МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Проектування розподілених систем

Лабораторна робота №2

Розгортання і робота з distributed in-memory data structures на основі Hazelcast: Distributed Map

Перевірив:
Родіонов А. М.

Виконав:
студент І курсу
групи ФБ-41мп
Сахній Н. Р.

Київ 2025

Мета роботи: Дослідити роботу розподіленої In-Memory бази даних Hazelcast на прикладі кластеру з 3-х нод, налаштувати розподілені структури даних (Distributed Map, Queue), оцінити їхню ефективність та відмовостійкість. Дослідити поведінку розподіленої мапи при втраті нод, а також перевірити роботу без блокувань, з песимістичним та оптимістичним блокуванням.
https://github.com/sazan24/KPI/tree/main/Master's%20degree/Distributed%20Systems%20Design/Task_2-HazelcastDistributedMap

Завдання до виконання:

1. Встановити і налаштувати Hazelcast.

- https://hazelcast.com/open-source-projects/downloads/

1) `docker network create -d bridge hazelcast-network`

```
nazar@ubuntu:~/KPI/DistSusDesign$ sudo docker network create -d bridge hazelcast-network
64608b4a0e473beb97e32b1d9293a564ef636e2278d729f7669c6e7f1c0c17a6
```

2) `pip install hazelcast-python-client==5.4.0`

```
nazar@ubuntu:~/KPI/DistSusDesign$ pip install hazelcast-python-client==5.4.0
Collecting hazelcast-python-client==5.4.0
  Downloading hazelcast_python_client-5.4.0-py3-none-any.whl (439 kB)
     |                                | 439 kB 330 kB/s
Installing collected packages: hazelcast-python-client
Successfully installed hazelcast-python-client-5.4.0
```

3) `nano ./docker-compose.yml`

```
version: "3.2"

services:
  # Hazelcast Node 1
  hazelcast-node1:
    container_name: 'hazelcast-node1'
    image: 'hazelcast/hazelcast:5.4.0'
    network_mode: 'hazelcast-network'
    environment:
      - HZ_CLUSTERNAME=distributed-map-cluster
      - HZ_NETWORK_PUBLICADDRESS=172.18.0.1:5701
    ports:
      - '5701:5701'

  # Hazelcast Node 2
  hazelcast-node2:
    container_name: 'hazelcast-node2'
    image: 'hazelcast/hazelcast:5.4.0'
    network_mode: 'hazelcast-network'
    environment:
      - HZ_CLUSTERNAME=distributed-map-cluster
      - HZ_NETWORK_PUBLICADDRESS=172.18.0.1:5702
    ports:
      - '5702:5701'

  # Hazelcast Node 3
  hazelcast-node3:
    container_name: 'hazelcast-node3'
    image: 'hazelcast/hazelcast:5.4.0'
    network_mode: 'hazelcast-network'
    environment:
      - HZ_CLUSTERNAME=distributed-map-cluster
      - HZ_NETWORK_PUBLICADDRESS=172.18.0.1:5703
    ports:
      - '5703:5701'

```

```
  # Management Center
  hazelcast-management:
    container_name: 'distmap-management-center'
    image: 'hazelcast/management-center:5.4.0'
    network_mode: 'hazelcast-network'
    depends_on:
      - hazelcast-node1
      - hazelcast-node2
      - hazelcast-node3
    ports:
      - '8080:8080'


# External Network
networks:
  hazelcast-network:
    driver: bridge

```

4) `docker-compose up`

```
hazelcast-node1 | 2025-04-03 09:25:02,071 [ INFO] [main] [c.h.s.logo]: [172.18.0.1]:5701 [distributed-map-cluster] [5.4.0]
hazelcast-node1 |         o    o      o     o---o   o--o o      o---o     o     o----o o--o--o
hazelcast-node1 |         |    |     / \       /          |     /         / \    |
hazelcast-node1 |         o----o        o     o   o----o   |    o          o   o----o    |
hazelcast-node1 |         |    |  *     \   /              |     \         *     \       |
hazelcast-node1 |         o    o *       o o---o   o--o o----o o---o *       o o----o    o

hazelcast-node2 | 2025-04-03 09:25:02,312 [ INFO] [main] [c.h.s.logo]: [172.18.0.1]:5702 [distributed-map-cluster] [5.4.0]
hazelcast-node2 |         o    o      o     o---o   o--o o      o---o     o     o----o o--o--o
hazelcast-node2 |         |    |     / \       /          |     /         / \    |
hazelcast-node2 |         o----o        o     o   o----o   |    o          o   o----o    |
hazelcast-node2 |         |    |  *     \   /              |     \         *     \       |
hazelcast-node2 |         o    o *       o o---o   o--o o----o o---o *       o o----o    o

hazelcast-node3 | 2025-04-03 09:25:02,106 [ INFO] [main] [c.h.s.logo]: [172.18.0.1]:5703 [distributed-map-cluster] [5.4.0]
hazelcast-node3 |         o    o      o     o---o   o--o o      o---o     o     o----o o--o--o
hazelcast-node3 |         |    |     / \       /          |     /         / \    |
hazelcast-node3 |         o----o        o     o   o----o   |    o          o   o----o    |
hazelcast-node3 |         |    |  *     \   /              |     \         *     \       |
hazelcast-node3 |         o    o *       o o---o   o--o o----o o---o *       o o----o    o
```

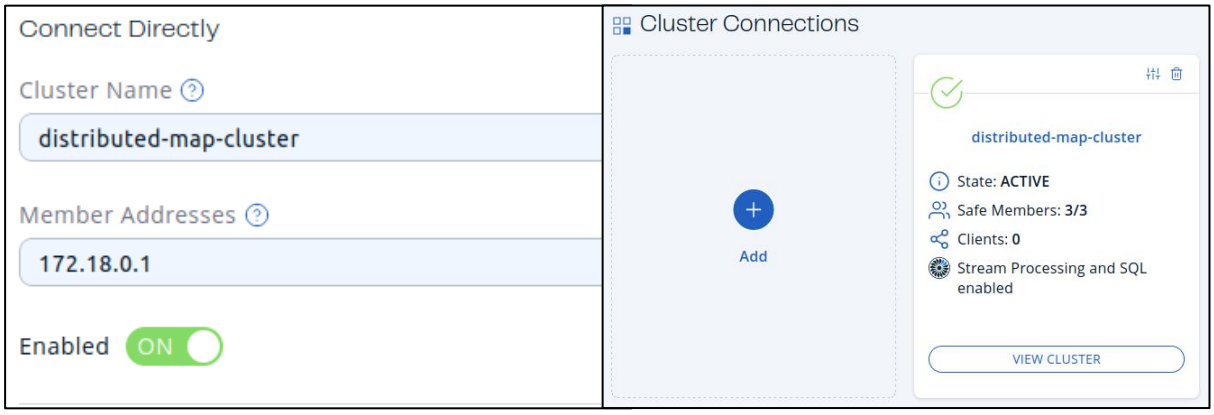
2. Сконфігурувати і запустити 3 ноди (інстанси) об'єднані в кластер або як частину Java-застосування, або як окремі застосування.

- https://docs.hazelcast.com/hazelcast/5.3/getting-started/get-started-binary#step-6-scale-your-cluster

1) `docker ps`

```
nazar@ubuntu:~/KPI/DistSusDesign$ sudo docker ps
CONTAINER ID   IMAGE                                 COMMAND                  CREATED         STATUS         PORTS                                                                                  NAMES
da06a192c19a   hazelcast/management-center:5.4.0     "bash ./bin/mc-start…"   3 minutes ago   Up 3 minutes   8081/tcp, 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp, 8443/tcp                        distmap-management-center
fb1bd5291566   hazelcast/hazelcast:5.4.0             "hz start"               4 minutes ago   Up 4 minutes   0.0.0.0:5703->5701/tcp, [::]:5703->5701/tcp                                            hazelcast-node3
b8249eaa2bb8   hazelcast/hazelcast:5.4.0             "hz start"               4 minutes ago   Up 4 minutes   0.0.0.0:5702->5701/tcp, [::]:5702->5701/tcp                                            hazelcast-node2
2c1ca009b360   hazelcast/hazelcast:5.4.0             "hz start"               4 minutes ago   Up 4 minutes   0.0.0.0:5701->5701/tcp, [::]:5701->5701/tcp                                            hazelcast-node1
```

2) Open URL-link: http://localhost:8080/cluster-connections



3. Продемонструйте працездатність структури Distributed Map.

- https://docs.hazelcast.com/hazelcast/5.3/data-structures/creating-a-map

– На мові, яка має API-клієнт для Hazelcast, створіть Distributed Map:

```
if __name__ == "__main__":
    # Підключення до Hazelcast через Python-клієнта
    hz = hazelcast.HazelcastClient(cluster_name="distributed-map-cluster",
                                   client_name="hazelcast-connection")
    dist_map = hz.get_map("lab-distributed-map").blocking()
    print("Successful connection to Hazelcast!")
```

– Запишіть в неї 1000 значень з ключами від 0 до 1000:

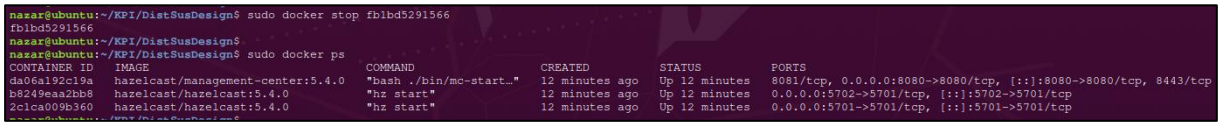
```
def distribute_1000keys():
    for key in range(1000): dist_map.set(key, key)
```

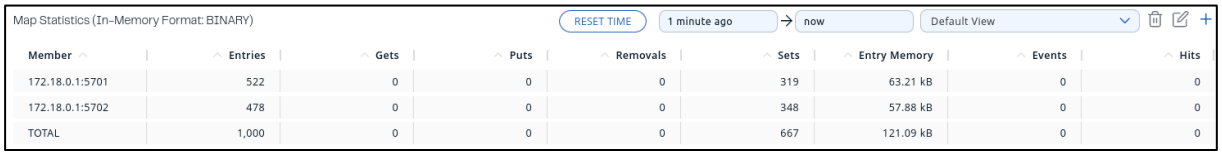
– За допомогою Management Center гляньте на розподіл ключів по нодах:

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Events	Hits
172.18.0.1:5701	319	0	0	0	319	38.63 kB	0	0
172.18.0.1:5702	348	0	0	0	348	42.14 kB	0	0
172.18.0.1:5703	333	0	0	0	333	40.32 kB	0	0
TOTAL	1,000	0	0	0	1,000	121.09 kB	0	0

– Подивитись як зміниться розподіл даних по нодах:

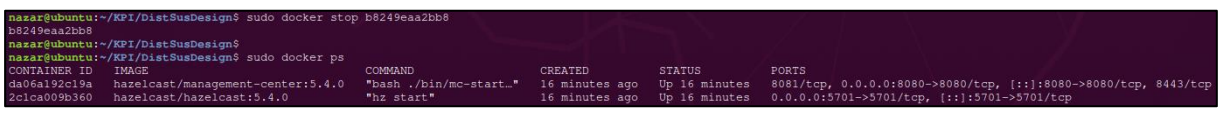
* Якщо відключити одну ноду

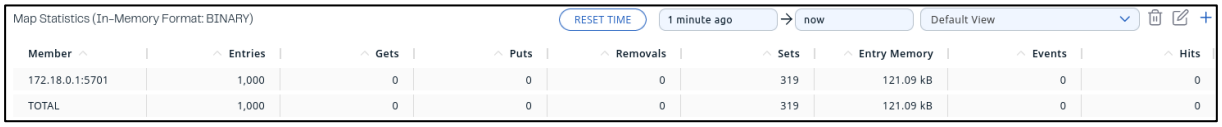




Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Events	Hits
172.18.0.1:5701	522	0	0	0	319	63.21 kB	0	0
172.18.0.1:5702	478	0	0	0	348	57.88 kB	0	0
TOTAL	1,000	0	0	0	667	121.09 kB	0	0

* Якщо відключити послідовно дві ноди





Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Events	Hits
172.18.0.1:5701	1,000	0	0	0	319	121.09 kB	0	0
TOTAL	1,000	0	0	0	319	121.09 kB	0	0

* Якщо відключити одночасно дві ноди (*емулюючи "падіння" серверів* – `kill`)

```
nazar@ubuntu:~/KPI/DistSusDesign$ sudo docker kill fb1bd5291566 b8249eaa2bb8
fb1bd5291566
b8249eaa2bb8
nazar@ubuntu:~/KPI/DistSusDesign$
nazar@ubuntu:~/KPI/DistSusDesign$ sudo docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED             STATUS          PORTS
da06a192c19a   hazelcast/management-center:5.4.0      "bash ./bin/mc-start…"   About an hour ago   Up 18 minutes   8081/tcp, 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp, 8443/tcp
2c1ca009b360   hazelcast/hazelcast:5.4.0              "hz start"               About an hour ago   Up 8 minutes    0.0.0.0:5701->5701/tcp, [::]:5701->5701/tcp
```

Map Statistics (In-Memory Format: BINARY) — RESET TIME — 1 minute ago → now — Default View

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Events	Hits
172.18.0.1:5701	668	0	0	0	343	80.89 kB	0	0
TOTAL	668	0	0	0	343	80.89 kB	0	0

* Чи була втрата даних?

  "`docker` **`stop`**": надіслано SIGTERM, що дозволило нодам завершити свою роботу із коректним перерозподілом та очисткою даних.

  "`docker` **`kill`**": надіслано SIGKILL, тим самим це миттєво відключило ноди без можливості перерозподілу, що призвело до втрати даних.

* Яким чином зробити щоб не було втрати даних?
  - ✓ Завжди використовувати "`docker` **`stop`**" при мануальному вимкнені нод, якщо немає крайньої необхідності використати "`docker` **`kill`**".
  - ✓ Налаштувати "**persistent volumes**" для зберігання даних поза контейнером, щоб уникнути їх втрати при непередбачуваних збоях.

4. Продемонструйте роботу "Locking Maps" в Distributed Map.

Clients — Default View — Address Type: Hostname — Search

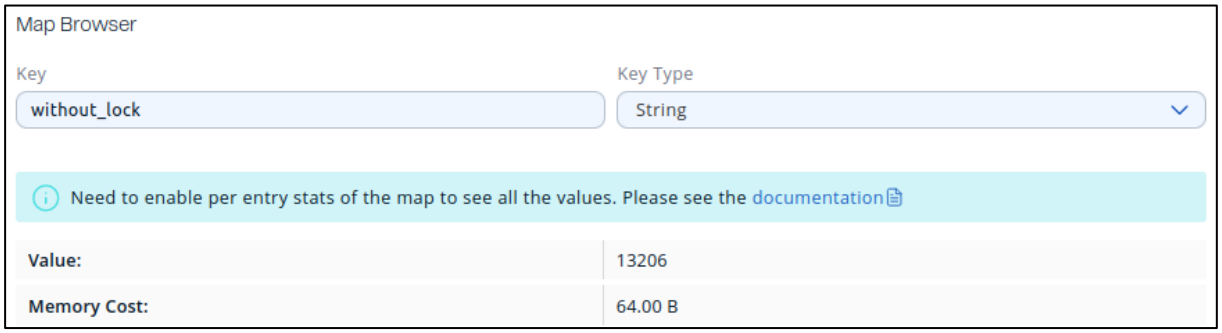
Name	Address	Type	Member Connection	Hazelcast Client Version	UUID
MC-Client-distributed-map-clus...	ubuntu.local	Management Center	ALL	5.4.0	4d3fd95b-d217-45dc-961e-4d4...
1hazelcast-connection	ubuntu.local	Python	N/A	5.4.0	e08353f1-05e8-40b7-ba3f-3258...
2hazelcast-connection	ubuntu.local	Python	N/A	5.4.0	2ae48a9d-9f83-4a8a-a62c-d668...
3hazelcast-connection	ubuntu.local	Python	N/A	5.4.0	32c7994e-9054-45a1-9fb6-215a...

a) Без жодного блокування

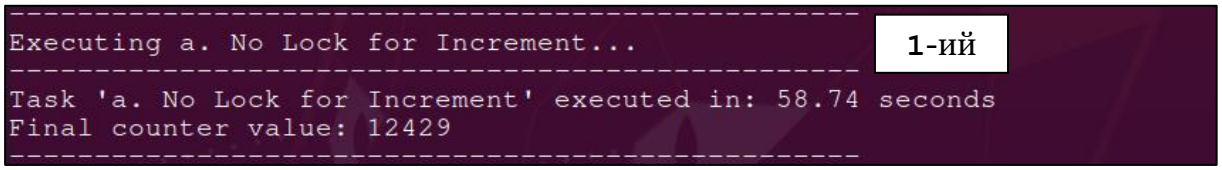
– Із 3-ох клієнтів, на кожному з них одночасно, запустіть інкремент значення для одного й того самого ключа в циклі на 10К ітерацій:

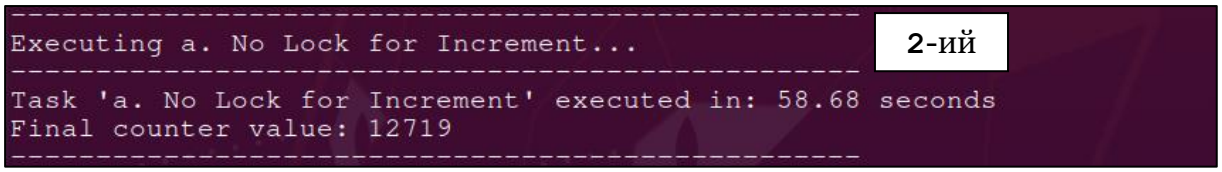
```python
# Функція для інкременту значення лічильника без Lock, з простим читанням і записом
def without_lock(key):
    dist_map.put(key, 0)
    for _ in range(10000):
        counter = dist_map.get(key) + 1
        dist_map.put(key, counter)
```

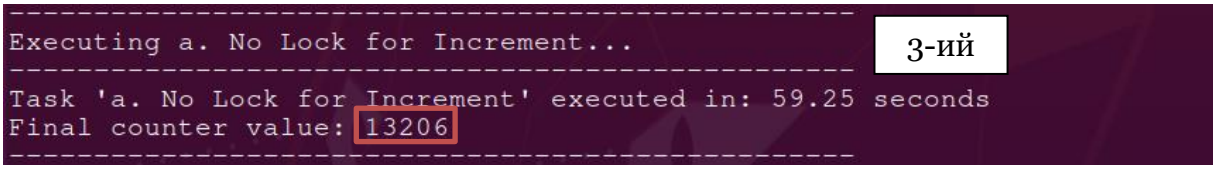
– Яке кінцеве значення для ключа “key” буде отримано (чи вийде 30К?)



– Які були отримані часові показники після виконання операції?





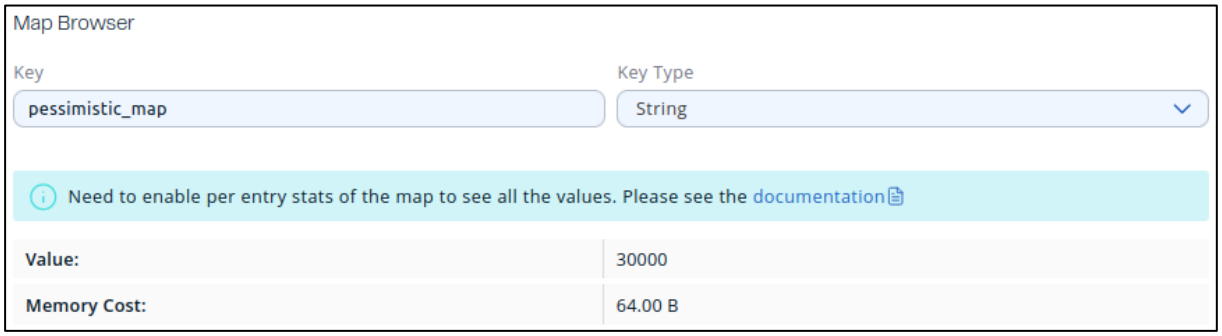


б) Песимістичне блокування

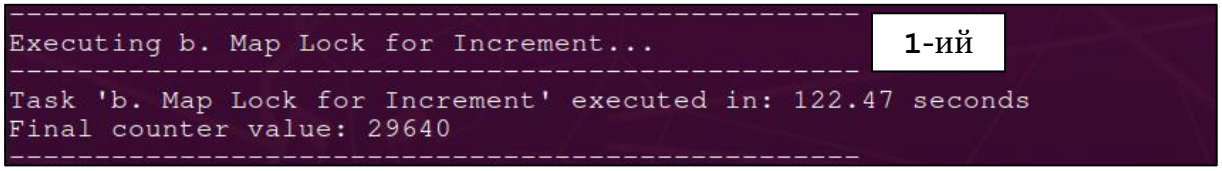
– Із 3-ох клієнтів, на кожному з них одночасно, запустіть інкремент значення для одного й того самого ключа в циклі на 10К ітерацій:

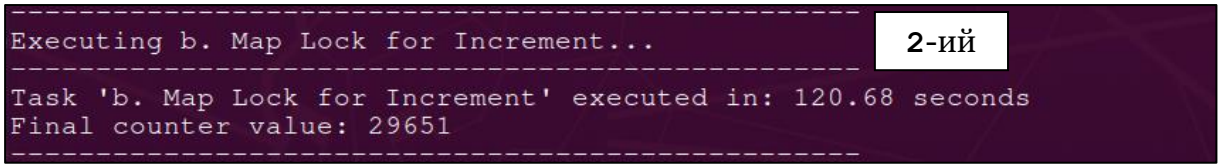
```
# Функція для інкременту значення лічильника з використанням Lock-мапи (map.lock)
def pessimistic_map(key):
    if (not dist_map.contains_key(key)): dist_map.put(key,0)
    for _ in range(10000):
        dist_map.lock(key)
        try:
            counter = dist_map.get(key) + 1
            dist_map.put(key, counter)
        finally:
            dist_map.unlock(key)
```

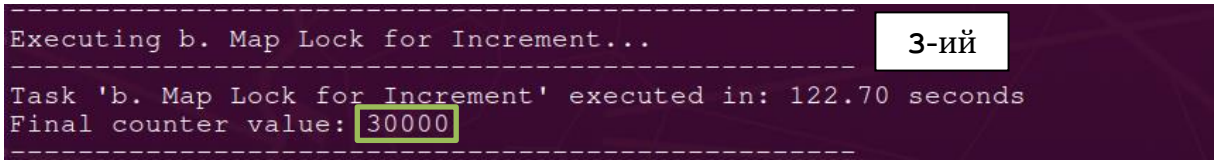
– Яке кінцеве значення для ключа “key” буде отримано (чи вийде 30К?)



– Які були отримані часові показники після виконання операції?





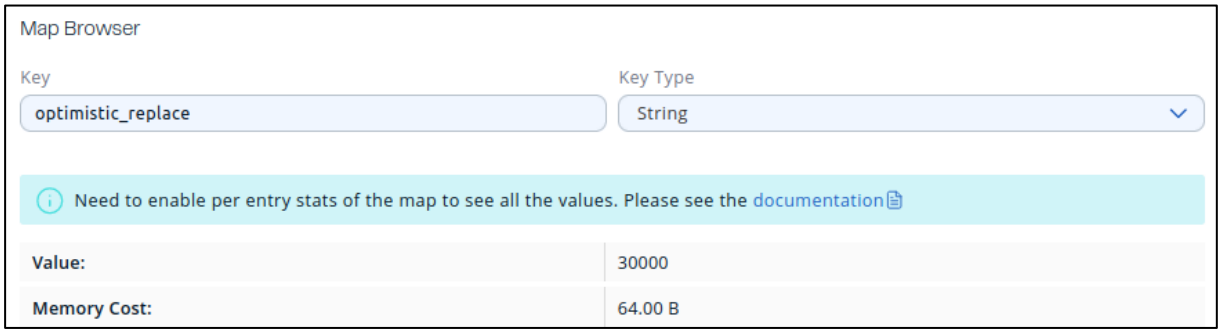


в) Оптимістичне блокування

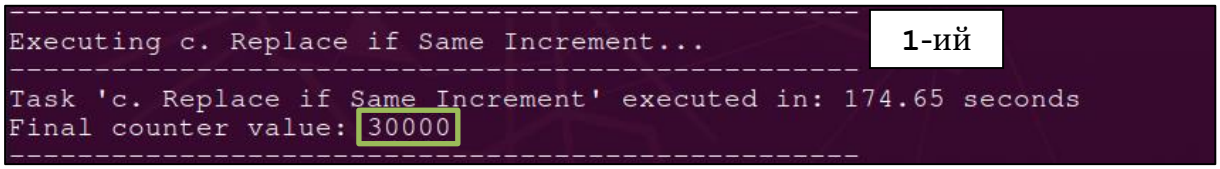
– Із 3-ох клієнтів, на кожному з них одночасно, запустіть інкремент значення для одного й того самого ключа в циклі на 10К ітерацій:

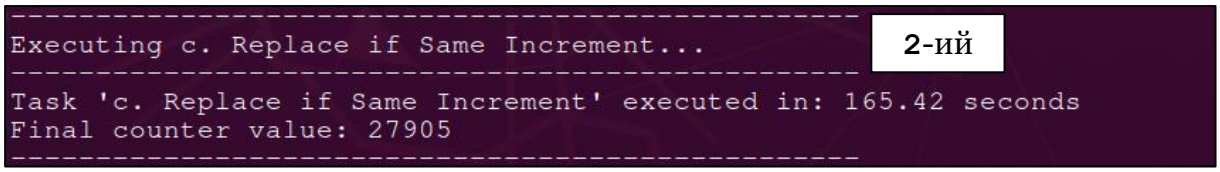
```python
# Функція для інкременту значення з використанням механізму заміни значень (replace_if_same)
def optimistic_replace(key):
    if (not dist_map.contains_key(key)): dist_map.put(key, 0)
    for _ in range(10000):
        while True:
            oldcounter = dist_map.get(key)
            newcounter = oldcounter + 1
            if dist_map.replace_if_same(key, oldcounter, newcounter): break
```

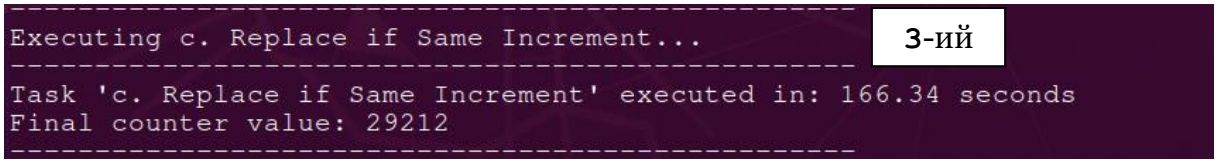
– Яке кінцеве значення для ключа “key” буде отримано (чи вийде 30К?)



– Які були отримані часові показники після виконання операції?







5. Порівняйте результати кожного з запусків

– Оцінка наявності втрат

* для реалізації без блокувань

Зафіксовано втрату даних: ***13206, замість 30000***.

* для реалізації з песимістичним та оптимістичним блокуванням

Отримано коректні результати: ***і там, і там 30000.***

– Оцінка часових показників

Песимістичний був швидший, ніж оптимістичний, проте вони обоє були повільніші за лічильник, який був реалізований без блокування.

6. Робота з “Bounded Queue”

– На основі “Distributed Queue” налаштуйте “Bounded Queue” на 10 елементів

> To turn a Hazelcast distributed queue into a bounded queue, set the capacity limit with the `max-size` property.

Зміни в конфігураційних файлах `hazelcast-docker.xml` на всіх 3-ох нодах:

```
/opt/hazelcast/config $ cat hazelcast-docker.xml | grep -in -A 8 "queue name"
205:    <queue name="default">
206-        <!--
207-            Maximum size of the queue. When a JVM's local queue size reaches the maximum,
208-            all put/offer operations will get blocked until the queue size
209-            of the JVM goes down below the maximum.
210-            Any integer between 0 and Integer.MAX_VALUE. 0 means
211-            Integer.MAX_VALUE. Default is 0.
212-        -->
213-        <max-size>10</max-size>
```

– Запустіть одного клієнта який буде писати в чергу значення 1...100, а двох інших які будуть паралельно читати з черги

1) Один клієнт, який записує:

* **writer**-bounded_queue.py

```python
import hazelcast

hz = hazelcast.HazelcastClient(cluster_name="distributed-map-cluster")
bounded_queue = hz.get_queue("lab-distributed-queue").blocking()
print("Successful connection to Hazelcast!")

for item in range(1, 101):
    bounded_queue.put(item)
    print(f"| Item: {item}\t | Size: {bounded_queue.size()}\t |\n"
          f"| Remaining Capacity: [{bounded_queue.remaining_capacity()}]\t |\n")

bounded_queue.put(-1)
hz.shutdown()
```

```
nazar@ubuntu:~/KPI/DistSusDesign$ python3 writer-bounded_queue.py
Successful connection to Hazelcast!
| Item: 0         | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 1         | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 2         | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 3         | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 4         | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 5         | Size: 0         |
| Remaining Capacity: [10]         |
```

Аргумент **size** постійно рівний **0**, **remaining_capacity** залишається **10**.

```
| Item: 95        | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 96        | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 97        | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 98        | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 99        | Size: 0         |
| Remaining Capacity: [10]         |

| Item: 100       | Size: 0         |
| Remaining Capacity: [10]         |
```

Це свідчить про те, що кожне повідомлення вичитується одразу.

2) Два клієнти, які читають

* **reader**-bounded_queue.py

```python
import hazelcast

hz = hazelcast.HazelcastClient(cluster_name="distributed-map-cluster")
bounded_queue = hz.get_queue("lab-distributed-queue").blocking()
print("Successful connection to Hazelcast!")

item = 0
while item != -1:
    item = bounded_queue.take()
    print(f"> Taken from queue: {item} \n")

else:
    bounded_queue.put(-1)
    hz.shutdown()
```

```
nazar@ubuntu:~/KPI/DistSusDesign$ python3 reader-bounded_queue.py
Successful connection to Hazelcast!
> Taken from queue: 1

> Taken from queue: 3

> Taken from queue: 5

> Taken from queue: 7

> Taken from queue: 9

> Taken from queue: 91

> Taken from queue: 93

> Taken from queue: 95

> Taken from queue: 97

> Taken from queue: 99
```

1-ий клієнт

```
nazar@ubuntu:~/KPI/DistSusDesign$ python3 reader-bounded_queue.py
Successful connection to Hazelcast!
> Taken from queue: 2

> Taken from queue: 4

> Taken from queue: 6

> Taken from queue: 8

> Taken from queue: 10

> Taken from queue: 92

> Taken from queue: 94

> Taken from queue: 96

> Taken from queue: 98

> Taken from queue: 100
```

2-ий клієнт

– Яким чином вичитувалися значення з черги двома клієнтами?

Значення вичитувалися клієнтами по черзі згідно з принципом FIFO (First-In-First-Out), а саме: 1-ий клієнт читав за порядком всі непарні значення, а 2-ий всі парні. Таким чином, клієнти почергово зчитували дані без дублювання.

– Яка буде поведінка на запис якщо відсутнє читання, і черга заповнена?

* Жоден `reader` не ввімкнений:

```
nazar@ubuntu:~/KPI/DistSusDesign$ python3 writer-bounded_queue.py
Successful connection to Hazelcast!
| Item: 1         | Size: 1         |
| Remaining Capacity: [9]           |

| Item: 2         | Size: 2         |
| Remaining Capacity: [8]           |

| Item: 3         | Size: 3         |
| Remaining Capacity: [7]           |

| Item: 4         | Size: 4         |
| Remaining Capacity: [6]           |

| Item: 5         | Size: 5         |
| Remaining Capacity: [5]           |

| Item: 6         | Size: 6         |
| Remaining Capacity: [4]           |

| Item: 7         | Size: 7         |
| Remaining Capacity: [3]           |

| Item: 8         | Size: 8         |
| Remaining Capacity: [2]           |

| Item: 9         | Size: 9         |
| Remaining Capacity: [1]           |

| Item: 10        | Size: 10        |
| Remaining Capacity: [0]           |
```

Черга заповнена, тому вільних місць для запису в ній більше немає.

Queues

Search

Name	Persistence	Items	Backups	Max Age	Min Age	Average Age
lab-distributed-queue	Disabled	10	10	30m 7s 406ms	1ms	34s 289ms

* Один `reader` знову ввімкнений:

```
| Item: 11        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 12        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 13        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 14        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 15        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 16        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 17        | Size: 10        |
| Remaining Capacity: [0]           |

| Item: 18        | Size: 10        |
| Remaining Capacity: [0]           |
```

Черга досі заповнюється при вичитуванні з неї 1-им клієнтом.

```
| Item: 19        | Size: 10        |
| Remaining Capacity: [1]           |

| Item: 20        | Size: 9         |
| Remaining Capacity: [2]           |

| Item: 21        | Size: 7         |
| Remaining Capacity: [4]           |

| Item: 22        | Size: 5         |
| Remaining Capacity: [5]           |

| Item: 23        | Size: 4         |
| Remaining Capacity: [6]           |

| Item: 24        | Size: 3         |
| Remaining Capacity: [8]           |

| Item: 25        | Size: 2         |
| Remaining Capacity: [9]           |

| Item: 26        | Size: 1         |
| Remaining Capacity: [10]          |

| Item: 27        | Size: 0         |
| Remaining Capacity: [10]          |

| Item: 28        | Size: 0         |
| Remaining Capacity: [10]          |
```

Той клієнт, що вичитує робить це швидше, ніж той який записує.