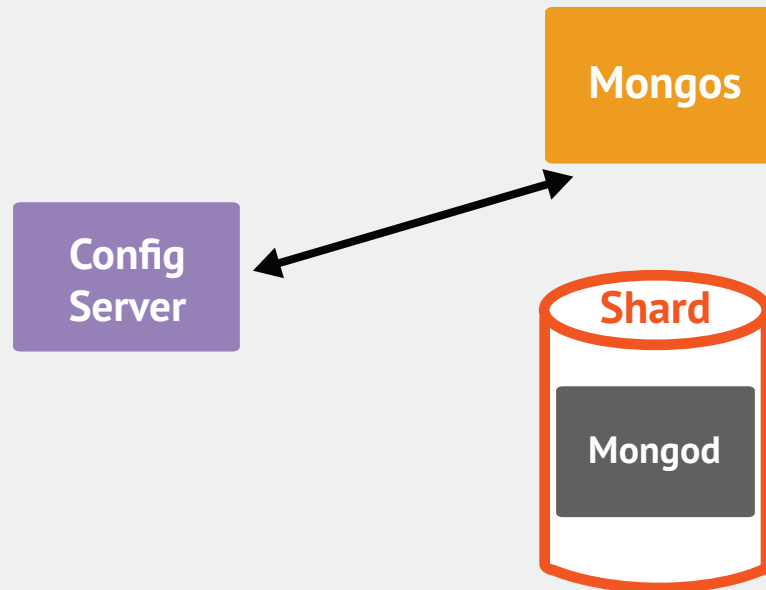
For 3 configuration servers:

```
mongos --configdb <host1>:<port1>,<host2>:<port2>,<host3>:<port3>
```

This is always how to start a new mongos, even if the cluster is already running



# Start the shard database



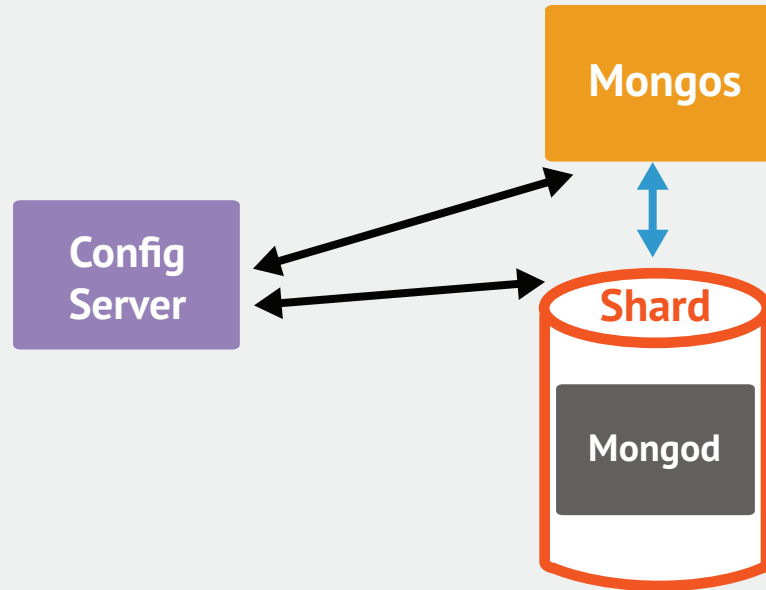
```
mongod --shardsvr
```

Starts a mongod with the default shard port (27018)

Shard is not yet connected to the rest of the cluster

Shard may have already been running in production

# Add the Shard



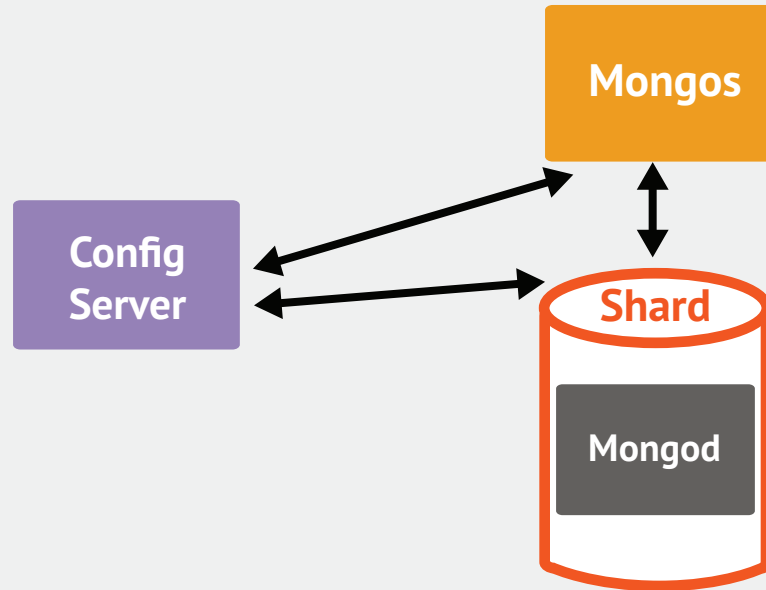
On mongos:

- `sh.addShard( '<host>:27018' )`

Adding a replica set:

- `sh.addShard( '<rsname>/<seedlist>' )`

# Verify that the shard was added



```
db.runCommand({ listshards:1 })

{ "shards" :
  [ { "_id": "shard0000", "host": "<hostname>:27018" } ],
  "ok" : 1
}
```

# Enabling Sharding

- Enable sharding on a database

```
sh.enableSharding("<dbname>")
```

- Shard a collection with the given key

```
sh.shardCollection("<dbname>.people", {"country":1})
```

- Use a compound shard key to prevent duplicates

```
sh.shardCollection("<dbname>.cars", {"year":1, "uniqueid":1})
```

# Tag Aware Sharding

- Tag aware sharding allows you to control the distribution of your data
- Tag a range of shard keys
  - `sh.addTagRange(<collection>,<min>,<max>,<tag>)`
- Tag a shard
  - `sh.addShardTag(<shard>,<tag>)`

# Mechanics

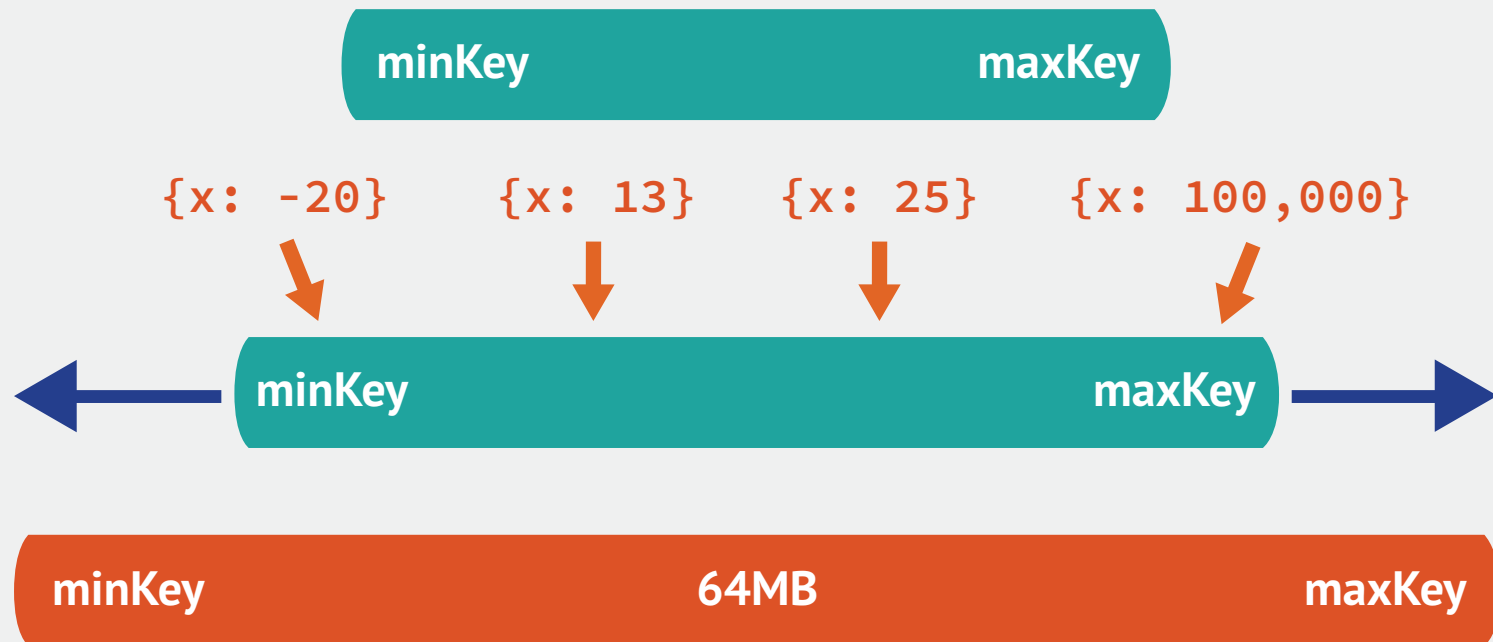


# Partitioning

- Remember it's based on ranges

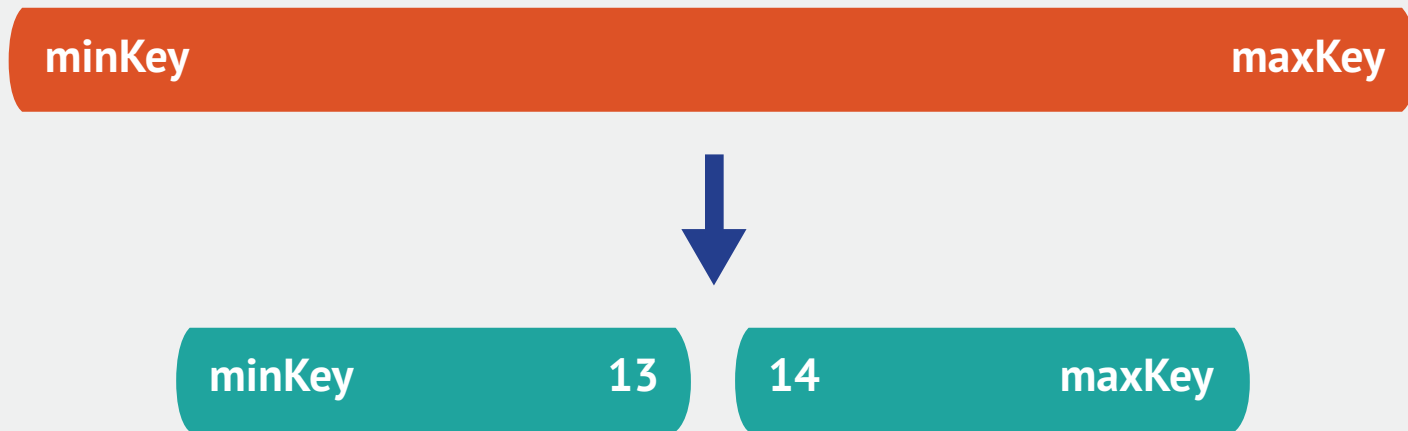


# Chunk is a section of the entire range

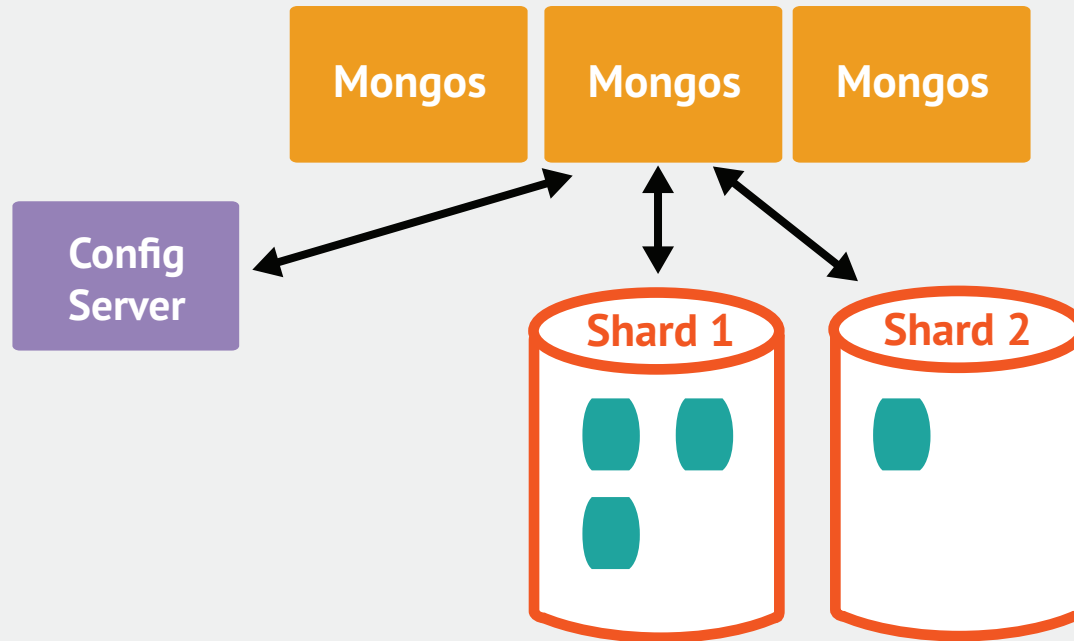




# Chunk splitting



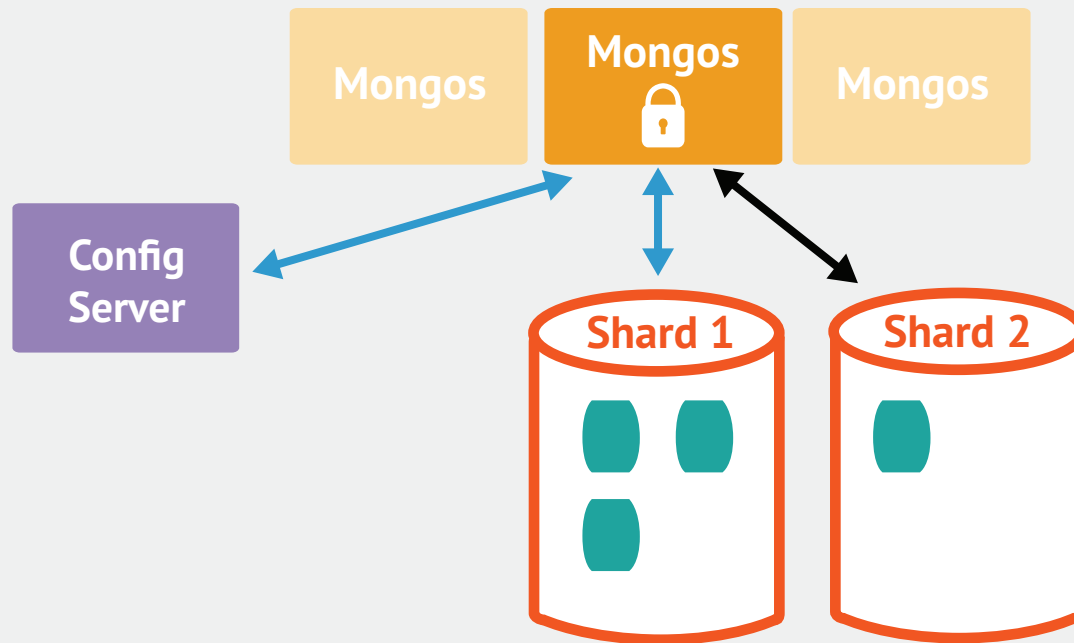
# Balancing



Balancer is running on mongos

Once the difference in chunks between the most dense shard and the least dense shard is above the migration threshold, a balancing round starts

# Acquiring the Balancer Lock

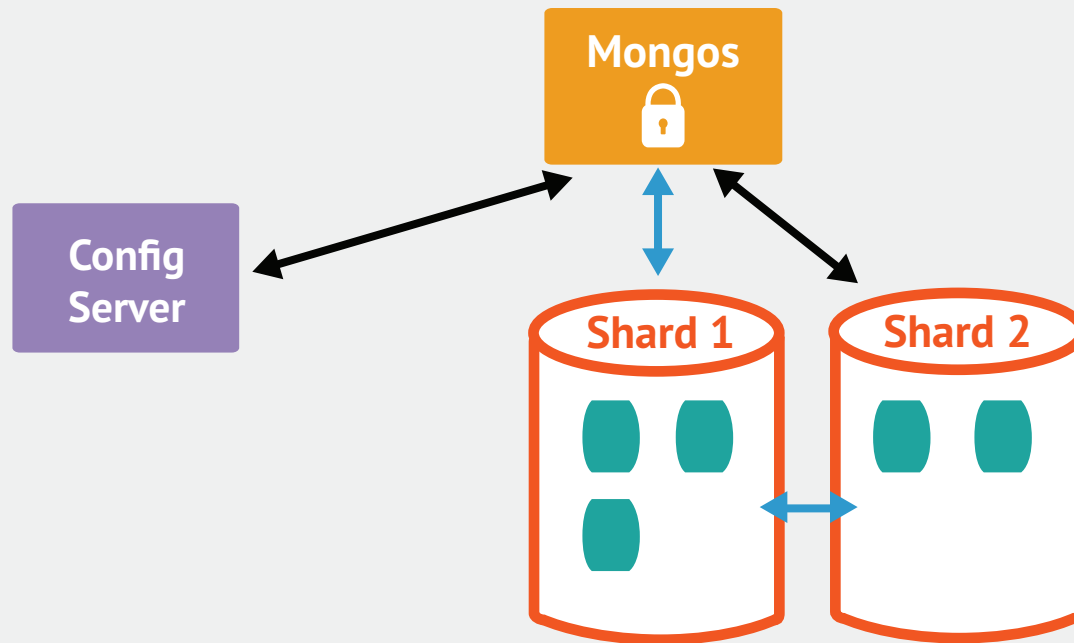


The balancer on mongos takes out a “balancer lock”

To see the status of these locks:

```
use config  
db.locks.find({ _id: "balancer" })
```

# Moving the chunk

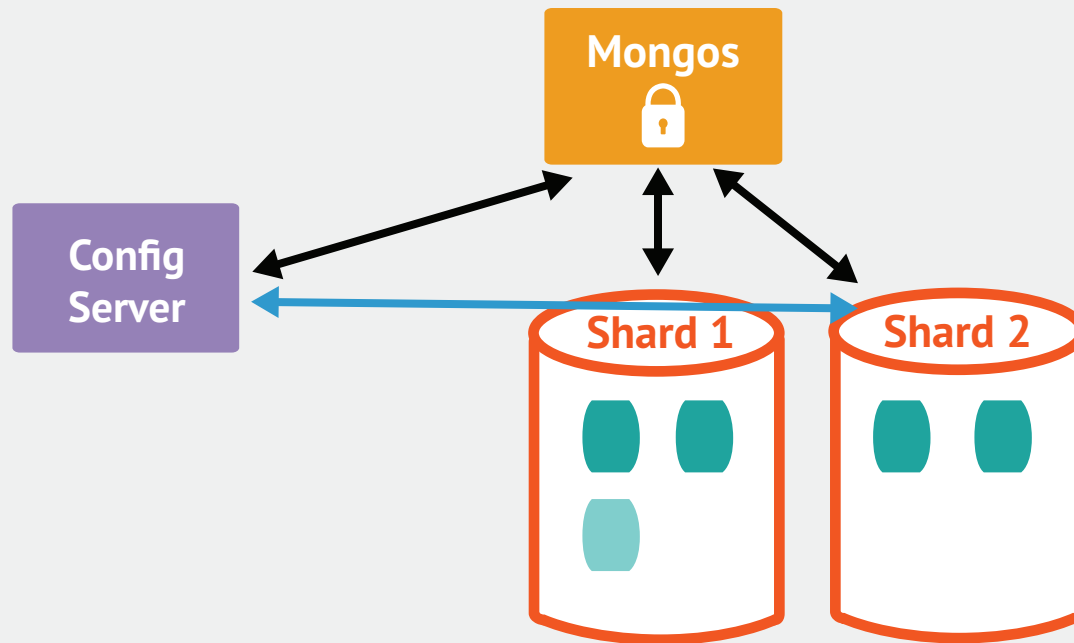


The mongos sends a `moveChunk` command to source shard

The source shard then notifies destination shard

Destination shard starts pulling documents from source shard

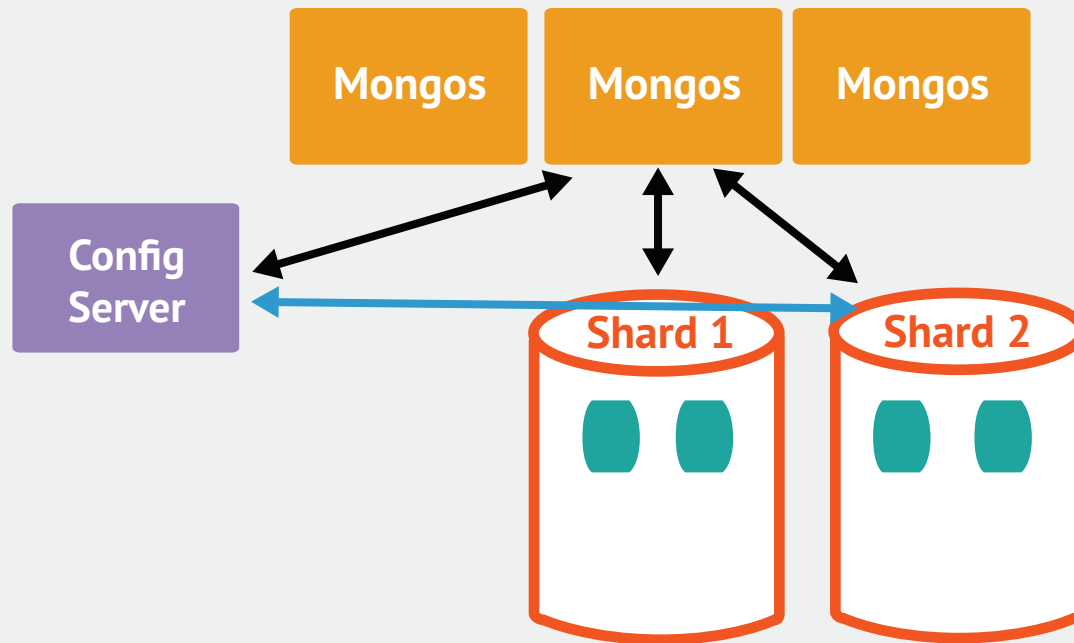
# Committing Migration



When complete, destination shard updates config server

- Provides new locations of the chunks

# Cleanup



Source shard deletes moved data

- Must wait for open cursors to either close or time out
- NoTimeout cursors may prevent the release of the lock

The mongos releases the balancer lock after old chunks are deleted

# Routing Requests

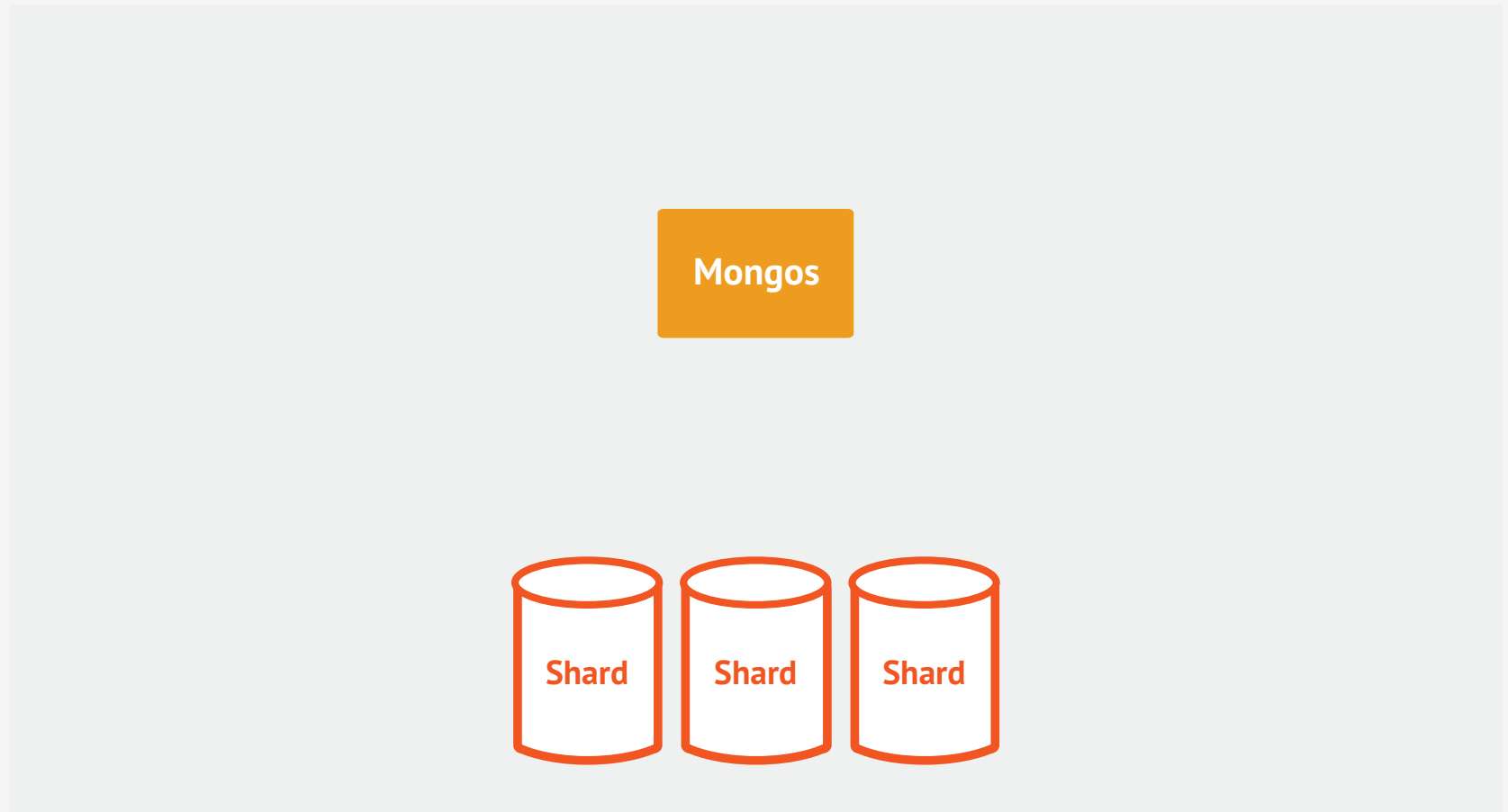


# Cluster Request Routing

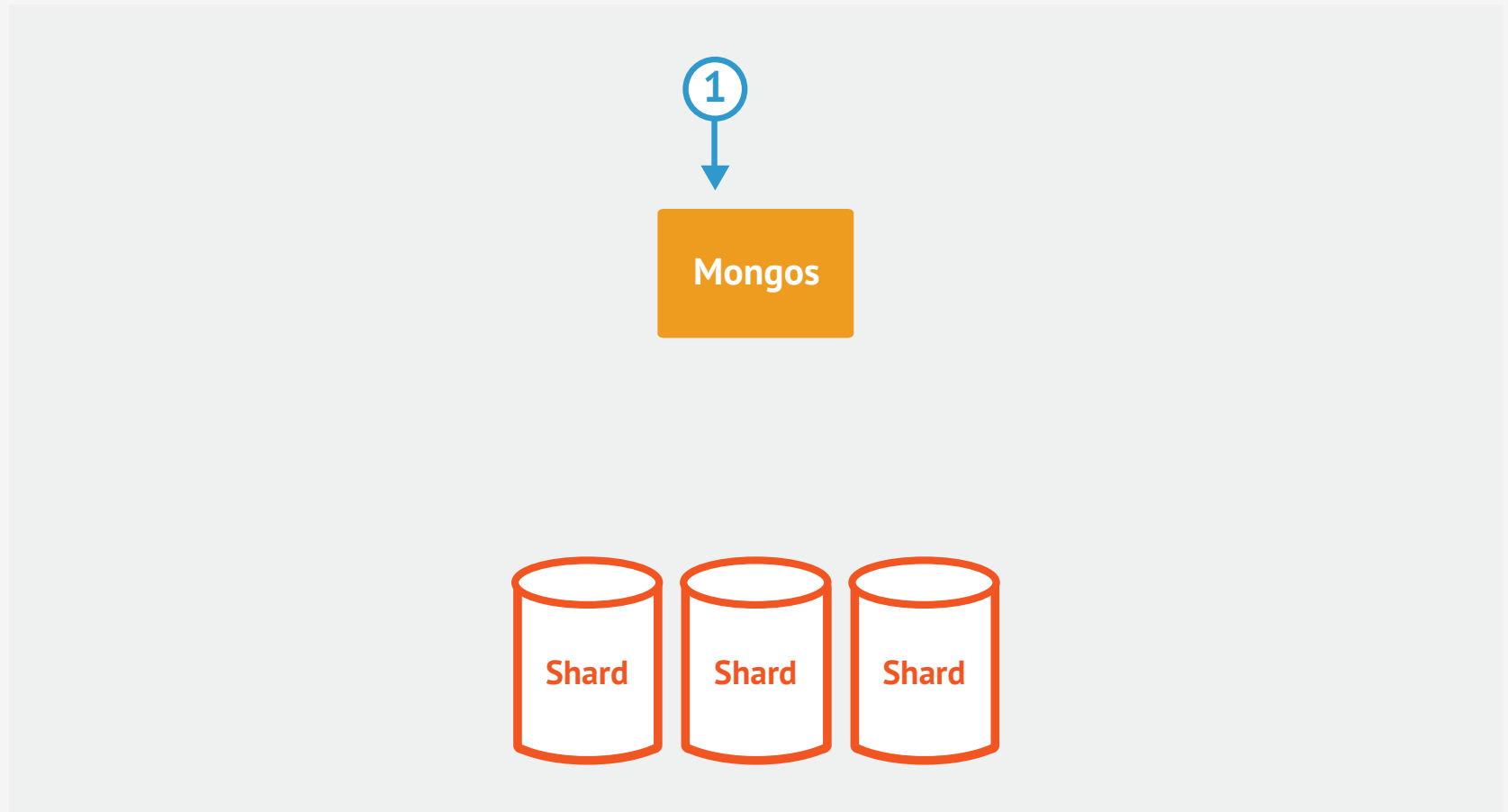
- Targeted Queries
- Scatter Gather Queries
- Scatter Gather Queries with Sort



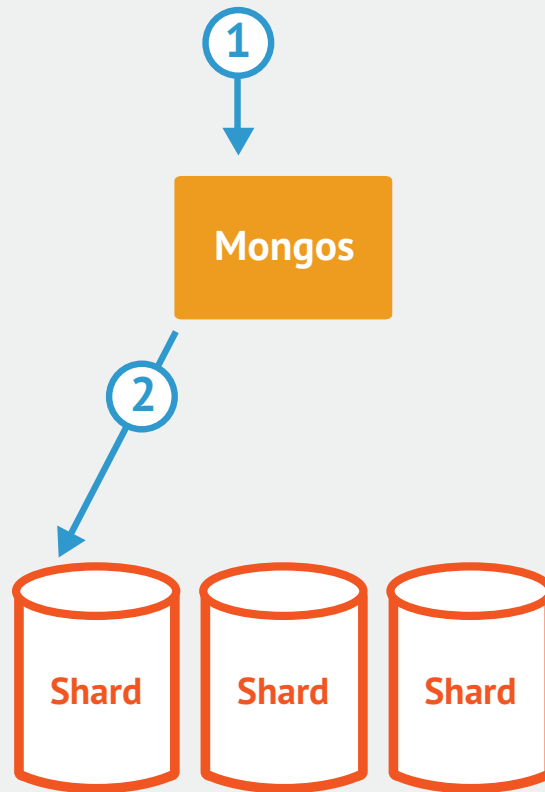
# Cluster Request Routing: Targeted Query



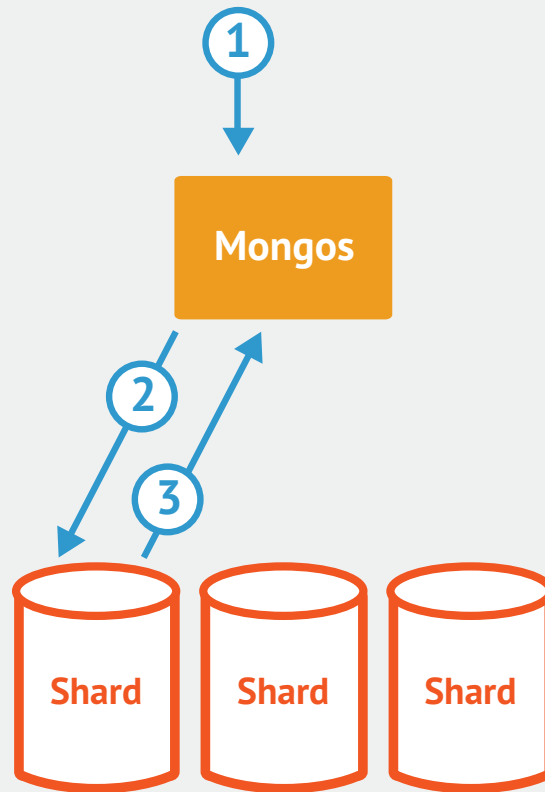
# Routable request received



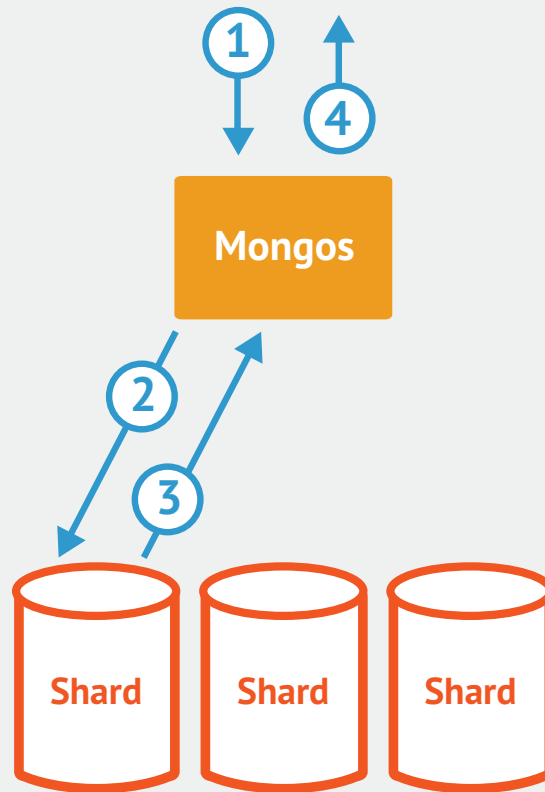
# Request routed to appropriate shard



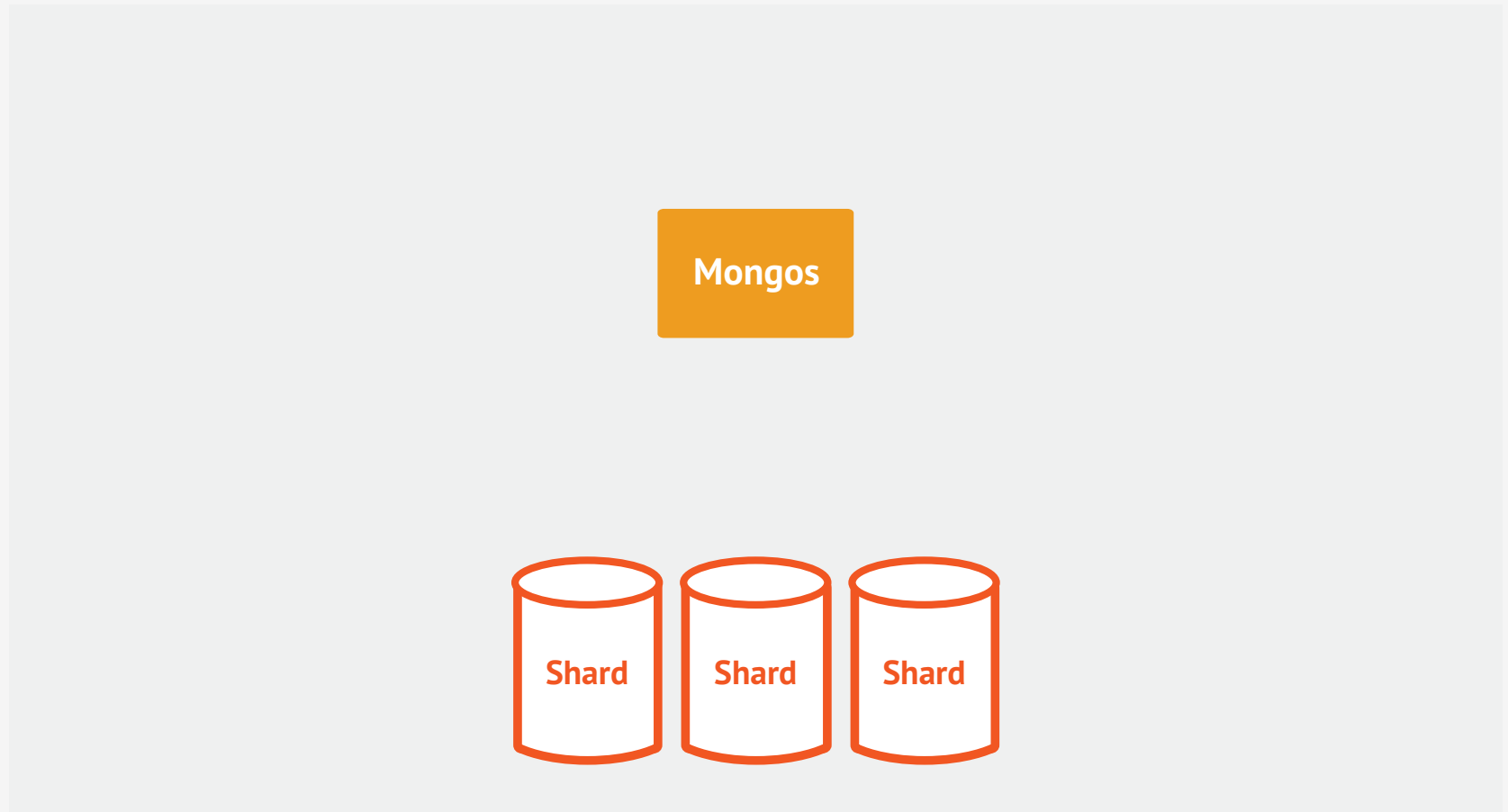
# Shard returns results



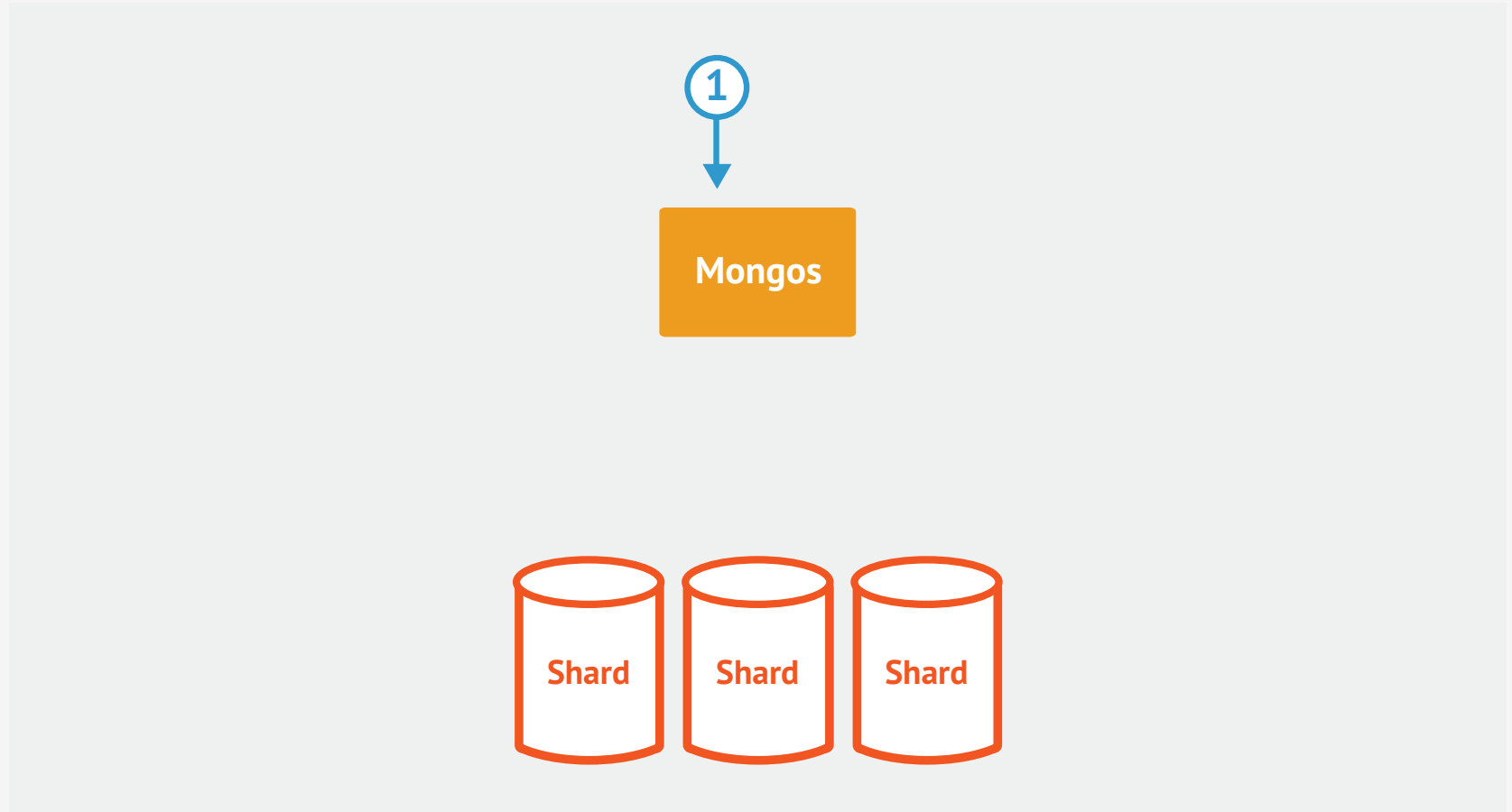
# Mongos returns results to client



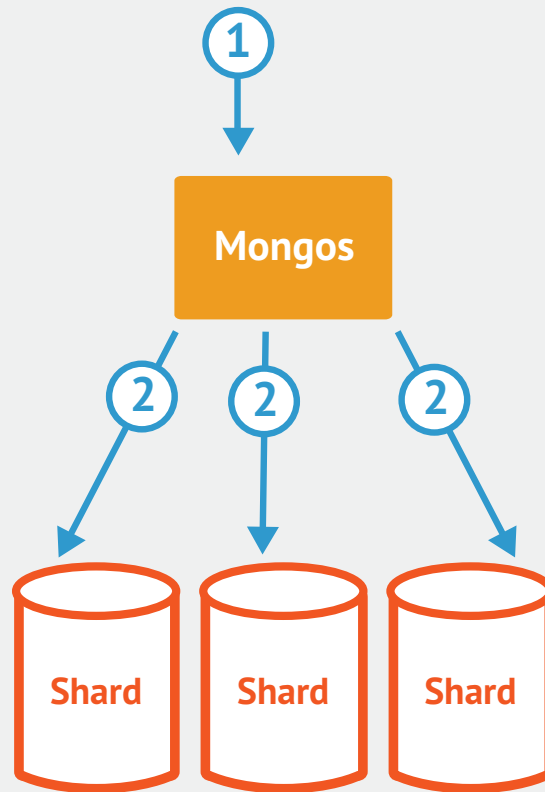
# Cluster Request Routing: Non-Targeted Query



# Non-Targeted Request Received

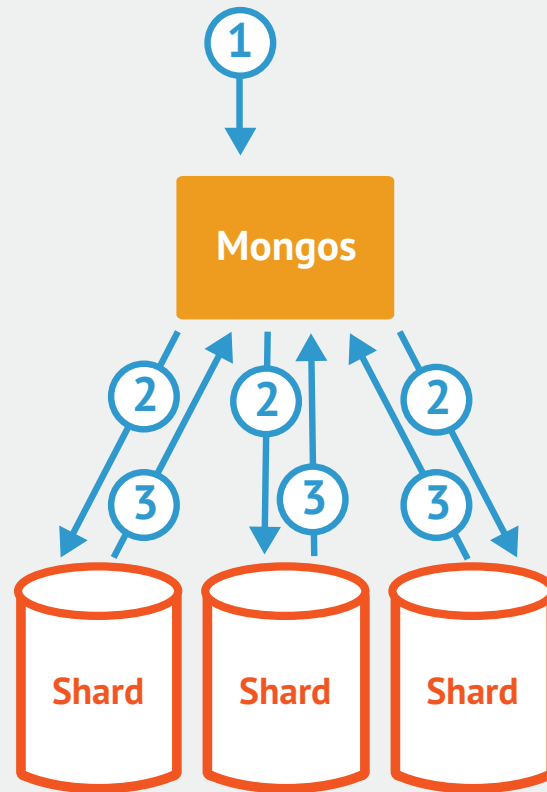


# Request sent to all shards

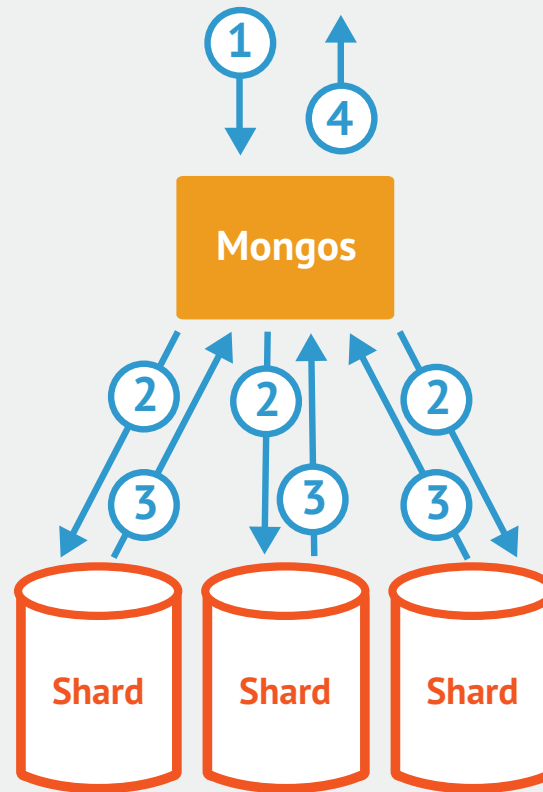




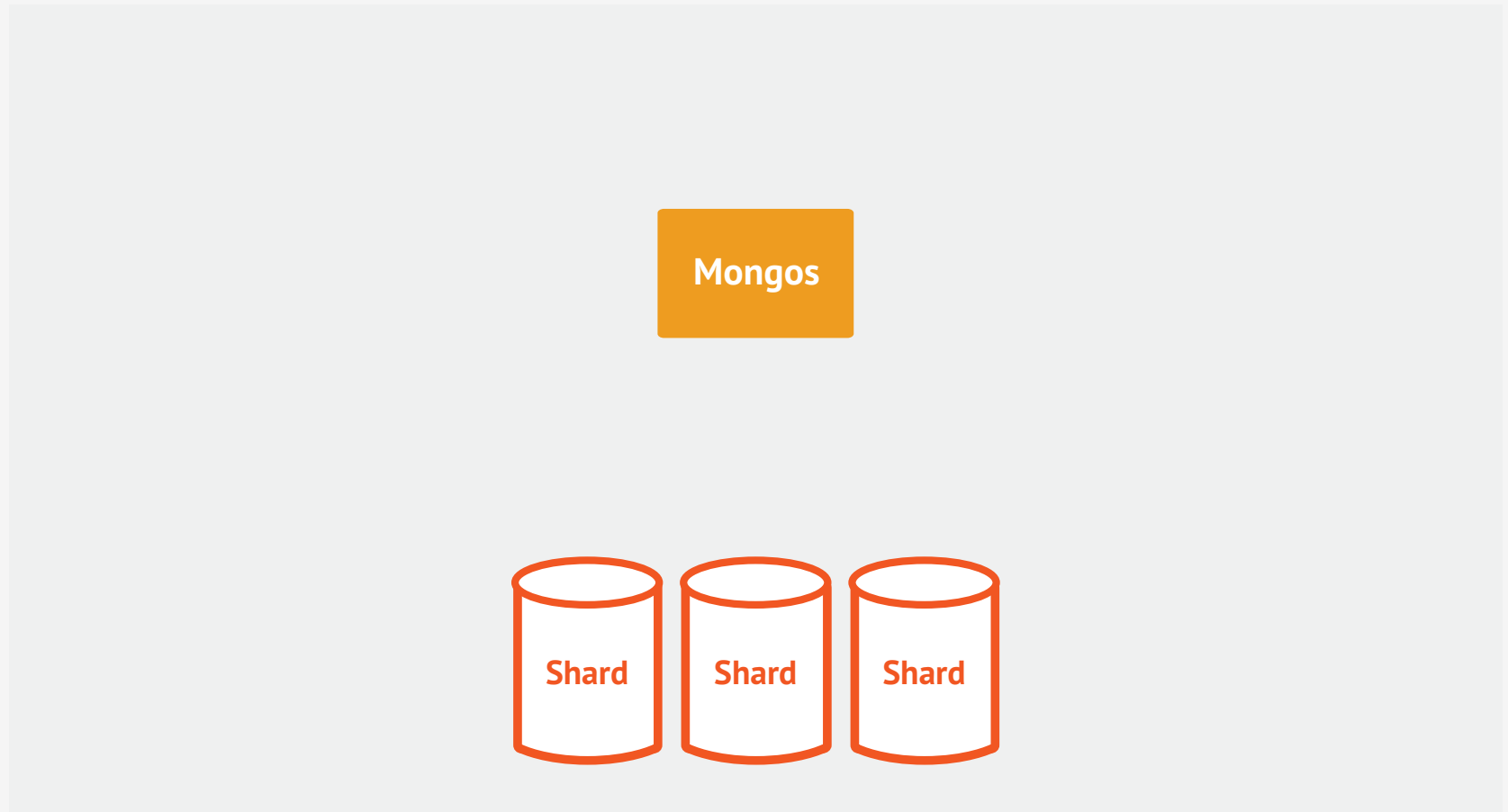
# Shards return results to mongos



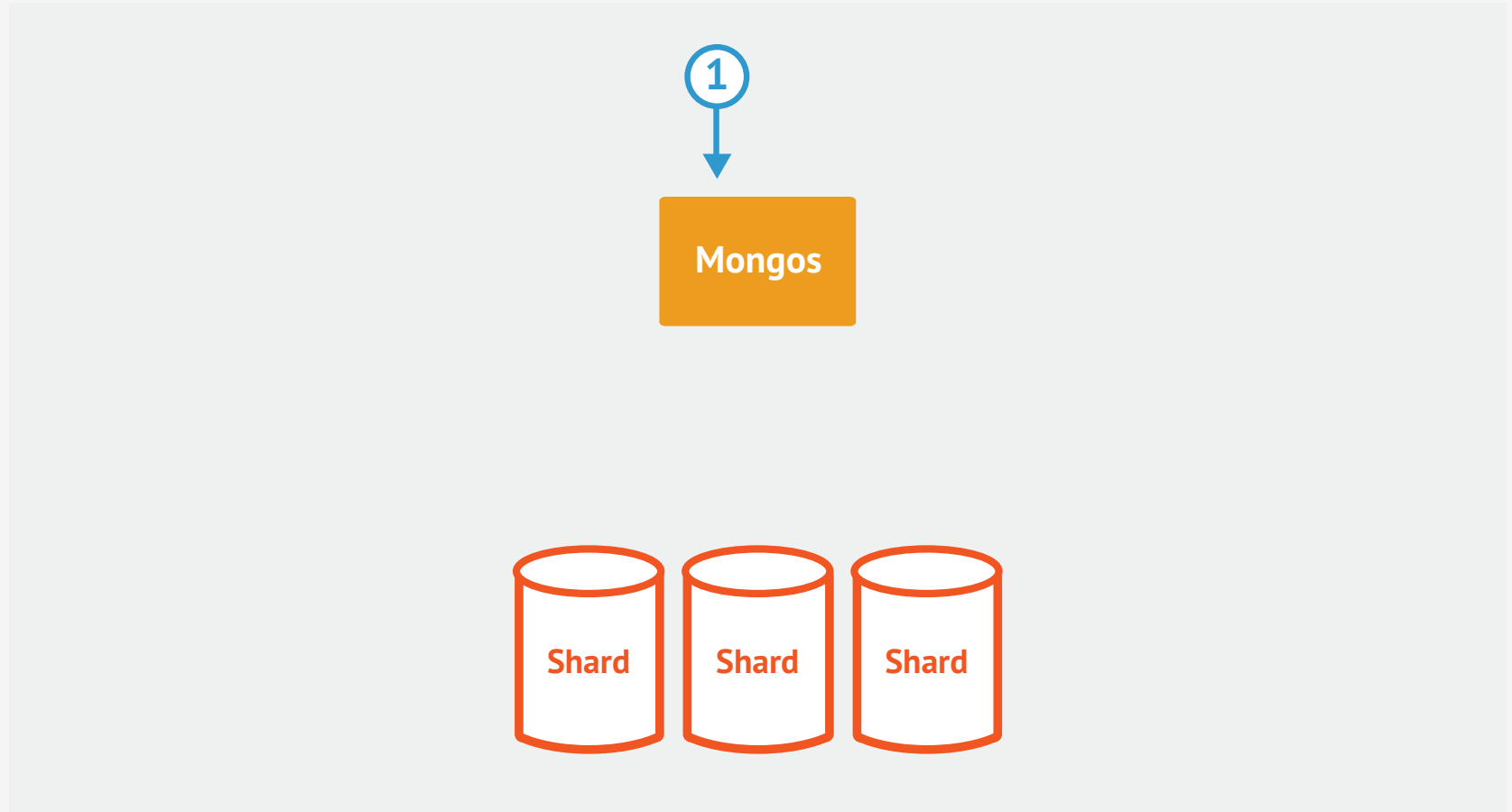
# Mongos returns results to client



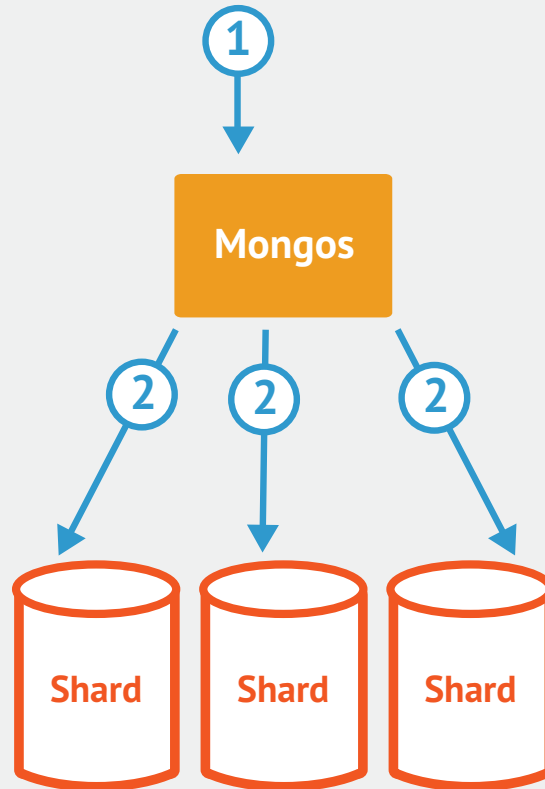
# Cluster Request Routing: Non-Targeted Query with Sort



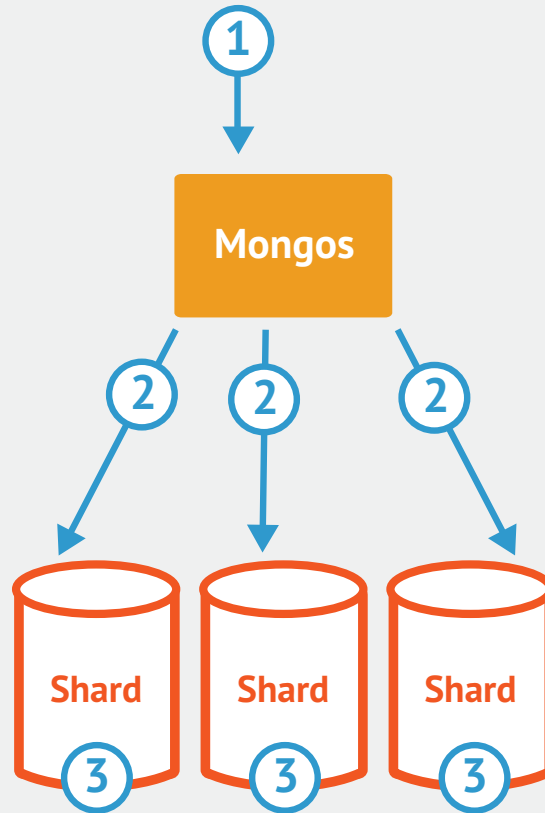
# Non-Targeted request with sort received



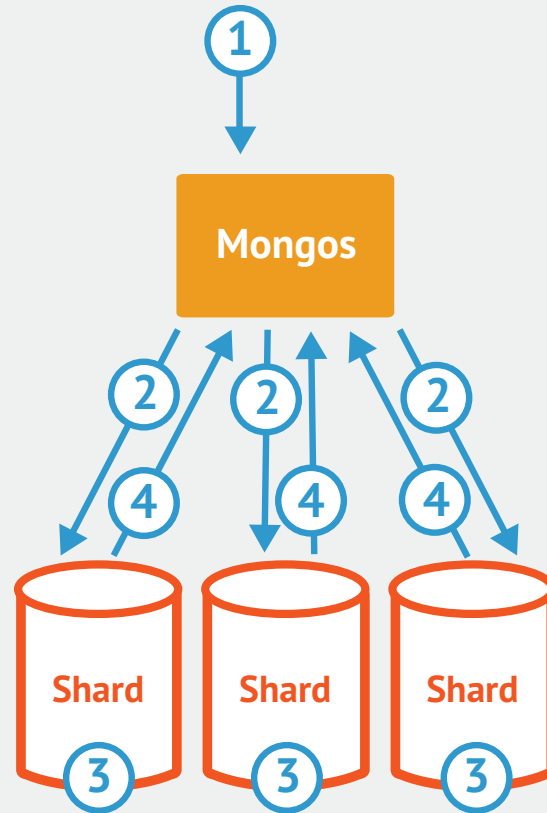
# Request sent to all shards



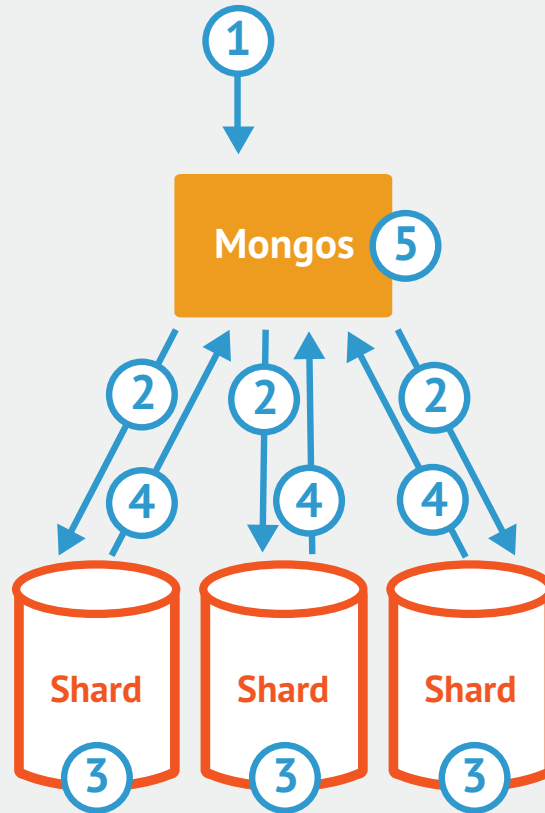
# Query and sort performed locally



# Shards return results to mongos

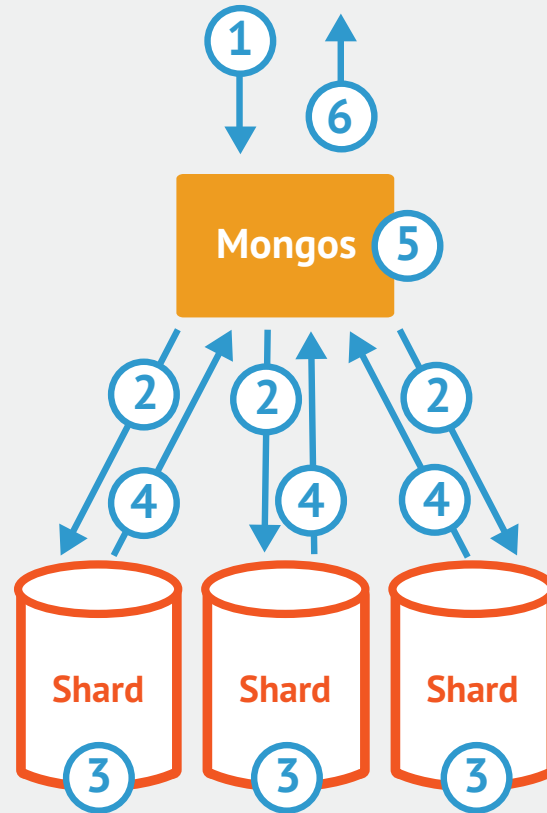


# Mongos merges sorted results





# Mongos returns results to client



# Shard Key



# Shard Key

- Shard key is **immutable**
- Shard key values are **immutable**
- Shard key must be indexed
- Shard key limited to 512 bytes in size
- Shard key used to route queries
  - Choose a field commonly used in queries
- Only shard key can be unique across shards
  - `\_id` field is only unique within individual shard

# Shard Key Considerations

- Cardinality
- Write Distribution
- Query Isolation
- Reliability
- Index Locality

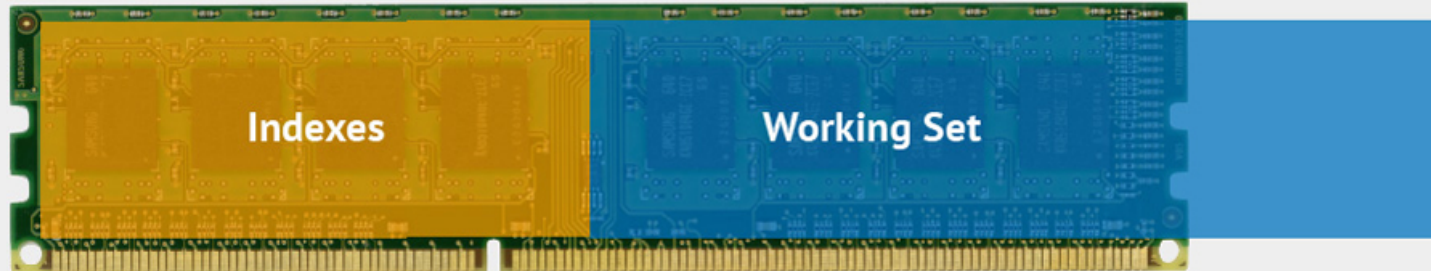
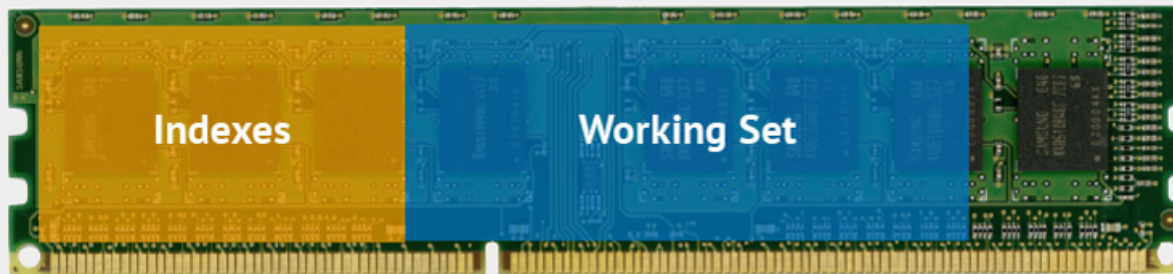
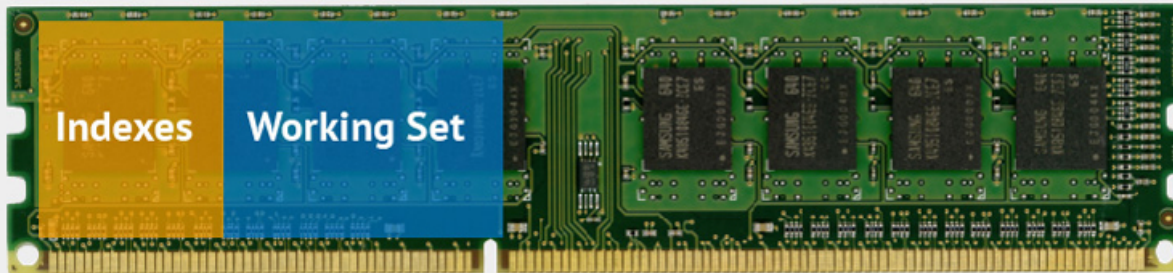
# Conclusion



# Read/Write Throughput Exceeds I/O



# Working Set Exceeds Physical Memory



# Sharding Enables Scale

- MongoDB's Auto-Sharding
  - Easy to Configure
  - Consistent Interface
  - Free and Open Source



- What's next?

- [ Insert Related Talks ]
- [ Insert Upcoming Webinars ]
- MongoDB User Group

- Resources

<https://education.10gen.com/>

<http://www.10gen.com/presentations>



#hashtag

# Thank You

Speaker Name

*Software Engineer, 10gen*