

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

**Проектування високонавантажених систем**

Лабораторна робота №2  
MassiveCounterIncrement

**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>9</b>
<b>3</b>	<b>Лог, який видають ноди Hazelcast</b>	<b>11</b>

# 1 Код реалізації

## BaseCounter.py

```
from abc import ABC, abstractmethod
import hazelcast
# import time
# from tqdm import tqdm

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): # , log_every: int = 1

        # for _ in tqdm(range(n), desc=f"{self.name}: "):
        for iteration in range(n):
            # t0 = time.time()
            self.increment()
            # dt = time.time() - t0
            # avg = dt if avg is None else avg * 0.95 + dt * 0.05

            # if iteration % log_every == 0:
            #     eta = avg * (n - iteration)
            #     print(f"[{iteration}/{n}] [{self.name}] ETA: {eta:.2f}s",
flush=True)
```

## AtomicLongCounter.py

```
from Counters.BaseCounter import BaseCounter

class AtomicLongCounter(BaseCounter):
    def __init__(self, client, name="atomic_counter", key="counter"):
        super().__init__(client, name, key) # HazelcastInstance
hazelcastInstance = Hazelcast.newHazelcastInstance();
        self.atomic = client.cp_subsystem.get_atomic_long(name).
blocking() # IAtomicLong counter = hazelcastInstance.getCPSubsystem().
        .getAtomicLong("counter");

    def increment(self):
```

```
    self.atomic.add_and_get(1) # counter.incrementAndGet();
```

### MapNoLockCounter.py

```
# import time
from Counters.BaseCounter import BaseCounter


class MapNoLockCounter(BaseCounter):
    def __init__(self, client, name="map_no_lock", key="counter"):
        super().__init__(client, name, key) # HazelcastInstance hz =
Hazelcast.newHazelcastInstance();
        self.map = client.get_map(name).blocking() # IMap<String, Value
> map = hz.getMap( "map" );
        if self.map.get(key) is None: # String key = "1";
            self.map.put(key, 0) # map.put( key, new Value() );

    def increment(self):
        v = self.map.get(self.key) or 0 # Value value = map.get( key )
;
        # time.sleep(1e-2) # Thread.sleep( 10 );
        v += 1 # value.amount++;
        self.map.put(self.key, v) # map.put( key, value );
```

### MapOptimisticCounter.py

```
# import time
from Counters.BaseCounter import BaseCounter


class MapOptimisticCounter(BaseCounter):
    def __init__(self, client, name="map_optimistic", key="counter"):
        super().__init__(client, name, key) # HazelcastInstance hz =
Hazelcast.newHazelcastInstance();
        self.map = client.get_map(name).blocking() # IMap<String,
Value> map = hz.getMap( "map" );
        if self.map.get(key) is None: # String key = "1";
            self.map.put(key, 0) # map.put( key, new Value() );

    def increment(self):
        while True: # for ( ; ; ) {
            old = self.map.get(self.key) or 0 # Value oldValue = map.
get( key );
            new = old + 1 # Value newValue = new Value( oldValue );
newValue.amount++;
            # time.sleep(1e-2) # Thread.sleep( 10 );

            if self.map.replace_if_same(self.key, old, new): # if (
map.replace( key, oldValue, newValue ) )
                break # break;
```

## MapPessimisticCounter.py

```
# import time
from Counters.BaseCounter import BaseCounter

class MapPessimisticCounter(BaseCounter):
    def __init__(self, client, name="map_pessimistic", key="counter"):
        super().__init__(client, name, key) # HazelcastInstance hz =
Hazelcast.newHazelcastInstance();
        self.map = client.getMap(name).blocking() # IMap<String,
Value> map = hz.getMap( "map" );
        if self.map.get(key) is None: # String key = "1";
            self.map.put(key, 0) # map.put( key, new Value() );

    def increment(self):
        self.map.lock(self.key) # map.lock( key );
        try: # try {
            v = self.map.get(self.key) or 0 # Value value = map.get(
key );
            # time.sleep(1e-2) # Thread.sleep( 10 );
            v += 1 # value.amount++;
            self.map.put(self.key, v) # map.put( key, value );
        finally: # } finally {
            self.map.unlock(self.key) # map.unlock( key );
```

## HazelcastCounters.py

```
import threading
import time
# from tqdm import tqdm
# import tqdm

import hazelcast

from Counters import AtomicLongCounter, MapNoLockCounter,
MapOptimisticCounter, MapPessimisticCounter

def run_benchmark(counter_cls, client, threads: int = 10,
increments_per_thread: int = 10_000): # , log_every: int = 1000
    total = threads * increments_per_thread
    counter = counter_cls(client)
    if isinstance(counter, AtomicLongCounter):
        counter.atomic.set(0)
    else:
        m = client.getMap(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
throughput = total / elapsed

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

return final, total, elapsed, throughput

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
    # log_every = 100
    try:
        # for _ in range(5):
        for name, cls in variants.items():
            print("[MAIN] Running:", name)
            val, expected, elapsed, throughput = run_benchmark(cls,
client, threads=threads, increments_per_thread=increments_per_thread
) # , log_every=log_every
            print(f"[Results] Received: {val}\n[Results] Expected: {expected}\n[Results] Time: {elapsed:.2f} sec\n[Results] Throughput:

```

```

{throughput:.2f} requests/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
```

---

**Не розумію у чому проблема, але на наступний день і далі воно більше не показувало такі хороші результати, навіть коли відкатувався до версії того дня.**

```
[PRE] Connecting to Hazelcast...
[MAIN] Starting different Counters...
[MAIN] Running: Map no-lock
[Results] Received: 10262
[Results] Expected: 100000
[Results] Time: 114.94 sec
[Results] Throughput: 870.03 requests/sec
```

---

```
[MAIN] Running: Map pessimistic
[Results] Received: 100000
[Results] Expected: 100000
[Results] Time: 3186.91 sec
[Results] Throughput: 31.38 requests/sec
```

---

[MAIN] Running: Map optimistic  
[Results] Received: 10062  
[Results] Expected: 100000  
[Results] Time: 225.83 sec  
[Results] Throughput: 442.82 requests/sec

---

[MAIN] Running: AtomicLong (CP)  
[Results] Received: 100000  
[Results] Expected: 100000  
[Results] Time: 14.97 sec  
[Results] Throughput: 6681.59 requests/sec

---

### **3 Лог, який видають ноди Hazelcast**