MongoDB Clustering/Sharding Lab

Comparing Single Instance vs. Sharded Cluster Performance on Big Data

(Paste your Screenshots and relevant logs in the very end)

Objective

Measure and compare the performance of a single MongoDB instance against a sharded cluster using a 100,000 document dataset.

Verify your setup:

docker --version

docker-compose --version

Lab files structure:

mongodb-lab/

├── docker-compose.single.yml

├── docker-compose.cluster.yml

└── scripts/

** ├── init_single.py**

** └── init_cluster.py**

Ensure docker-compose files are in root and Python scripts are in the `scripts/` folder.

Architecture Overview

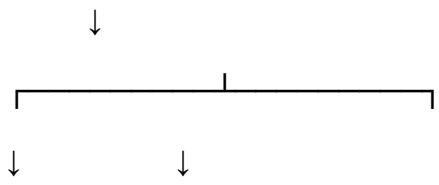
Single Instance Setup:

Application → MongoDB → All Data

Drawback: Single point of failure, limited by one machine's resources.

Sharded Cluster Setup:

Application → Mongos Router



Key components:

- **Mongos: Query router directing traffic to appropriate shards**
- **Config Servers: Metadata storage (3-node replica set)**
- **Shards: Data storage nodes (3-node replica sets)**
- **Shard Key: Distribution field (userId, hashed)**

Experiment 1: Single Instance Baseline

Run a single MongoDB server and benchmark insert, read, update, and aggregation operations.

 # Start single instance

docker-compose -f docker-compose.single.yml up

 # View output (wait 3-5 minutes for completion)

 # Check results

docker logs init-single

 # Save for comparison

docker logs init-single > single_results.txt

 # Stop and cleanup

docker-compose -f docker-compose.single.yml down -v

Record these metrics:

- **Insert throughput (ops/sec)**
- **Read throughput (queries/sec)**
- **Update throughput (ops/sec)**

- **Aggregation speed (docs/sec)**

Experiment 2: Sharded Cluster

Deploy a 10-container cluster (3 config servers, 6 shard nodes, 1 router, 1 initialization script).

 # Start cluster

docker-compose -f docker-compose.cluster.yml up

 # The initialization process:

 # - Sets up config server replica set

 # - Initializes shard A (3 nodes)

 # - Initializes shard B (3 nodes)

 # - Registers shards with cluster

 # - Enables sharding on database

 # - Runs performance benchmark

 # View results

docker logs init-cluster

 # Save results

docker logs init-cluster > cluster_results.txt

 # Cleanup

docker-compose -f docker-compose.cluster.yml down -v

Note the following:

- **Replica set initialization sequence**
- **Data split between shards (target: 50/50)**
- **Performance differences from single instance**

Cluster Inspection (Required)

 # Start cluster in detached mode

docker-compose -f docker-compose.cluster.yml up -d

sleep 60

 # View cluster configuration

docker exec -it mongos mongo --eval "sh.status()"

 # Check shard A status

docker exec -it shardA1 mongo --port 27018 --eval "rs.status()"

 # Check shard B status

docker exec -it shardB1 mongo --port 27018 --eval "rs.status()"

 # Interactive shell

docker exec -it mongos mongo

 # Inside mongo shell:

use analyticsDB

db.events.count()

db.events.getShardDistribution()

db.events.find({userId: 42}).limit(5).pretty()

exit

 # OR execute commands directly from terminal:

docker exec -it mongos mongo analyticsDB --eval "db.events.count()"

docker exec -it mongos mongo analyticsDB --eval "db.events.getShardDistribution()"

docker exec -it mongos mongo analyticsDB --eval "db.events.find({userId: 42}).limit(5).pretty()"

Command reference:

- `sh.status()` **- Cluster configuration and shard info**
- `rs.status()` **- Replica set member status (PRIMARY/SECONDARY roles)**
- `getShardDistribution()` **- Document distribution across shards**
- `count()` **- Total document count**

Results Analysis : Fill this up and submit along side the exercises

Fill in your measurements:

Operation	Single Instance	Cluster	Improvement
Insert (ops/sec)			**%**
Read (qps)			**%**
Update (ops/sec)			**%**
Aggregation (docs/sec)			**%**

Extract data from logs:

grep "ops/sec\|qps\|docs/sec" single_results.txt

grep "ops/sec\|qps\|docs/sec" cluster_results.txt

Typical improvements observed:

- **Insert: 40-60% (parallel disk writes)**
- **Read: 30-40% (distributed load)**
- **Update: 40-60% (reduced lock contention)**
- **Aggregation: 40-60% (parallel computation)**

Check data distribution:

grep "DATA DISTRIBUTION" cluster_results.txt -A 5

Expected result: approximately 50,000 documents per shard.

Performance Factors

1. Parallel Write Operations

- **Single: 100K inserts to one disk**
- **Cluster: 50K to shard A + 50K to shard B concurrently**

2. Read Load Distribution

- **Single: All queries hit one server**
- **Cluster: Queries spread across 6 nodes**

3. Write Lock Management

- **Single: All updates queue for single write lock**
- **Cluster: Updates distributed, reducing contention per shard**

4. Aggregation Pipeline

- **Single: One CPU processes 100K documents**
- **Cluster: Each shard processes 50K in parallel, mongos merges results**

Hashed Shard Key Implementation:

key={'userId': 'hashed'}

The hash function distributes documents evenly regardless of userId values. This prevents scenarios where sequential IDs would all land on the same shard.

Replica Set Behavior:

- **Each shard: 1 PRIMARY (accepts writes) + 2 SECONDARY (replicas)**
- **Automatic failover if PRIMARY becomes unavailable**
- **Election time: typically under 30 seconds**

Cleanup Procedure

docker-compose -f docker-compose.single.yml down -v

docker-compose -f docker-compose.cluster.yml down -v

docker ps -a | grep mongo

docker rm -f $(docker ps -aq --filter "name=mongo")

docker system prune -f

Lab Questions

Q1: Performance Analysis Which operation showed the greatest performance gain in the cluster? Explain why based on the architecture differences.

Q2: Data Distribution What problems would occur if shard A contained 90% of documents instead of 50%?

Q3: Shard Key Design Analyze these alternative shard key choices:

- **Timestamp field**
- **Boolean field**
- **Auto-incrementing integer**

What issues would each introduce?

Q4: Scaling Strategy The dataset will grow from 100K to 10M documents next year, with 10x traffic increase. Propose a scaling plan.

Q5: Failover Test Execute this failover scenario:

docker-compose -f docker-compose.cluster.yml up -d

sleep 60

docker exec -it shardA1 mongo --port 27018 --eval "rs.status()"

docker kill shardA1

sleep 30

docker exec -it shardA2 mongo --port 27018 --eval "rs.status()"

docker exec -it mongos mongo --eval "db.getSiblingDB('analyticsDB').events.count()"

docker start shardA1

Measure the downtime and describe shardA1's role after restart.

Q6: Query Routing Compare these queries using `.explain("executionStats")`:

You can directly run from terminal too:

db.events.find({userId: 42}) // targeted

db.events.find({eventType: "click"}) // scatter-gather

How many shards does each query hit?

Q7: Bottleneck Analysis Identify the primary bottleneck in the single instance (CPU/disk/RAM/locks). Cite specific log evidence.

Q8: Infrastructure Economics The cluster uses 10x the containers of single instance. In what scenarios does this cost multiply justify itself?

Command Reference

 # Single instance workflow

docker-compose -f docker-compose.single.yml up

docker logs init-single

docker-compose -f docker-compose.single.yml down -v

 # Cluster workflow

docker-compose -f docker-compose.cluster.yml up

docker logs init-cluster

docker exec -it mongos mongo --eval "sh.status()"

docker-compose -f docker-compose.cluster.yml down -v

 # System cleanup

docker system prune -a -f

End of Lab

```
"protocol":"op_msg","durationMillis":139}}
init-single      |
init-single      | ======================================================================
init-single      | SINGLE MONGODB INSTANCE - PERFORMANCE BASELINE
init-single      | ======================================================================
init-single      | Architecture: Single mongod process
init-single      | URI: mongodb://mongo-single:27017
init-single      | Dataset: 100,000 documents (simulating big data workload)
init-single      | ======================================================================
init-single      |
init-single      | ✓ Connected to MongoDB
init-single      | ✓ Collection cleaned
init-single      |
init-single      | [1/4] INSERT TEST: 100,000 documents
init-single      | ----------------------------------------------------------------------
init-single      | Simulating: Real-time event streaming (clicks, purchases, views)
init-single      |   Progress: 20,000 docs | Current rate: 85,047 ops/sec
init-single      |   Progress: 40,000 docs | Current rate: 90,668 ops/sec
init-single      |   Progress: 60,000 docs | Current rate: 88,747 ops/sec
init-single      |   Progress: 80,000 docs | Current rate: 89,601 ops/sec
init-single      |   Progress: 100,000 docs | Current rate: 88,892 ops/sec
init-single      |
init-single      | ----------------------------------------------------------------------
init-single      | ✓ INSERT COMPLETED
init-single      |   Time taken:              1.12 seconds
init-single      |   Throughput:            88,890 ops/sec
init-single      |   Total documents:      100,000
init-single      |   Bottleneck:        Single node disk I/O + CPU
init-single      | ----------------------------------------------------------------------
init-single      |
init-single      | [2/4] READ TEST: 10,000 targeted queries (by userId)
init-single      | ----------------------------------------------------------------------
init-single      | Simulating: User activity lookups, analytics dashboards
init-single      |   Progress: 2,000 queries | Current QPS:    242
init-single      |   Progress: 4,000 queries | Current QPS:    235
init-single      |   Progress: 6,000 queries | Current QPS:    237
```

```
✓ READ COMPLETED
  Time taken:             39.63 seconds
  Query rate:               252 queries/sec
  Avg latency:             3.96 ms/query
  Advantage:        Load balanced across replica sets
```

```
[3/4] UPDATE TEST: 10,000 random updates
------------------------------------------------------------------------
Benefit: Updates distributed to respective shard primaries
  Progress: 2,000 updates | Current rate:  1,758 ops/sec
  Progress: 4,000 updates | Current rate:  1,684 ops/sec
  Progress: 6,000 updates | Current rate:  1,585 ops/sec
  Progress: 8,000 updates | Current rate:  1,510 ops/sec
  Progress: 10,000 updates | Current rate:  1,473 ops/sec
```

```
✓ UPDATE COMPLETED
  Time taken:              6.79 seconds
  Update rate:            1,473 ops/sec
  Advantage:        No single-node write lock contention
```

```
[4/4] AGGREGATION TEST: Complex analytics query
------------------------------------------------------------------------
Benefit: Parallel processing across all shards
  Computed: Event type x Device breakdown
  Records processed: 100,000 (across 2 shards)
  Results returned:  15
```

```
✓ AGGREGATION COMPLETED
  Time taken:              0.14 seconds
  Processing rate:     729,234 docs/sec
  Advantage:        Parallel aggregation pipeline on each shard
```

```
init-cluster    |
init-cluster    | ========================================================================
init-cluster    | MONGODB SHARDED CLUSTER - SETUP AND PERFORMANCE TEST
init-cluster    | ========================================================================
init-cluster    |
init-cluster    | [1/5] Initializing Config Server Replica Set (3 nodes)
init-cluster    | ------------------------------------------------------------------------
init-cluster    |   Connecting to mongodb://cfg1:27019...
init-cluster    |   ✓ Initiated replica set: cfgRepl
init-cluster    |   ✓ Primary elected in replica set
init-cluster    |
init-cluster    | [2/5] Initializing Shard A Replica Set (3 nodes)
init-cluster    | ------------------------------------------------------------------------
init-cluster    |   Connecting to mongodb://shardA1:27018...
init-cluster    |   ✓ Initiated replica set: shardA
init-cluster    |   ✓ Primary elected in replica set
init-cluster    |
init-cluster    | [3/5] Initializing Shard B Replica Set (3 nodes)
init-cluster    | ------------------------------------------------------------------------
init-cluster    |   Connecting to mongodb://shardB1:27018...
init-cluster    |   ✓ Initiated replica set: shardB
init-cluster    |   ✓ Primary elected in replica set
init-cluster    |
init-cluster    | [4/5] Adding Shards to Cluster via Mongos Router
init-cluster    | ------------------------------------------------------------------------
init-cluster    |   Connecting to mongos: mongodb://mongos:27017
init-cluster    |   ✓ Added shardA to cluster
init-cluster    |   ✓ Added shardB to cluster
init-cluster    |
init-cluster    | [5/5] Configuring Database Sharding
init-cluster    | ------------------------------------------------------------------------
init-cluster    |   ✓ Enabled sharding on database: analyticsDB
init-cluster    |   ✓ Sharded collection on userId (hashed)
init-cluster    |
init-cluster    | ✓ Cluster setup complete!
init-cluster    |
init-cluster    | ========================================================================
init-cluster    | SHARDED CLUSTER - PERFORMANCE TEST
init-cluster    | ========================================================================
```

```
[3/4] UPDATE TEST: 10,000 updates
--------------------------------------------------------------------
  Progress: 2,000 updates | Rate:    589 ops/sec
  Progress: 4,000 updates | Rate:    579 ops/sec
  Progress: 6,000 updates | Rate:    581 ops/sec
  Progress: 8,000 updates | Rate:    577 ops/sec
  Progress: 10,000 updates | Rate:    578 ops/sec

✓ UPDATE COMPLETED
  Time: 17.31s | Throughput: 578 ops/sec

[4/4] AGGREGATION TEST
--------------------------------------------------------------------
  Processed: 100,000 docs
  Results: 15 groups

✓ AGGREGATION COMPLETED
  Time: 0.13s | Rate: 748,066 docs/sec

====================================================================
CLUSTER RESULTS SUMMARY
====================================================================
Insert:          56,252 ops/sec
Read:               231 queries/sec
Update:             578 ops/sec
Aggregation:    748,066 docs/sec
====================================================================
```

```
MongoDB server version: 4.4.29
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("6922b0a5a5e7a5c8a000fa56")
  }
  shards:
        {  "_id" : "shardA",  "host" : "shardA/shardA1:27018,shardA2:27018,shardA3:27018",  "state" :
        {  "_id" : "shardB",  "host" : "shardB/shardB1:27018,shardB2:27018,shardB3:27018",  "state" :
  active mongoses:
        "4.4.29" : 1
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                512 : Success
  databases:
        {  "_id" : "analyticsDB",  "primary" : "shardA",  "partitioned" : true,  "version" : {  "uuid
 : Timestamp(1763881152, 5) }
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shardA  512
                                shardB  512
                        too many chunks to print, use verbose if you want to force print
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c16c1db7-05b6-4d5a-a106-cbf862148c4c") }
MongoDB server version: 4.4.29
{
        "set" : "shardA",
        "date" : ISODate("2025-11-23T07:26:47.293Z"),
```