Проектування високонавантажених систем

Лабораторна робота 4

Торгало Ігнатій ФБ-51мп

1. Створення docker-compose.yml та keyfile

```yaml
version: '3.8'

services:
  mongo1:
    image: mongo:6.0
    hostname: mongo1
    container_name: mongo1
    command: ["--replSet", "rs0", "--bind_ip_all", "--keyFile", "/etc/mongo-keyfile/keyfile"]
    ports:
      - "27017:27017"
    volumes:
      - ./data1:/data/db
      - ./keyfile:/etc/mongo-keyfile
    networks:
      - mongo-net

  mongo2:
    image: mongo:6.0
    hostname: mongo2
    container_name: mongo2
    command: ["--replSet", "rs0", "--bind_ip_all", "--keyFile", "/etc/mongo-keyfile/keyfile"]
    ports:
      - "27018:27017"
    volumes:
      - ./data2:/data/db
      - ./keyfile:/etc/mongo-keyfile
    networks:
      - mongo-net

  mongo3:
    image: mongo:6.0
    hostname: mongo3
    container_name: mongo3
    command: ["--replSet", "rs0", "--bind_ip_all", "--keyFile", "/etc/mongo-keyfile/keyfile"]
    ports:
      - "27019:27017"
    volumes:
      - ./data3:/data/db
      - ./keyfile:/etc/mongo-keyfile
    networks:
      - mongo-net

networks:
  mongo-net:
```

```
~/Desktop/KPI/ПВС/lab4
openssl rand -base64 756 > keyfile/keyfile

~/Desktop/KPI/ПВС/lab4
chmod 600 keyfile/keyfile
```

2. Ініціалізація Replica Set

```
~/Desktop/KPI/ПВС/lab4                                                    07:44:50
sudo docker exec -it mongo1 mongosh
Current Mongosh Log ID: 694394d7be9e16352ece5f46
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:          6.0.26
Using Mongosh:          2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

test> rs.initiate({
...   _id: "rs0",
...   members: [
...     { _id: 0, host: "mongo1:27017" },
...     { _id: 1, host: "mongo2:27017" },
...     { _id: 2, host: "mongo3:27017" }
...   ]
... })
...
{ ok: 1 }
```

```
members: [
  {
    _id: 0,
    name: 'mongo1:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 108,
    optime: { ts: Timestamp({ t: 1766036773, i: 1 }), t: Long('1') },
    optimeDate: ISODate('2025-12-18T05:46:13.000Z'),
    lastAppliedWallTime: ISODate('2025-12-18T05:46:13.356Z'),
    lastDurableWallTime: ISODate('2025-12-18T05:46:13.356Z'),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1766036713, i: 1 }),
    electionDate: ISODate('2025-12-18T05:45:13.000Z'),
    configVersion: 1,
    configTerm: 1,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: 'mongo2:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 70,
    optime: { ts: Timestamp({ t: 1766036763, i: 1 }), t: Long('1') },
    optimeDurable: { ts: Timestamp({ t: 1766036763, i: 1 }), t: Long('1') },
    optimeDate: ISODate('2025-12-18T05:46:03.000Z'),
    optimeDurableDate: ISODate('2025-12-18T05:46:03.000Z'),
    lastAppliedWallTime: ISODate('2025-12-18T05:46:13.356Z'),
    lastDurableWallTime: ISODate('2025-12-18T05:46:13.356Z'),
    lastHeartbeat: ISODate('2025-12-18T05:46:13.361Z'),
    lastHeartbeatRecv: ISODate('2025-12-18T05:46:12.867Z'),
    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: 70,
    optime: { ts: Timestamp({ t: 1766036763, i: 1 }), t: Long('1') },
    optimeDurable: { ts: Timestamp({ t: 1766036763, i: 1 }), t: Long('1') },
    optimeDate: ISODate('2025-12-18T05:46:03.000Z'),
    optimeDurableDate: ISODate('2025-12-18T05:46:03.000Z'),
```

3. Перевірка writeConcern = 3 при вимкненій ноді

```
sudo docker stop mongo2

mongo2
```

```
rs0 [direct: primary] admin> db.test.insertOne({x:1}, {writeConcern:{w:3, wtimeout:0}})
...
```

Операція зависає, тепер запустимо назад і операція одразу завершилась успішно

```
sudo docker start mongo2

mongo2
```

```
rs0 [direct: primary] admin> db.test.insertOne({x:1}, {writeConcern:{w:3, wtimeout:0}})
...
{
  acknowledged: true,
  insertedId: ObjectId('694398810308b85eb9ce5f47')
}
```

4. Перевірка writeConcern = 3 з таймаутом та недоступної ноди

Вимикаємо ноду

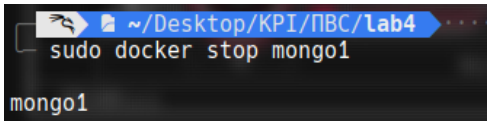
```
sudo docker stop mongo2

mongo2
```

```
rs0 [direct: primary] admin> db.test.insertOne({x:2}, {writeConcern:{w:3, wtimeout:2000}})
Uncaught:
MongoWriteConcernError[WriteConcernFailed]: waiting for replication timed out
Additional information: {
  wtimeout: true,
  writeConcern: { w: 3, wtimeout: 2000, provenance: 'clientSupplied' }
}
Result: {
  n: 1,
  electionId: ObjectId('7fffffff0000000000000002'),
  opTime: { ts: Timestamp({ t: 1766037928, i: 1 }), t: 2 },
  writeConcernError: {
    code: 64,
    codeName: 'WriteConcernFailed',
    errmsg: 'waiting for replication timed out',
    errInfo: {
      wtimeout: true,
      writeConcern: { w: 3, wtimeout: 2000, provenance: 'clientSupplied' }
    }
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1766037928, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('ASyekj5hgPafppXwZMm5eA0jIqY=', 0),
      keyId: Long('7585069925870338054')
    }
  },
  operationTime: Timestamp({ t: 1766037928, i: 1 })
}
rs0 [direct: primary] admin> db.test.find().readConcern("majority")
[
  { _id: ObjectId('694398810308b85eb9ce5f47'), x: 1 },
  { _id: ObjectId('6943994a0308b85eb9ce5f48'), x: 2 },
  { _id: ObjectId('694399a10308b85eb9ce5f49'), x: 2 },
  { _id: ObjectId('694399a80308b85eb9ce5f4a'), x: 2 }
]
```

Оскільки рівень writeConcern=3 вимагає підтвердження запису *усіма трьома нодами Replica Set*, а одна з нод була недоступною або не встигла підтвердити реплікацію, система повернула: WAITING FOR REPLICATION TIMED OUT. При цьому запис фактично був успішно виконаний на Primary (ok:1), але не був підтверджений Secondary у встановлений таймаут.

5. Перевибори Primary (Replica Set Elections)

Вимикаємо Primary node:



Нова Primary з’явилась автоматично - mongo2

6. Тестування продуктивності + Counter

Створюємо колекцію

```
rs0 [direct: primary] admin> db.counter.insertOne({ _id: 1, value: 0 })
...
{ acknowledged: true, insertedId: 1 }
```

```
~/Desktop/KPI/ПВС/lab4
python3 test.py

--- writeConcern=1 ---
Time: 60.968472957611084
Final value: 100000
--------------------------------

--- writeConcern=majority ---
Time: 128.0373272895813
Final value: 100000
--------------------------------
```

test.py

```python
from pymongo import MongoClient
import threading, time

CONN_STR = "mongodb://root:password@localhost:27017,localhost:27018,localhost:27019/?replicaSet=rs0&authSource=admin

client = MongoClient(CONN_STR)
db = client["lab4"]
col = db["counter"]

N_THREADS = 10
ITER = 10000

def worker(write_concern):
    local_client = MongoClient(CONN_STR, w=write_concern)
    c = local_client["lab4"]["counter"]
    for _ in range(ITER):
        c.find_one_and_update(
            {"_id": 1},
            {"$inc": {"value": 1}}
        )

def reset_counter():
    col.update_one({"_id": 1}, {"$set": {"value": 0}}, upsert=True)

def run_test(wc):
    reset_counter()

    threads = []
    start = time.time()

    for _ in range(N_THREADS):
        t = threading.Thread(target=worker, args=(wc,))
        t.start()
        threads.append(t)

    for t in threads:
        t.join()

    end = time.time()

    print(f"\n—— writeConcern={wc} ——")
    print("Time:", end - start)
    print("Final value:", col.find_one({"_id": 1})["value"])
    print("————————————————————————")

run_test(1)
run_test("majority")
```

7. Висновок

У ході лабораторної роботи було розгорнуто кластер MongoDB з трьох реплік за допомогою Docker та налаштовано ReplicaSet *rs0*. Проведено тестування впливу параметра **writeConcern** на швидкість і надійність запису в розподілену систему.

Було виконано два експерименти:

- при **writeConcern = 1** (підтвердження запису тільки від PRIMARY);
- при **writeConcern = majority** (підтвердження запису більшістю реплік).

Результати:

- writeConcern=1: час виконання ≈ **61 сек**, фінальне значення лічильника — **100000**
- writeConcern=majority: час виконання ≈ **128 сек**, фінальне значення — **100000**

Отже, збільшення writeConcern забезпечує більш високу гарантію збереження даних (надійність у випадку відмов окремих вузлів), але суттєво впливає на продуктивність, збільшуючи час обробки операцій запису. Кластер відпрацював коректно, і всі інкрементальні операції були виконані без втрат.

Таким чином, лабораторна робота продемонструвала практичний вплив writeConcern на консистентність та швидкодію MongoDB у реплікованому середовищі.