Проєктування високонавантаженних систем. Лабораторна 2, звіт

Михайло Голуб

19 листопада 2025 р.

1 Завдання лабораторної роботи

1.1 Загальне

1. Встановити і налаштувати Hazelcast 5.4.x (у новіших версіях частина необхідного для виконання завдань функціоналу є платною)
2. Сконфігурувати і запустити 3 ноди (інстанси) об'єднані в кластер або як частину Java-застосування, або як окремі застосування У справжній системі кожна нода має запускатись на окремому сервері.
3. Далі, на основі прикладу з Distributed Map, напишіть код який буде емулювати інкремент значення для одного й того самого ключа у циклі до 10K. Це необхідно робити у 10 потоках.
4. Виходячи з того, що 10 потоків інкрементують каунтер 10K разів кожен, то остаточне значення каунтера має бути 10*10_000 = 100_000. Для імплементації спочатку скористаємось Distributed Map у Hazelcast
5. Реалізуйте каунтер без блокувань. Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.
6. Реалізуйте каунтер з використанням песимістичного блокування. Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.
7. реалізуйте каунтер з використанням оптимістичного блокування. Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.
8. Реалізуйте каунтер з використанням IAtomicLong та увімкнувши підтимку CP Sysbsystem на основі трьох нод. УВАГА! Без CP Sysbsystem не гарантується коректність результату (у протоколі мають бути логи Hazelcast з яких видно, що CP Subsystem активована та складається з 3-х нод) Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.

1.2 Вимоги до звіту та реалізації

- Мова реалізації будь-яка
- Має бути надано код програми/скрипта та результати виконання
- Приведно лог, який видають ноди Hazelcast, де буде видно що кластер складається з 3-х нод і що активована CP Subsystem

2 Запуск Hazelcast

Hazelcast локально захощено через Docker. Створюються три контейнери контейнери Hazelcast 5.4.0: `hz-node1`, `hz-node2`, `hz-node3` на портах 5701, 5702 та 5703 відповідно. Та контейнер Hazelcast management-center 5.4.0: `hz-mz` на порті 8080

Лістинг 1: docker-compose.yml

```
version: '3.8'

services:
  hazelcast1:
    image: hazelcast/hazelcast:5.4.0
    container_name: hz-node1
    ports:
      - "5701:5701"
    environment:
      - HZ_NETWORK_JOIN_MULTICAST_ENABLED=false
      - HZ_NETWORK_JOIN_TCPIP_ENABLED=true
      - HZ_NETWORK_JOIN_TCPIP_MEMBERS=hazelcast1,
   hazelcast2,hazelcast3
      - HZ_CP_MEMBER_COUNT=3
      - JAVA_OPTS=-Dhazelcast.local.publicAddress=
   hazelcast1:5701
    networks:
      - hz-network

  hazelcast2:
    image: hazelcast/hazelcast:5.4.0
    container_name: hz-node2
    ports:
      - "5702:5701"
    environment:
      - HZ_NETWORK_JOIN_MULTICAST_ENABLED=false
      - HZ_NETWORK_JOIN_TCPIP_ENABLED=true
      - HZ_NETWORK_JOIN_TCPIP_MEMBERS=hazelcast1,
   hazelcast2,hazelcast3
```

```
      - HZ_CP_MEMBER_COUNT=3
      - JAVA_OPTS=-Dhazelcast.local.publicAddress=hazelcast2:5701
    depends_on:
      - hazelcast1
    networks:
      - hz-network

  hazelcast3:
    image: hazelcast/hazelcast:5.4.0
    container_name: hz-node3
    ports:
      - "5703:5701"
    environment:
      - HZ_NETWORK_JOIN_MULTICAST_ENABLED=false
      - HZ_NETWORK_JOIN_TCPIP_ENABLED=true
      - HZ_NETWORK_JOIN_TCPIP_MEMBERS=hazelcast1,hazelcast2,hazelcast3
      - HZ_CP_MEMBER_COUNT=3
      - JAVA_OPTS=-Dhazelcast.local.publicAddress=hazelcast3:5701
    depends_on:
      - hazelcast1
    networks:
      - hz-network

  management-center:
    image: hazelcast/management-center:5.4.0
    container_name: hz-mc
    ports:
      - "8080:8080"
    environment:
      - MC_DEFAULT_CLUSTER=dev
      - MC_DEFAULT_CLUSTER_MEMBERS=hazelcast1,hazelcast2,hazelcast3
    depends_on:
      - hazelcast1
    networks:
      - hz-network

networks:
  hz-network:
    driver: bridge
```

Після запуску усі чотири контейнери запущені успішно і існує кластер з трьома нодами. Про це свідчить ця частина логу:

```
[ WARN] [main] [c.h.c.CPSubsystem]: [hazelcast3]:5701 [dev] [5.4.0] CP Subsystem is not enab
Members {size:3, ver:3} [
    Member [hazelcast1]:5701 - 360c484e-0413-4f01-b810-edd4f20fb514
    Member [hazelcast2]:5701 - 912b48a7-4b31-4303-9e30-06560e463ef1
    Member [hazelcast3]:5701 - ae62cc6a-26c6-4188-b174-ce60c234888c this
]
```

3 Клієнт

Майже ідентичний клієнту реалізованому в першій лабораторній роботі. Додано лог проходження певної частки запитів, для розуміння чи просувається виконання запитів.

Лістинг 2: client.py

```python
import requests
import threading
import time
import random
from tqdm import tqdm

SERVER = "http://127.0.0.1:8080"
CALLS_PER_CLIENT = 10
NUM_CLIENTS = 10

def worker(client_id):
    for i in tqdm(range(CALLS_PER_CLIENT), desc = str(client_id)):
        trying = True
        while trying:
            try:
                requests.get(f"{SERVER}/inc")
                trying = False
            except requests.exceptions.RequestException:
                print(client_id, "worker, request denied, N =",j*
    CALLS_PER_CLIENT//N_REPORTS + i)
                time.sleep(random.random())

def main():
    print(f"Starting {NUM_CLIENTS} clients x {CALLS_PER_CLIENT}
    calls each...")
    start = time.time()

    threads = []
    for i in range(NUM_CLIENTS):
        t = threading.Thread(target=worker, args=(i,))
        t.start()
        threads.append(t)
    print("MAIN: Threads created")
    for t in threads:
        t.join()
    print("MAIN: Threads joined")
    end = time.time()
    elapsed = end - start
    final_count = -1
```

```
    i = 0
    while final_count == -1 and i < 5:
        try:
            final_count = int(requests.get(f"{SERVER}/count").text)
            print(final_count)
        except:
            final_count = -1
            i+=1
            time.sleep(1)

    total_calls = CALLS_PER_CLIENT * NUM_CLIENTS
    throughput = total_calls / elapsed

    print(f"Final count: {final_count}")
    print(f"Total time:  {elapsed:.2f} s")
    print(f"Throughput:  {throughput:.2f} requests/sec")

if __name__ == "__main__":
    main()
    input()
```

Приклад результату роботи клієнта:

```
Starting 10 clients x 10000 calls each...
0 worker started
1 worker started
2 worker started
3 worker started
4 worker started
5 worker started
6 worker started
7 worker started
8 worker started
9 worker started
MAIN: Threads created
1 worker reporting 0 / 10
9 worker reporting 0 / 10
```

І через декілька десятків рядків, завершальний блок:

```
6 worker reporting 9 / 10
6 worker end
MAIN: Threads joined
100000
Final count: 100000
Total time:  511.19 s
Throughput:  195.62 requests/sec
```

4 Лічильник без блокування

Застосунок отримує значення лічильника, інкрементує його всередині потоку застосунку і записує назад в Hazelcast. Таким чином, блокування від-

сутні.

Лістинг 3: app_no_block.py

```python
from flask import Flask
from hazelcast.client import HazelcastClient

print("Starting with no block")

app = Flask(__name__)

HZ_ADDRESSES = [
    "127.0.0.1:5701",
    "127.0.0.1:5702",
    "127.0.0.1:5703"
]

COUNTER_NAME = "likes-counter"

print("Connecting to Hazelcast...")
client = HazelcastClient(
    cluster_members=HZ_ADDRESSES,
    cluster_name="dev",
)

cp = client.cp_subsystem
counter = cp.get_atomic_long(COUNTER_NAME).blocking()

counter.set(0)
print(f"Hazelcast connected. Counter '{COUNTER_NAME}' reset to 0.")


@app.route("/inc")
def inc():
    new_value = counter.get() + 1
    counter.set(new_value)
    return str(new_value)


@app.route("/count")
def count():
    current = counter.get()
    return str(current)


if __name__ == "__main__":
    try:
        app.run(host="0.0.0.0", port=5000, debug=False)
    finally:
        client.shutdown()
```

Результати тестування:
Час роботи: 3217 секунд
Пропускна здатність: 31.08 запитів/с
Кінцеве значення лічильника: 32582

5 Лічильник з песимістичним блокуванням

Застосунок блокує блок (або очікує доки може заблокувати), отримує значення лічильника, інкрементує його всередині потоку застосунку, записує назад в Hazelcast та відпускає блок.

Лістинг 4: app_pessimistic_block.py

```python
from flask import Flask
from hazelcast.client import HazelcastClient

print("Starting with pessimistic block")

app = Flask(__name__)

HZ_ADDRESSES = [
    "127.0.0.1:5701",
    "127.0.0.1:5702",
    "127.0.0.1:5703"
]

client = HazelcastClient(cluster_members=HZ_ADDRESSES)
counter_map = client.get_map("likes-map").blocking()

COUNTER_KEY = "likes"

counter_map.put(COUNTER_KEY, 0)
print("Hazelcast connected. Counter reset.")


@app.route("/inc")
def inc():
    counter_map.lock(COUNTER_KEY)
    value = counter_map.get(COUNTER_KEY)
    new_value = value + 1
    counter_map.put(COUNTER_KEY, new_value)
    counter_map.unlock(COUNTER_KEY)
    return str(new_value)



@app.route("/count")
def count():
    value = counter_map.get(COUNTER_KEY)
    return str(value)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=False)
```

Задля економії часу, зменшено кількість запитів клієнтів з 10000 до 10.
Результати тестування:
Час роботи: 1290 секунд
Пропускна здатність: 0.08 запитів/с
Кінцеве значення лічильника: 100