

Міністерство освіти і науки України  
НТУУ «Київський політехнічний інститут»  
Фізико-технічний інститут

**Проектування високонавантажених систем**  
**Лабораторна робота №4**  
Налаштування реплікації та перевірка відмовостійкості MongoDB

**Виконав:**  
Студент 4-го курсу  
групи ФІ-21  
Климент'єв Максим  
**Перевірив:**

---

# **Зміст**

<b>1</b>	<b>Код реалізації</b>	<b>3</b>
<b>2</b>	<b>Результати</b>	<b>5</b>

# 1 Код реалізації

```
import time
import subprocess
from multiprocessing import Process, Pool
from pymongo import MongoClient, WriteConcern
from pymongo.read_concern import ReadConcern
from pymongo.read_preferences import ReadPreference
from pymongo.errors import AutoReconnect, ConnectionFailure, WTimeoutError, ServerSelectionTimeoutError, NetworkTimeout

# URI = "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?replicaSet=replicaset"
URI = "mongodb://localhost:27017,localhost:27018,localhost:27019/?replicaSet=replicaset"

def sh(command: str):
    print(command)
    subprocess.run(command, shell=True)

def preprocess_db(info: bool = False):
    with MongoClient(URI) as client:
        try:
            client.admin.command("ping")
            database = client["testdb"]
            if database.get_collection("likes") is not None:
                database.drop_collection("likes")
            database.create_collection("likes")

            collection = database.get_collection("likes")
            collection.insert_one({"_id": "1", "likes": 0})

            if info:
                print(collection)
                print(client.primary)
                print(client.nodes)
                print(get_topology_status())
        except WTimeoutError:
            print(f"[TIMEOUT] {preprocess_db.__name__} WTimeoutError")
        except ServerSelectionTimeoutError:
            print(f"[TIMEOUT] {preprocess_db.__name__} ServerSelectionTimeoutError")

def client_work(write_concern, timeout, read_concern=None):
    with MongoClient(URI, timeoutMS=timeout) as client:
        try:
            collection = client["testdb"].get_collection("likes", write_concern=write_concern, read_concern=read_concern)
            collection.find_one_and_update({"_id": "1"}, {"$inc": {"likes": 1}})
        except WTimeoutError:
            print(f"[TIMEOUT] {client_work.__name__} WTimeoutError")
        except ServerSelectionTimeoutError:
            print(f"[TIMEOUT] {client_work.__name__} ServerSelectionTimeoutError")
        except NetworkTimeout:
            print(f"[TIMEOUT] {client_work.__name__} NetworkTimeout")

def receive_result(read_concern=None):
    with MongoClient(URI) as client:
        try:
            collection = client["testdb"].get_collection("likes", read_concern=read_concern)
            print(collection.find_one({"_id": "1"}))
        except WTimeoutError:
            print(f"[TIMEOUT] {receive_result.__name__} WTimeoutError")
        except ServerSelectionTimeoutError:
            print(f"[TIMEOUT] {receive_result.__name__} ServerSelectionTimeoutError")

def get_topology_status():
    with MongoClient(URI) as client:
        try:
            while True:
                try:
                    status = client.admin.command("hello")
                    return status.get("primary"), status.get("me")
                except (AutoReconnect, ConnectionFailure):
                    print("Primary Election still going")
                    time.sleep(1)
        except WTimeoutError:
            print(f"[TIMEOUT] [{get_topology_status.__name__}] WTimeoutError")
        except ServerSelectionTimeoutError:
            print(f"[TIMEOUT] [{get_topology_status.__name__}] ServerSelectionTimeoutError")

def task(welcome_text, node_to_stop, node_type, timeout=None, read_concern=None):
```

```

print(f"[INFO] [{node_type}] {welcome_text}")
print(f"[INFO] [START RESULT] Right before stoping {node_type}...")
receive_result(read_concern)
processes = []
sh(f"docker stop {node_to_stop}")
for i in range(1):
    process = Process(target=client_work, args=(WriteConcern(w=3), timeout))
    process.start()
    processes.append(process)

print(f"[INFO] [MID RESULT] Right after stoping {node_type}...")
receive_result(read_concern)
time.sleep(10)
print(f"[INFO] [MID RESULT] After 10 seconds stoping {node_type}...")
receive_result(read_concern)
sh(f"docker start {node_to_stop}")

for process in processes:
    process.join()

print("[INFO] [FINAL RESULT]")
receive_result(read_concern)

if __name__ == "__main__":
    preprocess_db(True)
    text = "Starting 1 client with w=3 write concern and no timeout..."
    task(text, "mongol", "PRIMARY")
    task(text, "mongo3", "SECONDARY")

    # preprocess_db(True)
    text = "Starting 1 client with w=3 write concern and 'majority' read concern, but timeout = 3000..."
    task(text, "mongol", "PRIMARY")
    task(text, "mongo3", "SECONDARY", read_concern=ReadConcern("majority"), timeout=3000)

    direct_uri = f"mongodb://localhost:27017/?directConnection=true"
    print(f"[VERIFY] Checking local data directly on mongol...")
    try:
        with MongoClient(direct_uri) as client:
            db = client.get_database("testdb", read_preference=ReadPreference.SECONDARY_PREFERRED)
            doc = db["likes"].find_one({"_id": "1"})
            print(doc)
    except Exception as e:
        print(f"[VERIFY ERROR] Could not read from mongol: {e}")

```

## 2 Результати

### I Налаштування реплікації

1. Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S) (всі ноди можуть бути запущені як окремі процеси або у Docker контейнерах)

```
replicaset [direct: primary] test> rs.status()
{
  set: 'replicaset',
  date: ISODate('2025-12-14T14:45:33.199Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-12-14T14:45:31.010Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
    writtenOpTime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-12-14T14:45:31.010Z'),
    lastDurableViewOpTime: ISODate('2025-12-14T14:45:31.010Z'),
    lastWrittenWallTime: ISODate('2025-12-14T14:45:31.010Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1765723511, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-12-14T14:35:30.954Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1765722919, i: 1 }), t: Long('-1') },
    lastSeenWrittenOpTimeAtElection: { ts: Timestamp({ t: 1765722919, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1765722919, i: 1 }), t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 2,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2025-12-14T14:35:31.003Z'),
    wMajorityWriteAvailabilityDate: ISODate('2025-12-14T14:35:31.473Z')
  },
  members: [
    {
      _id: 0,
      name: 'mongo1:27017',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 620,
      optime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2025-12-14T14:45:31.000Z'),
      optimeWritten: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
      optimeWrittenDate: ISODate('2025-12-14T14:45:31.000Z'),
      lastAppliedWallTime: ISODate('2025-12-14T14:45:31.010Z'),
      lastDurableViewOpTime: ISODate('2025-12-14T14:45:31.010Z'),
      lastWrittenWallTime: ISODate('2025-12-14T14:45:31.010Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1765722930, i: 1 }),
      electionDate: ISODate('2025-12-14T14:35:30.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'mongo2:27017',
      health: 1,
    }
  ]
}
```

```

state: 2,
stateStr: 'SECONDARY',
uptime: 613,
optime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
optimeDurability: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
optimeWritten: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-12-14T14:45:31.000Z'),
optimeDurabilityDate: ISODate('2025-12-14T14:45:31.000Z'),
optimeWrittenDate: ISODate('2025-12-14T14:45:31.000Z'),
lastAppliedWallTime: ISODate('2025-12-14T14:45:31.010Z'),
lastDurableWallTime: ISODate('2025-12-14T14:45:31.010Z'),
lastWrittenWallTime: ISODate('2025-12-14T14:45:31.010Z'),
lastHeartbeat: ISODate('2025-12-14T14:45:32.968Z'),
lastHeartbeatRecv: ISODate('2025-12-14T14:45:31.970Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
},
{
_id: 2,
name: 'mongo3:27017',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 613,
optime: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
optimeDurability: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
optimeWritten: { ts: Timestamp({ t: 1765723531, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-12-14T14:45:31.000Z'),
optimeDurabilityDate: ISODate('2025-12-14T14:45:31.000Z'),
optimeWrittenDate: ISODate('2025-12-14T14:45:31.000Z'),
lastAppliedWallTime: ISODate('2025-12-14T14:45:31.010Z'),
lastDurableWallTime: ISODate('2025-12-14T14:45:31.010Z'),
lastWrittenWallTime: ISODate('2025-12-14T14:45:31.010Z'),
lastHeartbeat: ISODate('2025-12-14T14:45:32.969Z'),
lastHeartbeatRecv: ISODate('2025-12-14T14:45:31.970Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
}
],
ok: 1,
'$clusterTime': {
clusterTime: Timestamp({ t: 1765723531, i: 1 }),
signature: {
hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAA='),
keyId: Long('0')
}
},
operationTime: Timestamp({ t: 1765723531, i: 1 })
}

```

ДАЛІ В МЕНЕ БУЛИ ПРОБЛЕМИ, ЯКІ Я ВЗАГАЛІ НЯК НЕ МІГ ПОФІКСИТИ, КОЛИ ВБИВАЮ PRIMARY НОДУ ЧОМУСЬ ПАЙТОН НІКОЛИ НЕ БАЧИТЬ, ЩО ВОНИ ВЖЕ ПЕРЕОБРАЛИСЬ

2. Спробувати зробити запис з однією відключеною нодою та write concern рівнем 3 та нескінченім таймаутом. Спробувати під час таймаута включити відключену ноду

```

[INFO] [PRIMARY] Starting 1 client with w=3 write concern and no timeout...
[INFO] [START RESULT] Right before stopping PRIMARY...
{'_id': '1', 'likes': 0}
docker stop mongo1
mongo1

```

```

[INFO] [MID RESULT] Right after stoping PRIMARY...
[TIMEOUT] receive_result ServerSelectionTimeoutError
[TIMEOUT] client_work ServerSelectionTimeoutError
[INFO] [MID RESULT] After 10 seconds stoping PRIMARY...
[TIMEOUT] receive_result ServerSelectionTimeoutError
docker start mongo1
mongo1
[INFO] [FINAL RESULT]
{'_id': '1', 'likes': 0}

[INFO] [SECONDARY] Starting 1 client with w=3 write concern and no timeout...
[INFO] [START RESULT] Right before stoping SECONDARY...
{'_id': '1', 'likes': 0}
docker stop mongo3
mongo3
[INFO] [MID RESULT] Right after stoping SECONDARY...
{'_id': '1', 'likes': 0}
[INFO] [MID RESULT] After 10 seconds stoping SECONDARY...
{'_id': '1', 'likes': 1}
docker start mongo3
mongo3
[INFO] [FINAL RESULT]
{'_id': '1', 'likes': 1}

```

3. Аналогічно попередньому пункту, але задати скінченний таймаут та дочекатись його закінчення. Перевірити чи данні записались і чи доступні на читання з рівнем readConcern: “majority”

```

[INFO] [PRIMARY] Starting 1 client with w=3 write concern and 'majority' read concern, but timeout = 3000...
[INFO] [START RESULT] Right before stoping PRIMARY...
{'_id': '1', 'likes': 1}
docker stop mongo1
mongo1
[INFO] [MID RESULT] Right after stoping PRIMARY...
[TIMEOUT] receive_result ServerSelectionTimeoutError
[TIMEOUT] client_work ServerSelectionTimeoutError
[INFO] [MID RESULT] After 10 seconds stoping PRIMARY...
[TIMEOUT] receive_result ServerSelectionTimeoutError
docker start mongo1
mongo1
[INFO] [FINAL RESULT]
{'_id': '1', 'likes': 1}

[INFO] [SECONDARY] Starting 1 client with w=3 write concern and 'majority' read concern, but timeout = 3000...
[INFO] [START RESULT] Right before stoping SECONDARY...
{'_id': '1', 'likes': 1}
docker stop mongo3
mongo3
[INFO] [MID RESULT] Right after stoping SECONDARY...
{'_id': '1', 'likes': 1}
[TIMEOUT] client_work WTimeoutError
[INFO] [MID RESULT] After 10 seconds stoping SECONDARY...
{'_id': '1', 'likes': 2}
docker start mongo3
mongo3
[INFO] [FINAL RESULT]
{'_id': '1', 'likes': 2}

```

4. Продемонстрував перевибори primary node відключивши поточний primary

```

replicaset [direct: primary] test> rs.status()
{
  set: 'replicaset',
  date: ISODate('2025-12-14T19:13:48.632Z'),
  myState: 1,
  term: Long('4'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,

```

```

optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
    lastCommittedWallTime: ISODate('2025-12-14T19:13:39.283Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
    appliedOpTime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
    durableOpTime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
    writtenOpTime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
    lastAppliedWallTime: ISODate('2025-12-14T19:13:39.283Z'),
    lastDurableWallTime: ISODate('2025-12-14T19:13:39.283Z'),
    lastWrittenWallTime: ISODate('2025-12-14T19:13:39.283Z')
},
lastStableRecoveryTimestamp: Timestamp({ t: 1765739560, i: 1 }),
electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-12-14T19:13:19.273Z'),
    electionTerm: Long('4'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    lastSeenWrittenOpTimeAtElection: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    priorPrimaryMemberId: 0,
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2025-12-14T19:13:29.282Z'),
    wMajorityWriteAvailabilityDate: ISODate('2025-12-14T19:13:29.383Z')
},
electionParticipantMetrics: {
    votedForCandidate: false,
    electionTerm: Long('4'),
    lastVoteDate: ISODate('2025-12-14T19:13:19.277Z'),
    electionCandidateMemberId: 1,
    voteReason: 'already voted for another candidate (mongo2:27017) this term (4)',
    lastWrittenOpTimeAtElection: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    maxWrittenOpTimeInSet: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1765739588, i: 2 }), t: Long('3') },
    priorityAtElection: 1
},
members: [
    {
        _id: 0,
        name: 'mongo1:27017',
        health: 0,
        state: 8,
        stateStr: '(not reachable/healthy)',
        uptime: 0,
        optime: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
        optimeDurable: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
        optimeWritten: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
        optimeDate: ISODate('1970-01-01T00:00:00.000Z'),
        optimeDurableDate: ISODate('1970-01-01T00:00:00.000Z'),
        optimeWrittenDate: ISODate('1970-01-01T00:00:00.000Z'),
        lastAppliedWallTime: ISODate('2025-12-14T19:13:08.755Z'),
        lastDurableWallTime: ISODate('2025-12-14T19:13:08.755Z'),
        lastWrittenWallTime: ISODate('2025-12-14T19:13:08.755Z'),
        lastHeartbeat: ISODate('2025-12-14T19:13:44.290Z'),
        lastHeartbeatRecv: ISODate('2025-12-14T19:13:08.736Z'),
        pingMs: Long('0'),
        lastHeartbeatMessage: 'no response within election timeout period',
        syncSourceHost: '',
        syncSourceId: -1,
        infoMessage: '',
        configVersion: 1,
        configTerm: 3
    },
    {
        _id: 1,
        name: 'mongo2:27017',
        health: 1,
        state: 1,
        stateStr: 'PRIMARY',
        uptime: 197,
        optime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
        optimeDate: ISODate('2025-12-14T19:13:39.000Z'),
        optimeWritten: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
        optimeWrittenDate: ISODate('2025-12-14T19:13:39.000Z'),
        lastAppliedWallTime: ISODate('2025-12-14T19:13:39.283Z'),
        lastDurableWallTime: ISODate('2025-12-14T19:13:39.283Z'),
        lastWrittenWallTime: ISODate('2025-12-14T19:13:39.283Z'),
        syncSourceHost: ''
    }
]

```

```

        syncSourceId: -1,
        infoMessage: 'Could not find member to sync from',
        electionTime: Timestamp({ t: 1765739599, i: 1 }),
        electionDate: ISODate('2025-12-14T19:13:19.000Z'),
        configVersion: 1,
        configTerm: 4,
        self: true,
        lastHeartbeatMessage: ''
    },
    {
        _id: 2,
        name: 'mongo3:27017',
        health: 1,
        state: 2,
        stateStr: 'SECONDARY',
        uptime: 189,
        optime: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
        optimeDurable: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
        optimeWritten: { ts: Timestamp({ t: 1765739619, i: 1 }), t: Long('4') },
        optimeDate: ISODate('2025-12-14T19:13:39.000Z'),
        optimeDurableDate: ISODate('2025-12-14T19:13:39.000Z'),
        optimeWrittenDate: ISODate('2025-12-14T19:13:39.000Z'),
        lastAppliedWallTime: ISODate('2025-12-14T19:13:39.283Z'),
        lastDurableViewTime: ISODate('2025-12-14T19:13:39.283Z'),
        lastWrittenWallTime: ISODate('2025-12-14T19:13:39.283Z'),
        lastHeartbeat: ISODate('2025-12-14T19:13:47.285Z'),
        lastHeartbeatRecv: ISODate('2025-12-14T19:13:47.786Z'),
        pingMs: Long('0'),
        lastHeartbeatMessage: '',
        syncSourceHost: 'mongo2:27017',
        syncSourceId: 1,
        infoMessage: '',
        configVersion: 1,
        configTerm: 4
    }
],
ok: 1,
'$clusterTime': {
    clusterTime: Timestamp({ t: 1765739619, i: 1 }),
    signature: {
        hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAA='),
        keyId: Long('0')
    }
},
operationTime: Timestamp({ t: 1765739619, i: 1 })
}

```

5. і що після відновлення роботи старої primary на неї реплікуються нові дані, які з'явилися під час її простою

```
[VERIFY] Checking local data directly on mongo1...
{'_id': '1', 'likes': 2}
```

## II Аналіз продуктивності та перевірка цілісності

Створити колекцію з каунтером лайків. Далі з 10 окремих клієнтів одночасно запустити інкрементацію каунтеру лайків по 10000 на кожного клієнта з різними опціями взаємодії з MongoDB.

- Вказавши у параметрах findOneAndUpdate writeConcern = 1

```
[REPLICA_SET] Initiating full cluster shutdown on mongo1...
```

- Вказавши у параметрах findOneAndUpdate writeConcern = majority

```
[REPLICA_SET] Initiating full cluster shutdown on mongo1...
```

3. Повторно запустить код при writeConcern = 1, але тепер під час роботи відключіть Primary ноду і подивитись що буде обрана інша Primary нода, яка продовжить обробку запитів, і чи кінцевий результат буде коректним

Відповідь:

4. Повторно запустить код при writeConcern = majority, але тепер під час роботи відключіть Primary ноду і подивитись що буде обрана інша Primary нода, яка продовжить обробку запитів, і чи кінцевий результат буде коректним

Відповідь: