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

Проектування високонавантажених систем
Лабораторна робота No2
MassiveCounterIncrement

Виконав:
Студент 4-го курсу
групи ФІ-21
Климентьєв Максим
Перевірив:
_____

Зміст

1 Код реалізації

BaseCounter.py

```
from abc import ABC, abstractmethod
import hazelcast


class BaseCounter(ABC):
    def __init__(self, client: hazelcast.HazelcastClient, name: str,
   key: str = "counter"):
        self.client = client
        self.name = name
        self.key = key

    @abstractmethod
    def increment(self):
        pass

    def bulk_increment(self, n: int):
        for _ in range(n):
            self.increment()
```

AtomicLongCounter.py

```
from Counters.BaseCounter import BaseCounter


class AtomicLongCounter(BaseCounter):
    def __init__(self, client, name="atomic_counter", key="counter"):
        super().__init__(client, name, key)
        self.atomic = client.cp_subsystem.get_atomic_long(name).
   blocking()
        # if self.atomic.get() is None: self.atomic.set(0)

    def increment(self):
        self.atomic.add_and_get(1)
```

MapNoLockCounter.py

```
from Counters.BaseCounter import BaseCounter


class MapNoLockCounter(BaseCounter):
    def __init__(self, client, name="map_no_lock", key="counter"):
        super().__init__(client, name, key)
        self.map = client.get_map(name).blocking()
        if self.map.get(key) is None:
            self.map.put(key, 0)
```

```
    def increment(self):
        v = self.map.get(self.key)
        # симуляція невеликої операційної затримки
        v = 0 if v is None else v
        new = v + 1
        self.map.put(self.key, new)
```

MapOptimisticCounter.py

```
from Counters.BaseCounter import BaseCounter

class MapOptimisticCounter(BaseCounter):
    def __init__(self, client, name="map_optimistic", key="counter",
   max_retries=100):
        super().__init__(client, name, key)
        self.map = client.get_map(name).blocking()
        if self.map.get(key) is None:
            self.map.put(key, 0)
        self.max_retries = max_retries

    def increment(self):
        for _ in range(self.max_retries):
            old = self.map.get(self.key) or 0
            new = old + 1
            # replace returns True if successful (atomic compare-and-
   replace)
            ok = self.map.replace_if_same(self.key, old, new)
            if ok:
                return
        # якщо після max_retries не вдалося — як fallback зробимо
   блокування
        # щоб( не втрачати значення)
        # Простий fallback:
        self.map.lock(self.key)
        try:
            v = self.map.get(self.key) or 0
            self.map.put(self.key, v + 1)
        finally:
            self.map.unlock(self.key)
```

MapPessimisticCounter.py

```
from Counters.BaseCounter import BaseCounter

class MapPessimisticCounter(BaseCounter):
    def __init__(self, client, name="map_pessimistic", key="counter"):
        super().__init__(client, name, key)
        self.map = client.get_map(name).blocking()
        if self.map.get(key) is None:
```

```
            self.map.put(key, 0)

    def increment(self):
        # блокування ключа на час операції
        self.map.lock(self.key)
        try:
            v = self.map.get(self.key) or 0
            self.map.put(self.key, v + 1)
        finally:
            self.map.unlock(self.key)
```

HazelcastCounters.py

```
import threading
import time

import hazelcast

from Counters import AtomicLongCounter, MapNoLockCounter,
   MapOptimisticCounter, MapPessimisticCounter

def run_benchmark(counter_cls, client, threads=10,
   increments_per_thread=10_000):
    counter = counter_cls(client)
    # reset value to 0 where possible
    if isinstance(counter, AtomicLongCounter):
        counter.atomic.set(0)
    else:
        m = client.get_map(counter.name).blocking()
        m.put(counter.key, 0)

    threads_list = []
    start = time.perf_counter()
    for t in range(threads):
        th = threading.Thread(target=counter.bulk_increment, args=(
   increments_per_thread,))
        th.start()
        threads_list.append(th)
    for th in threads_list:
        th.join()
    elapsed = time.perf_counter() - start

    if isinstance(counter, AtomicLongCounter):
        final = counter.atomic.get()
    else:
        final = client.get_map(counter.name).blocking().get(counter.key
   )
    return final, elapsed
```

```
if __name__ == "__main__":
    print("[PRE] Connecting to Hazelcast...")
    client = hazelcast.HazelcastClient(
        cluster_name="dev",
        cluster_members=["hazelcast1:5701","hazelcast2:5702","
   hazelcast3:5703"]
    )


    print("[MAIN] Starting different Counters...")
    variants = {
        "Map no-lock": MapNoLockCounter,
        "Map pessimistic": MapPessimisticCounter,
        "Map optimistic": MapOptimisticCounter,
        "AtomicLong (CP)": AtomicLongCounter,
    }
    threads = 10
    increments_per_thread = 10_000
    try:
        for name, cls in variants.items():
            print("[MAIN] Running:", name)
            val, took = run_benchmark(cls, client, threads=threads,
   increments_per_thread=increments_per_thread)
            print(f"[Results] Received: {val}\n[Results] Expected: {
   increments_per_thread * threads}\n[Results] Time: {took:.2f} sec\n",
    "-"*60)
    finally:
        client.shutdown()
```

Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY req.txt .
RUN pip install --no-cache-dir -r req.txt

COPY ./Counters ./Counters
COPY HazelcastCounters.py .

CMD ["python", "HazelcastCounters.py"]
```

docker-compose.yml

```
services:
  hazelcast1:
    image: hazelcast/hazelcast:5.4
    container_name: hz1
```

```
    ports:
      - "5701:5701"
    volumes:
      - ./hazelcast.yaml:/opt/hazelcast/hazelcast.yaml
    environment:
      - HZ_CLUSTERNAME=dev

  hazelcast2:
    image: hazelcast/hazelcast:5.4
    container_name: hz2
    ports:
      - "5702:5701"
    volumes:
      - ./hazelcast.yaml:/opt/hazelcast/hazelcast.yaml
    environment:
      - HZ_CLUSTERNAME=dev

  hazelcast3:
    image: hazelcast/hazelcast:5.4
    container_name: hz3
    ports:
      - "5703:5701"
    volumes:
      - ./hazelcast.yaml:/opt/hazelcast/hazelcast.yaml
    environment:
      - HZ_CLUSTERNAME=dev

  python-client:
    build: .
    container_name: hz-client
    depends_on:
      - hazelcast1
      - hazelcast2
      - hazelcast3
    environment:
      - PYTHONUNBUFFERED=1
```

hazelcast.yaml

```
hazelcast:
  cluster-name: dev

  network:
    join:
      multicast:
        enabled: false
      tcp-ip:
        enabled: true
        member-list:
```

```
      - hazelcast1
      - hazelcast2
      - hazelcast3
```

2 Результати

Результат:

[PRE] Connecting to Hazelcast...
[MAIN] Starting different Counters...
[MAIN] Running: Map no-lock
[Results] Received: 15414
[Results] Expected: 100000
[Results] Time: 22.10 sec

[MAIN] Running: Map pessimistic
[Results] Received: 100000
[Results] Expected: 100000
[Results] Time: 123.39 sec

[MAIN] Running: Map optimistic
[Results] Received: 100000
[Results] Expected: 100000
[Results] Time: 117.23 sec

[MAIN] Running: AtomicLong (CP)
[Results] Received: 100000
[Results] Expected: 100000
[Results] Time: 11.59 sec

---

3 Лог, який видають ноди Hazelcast