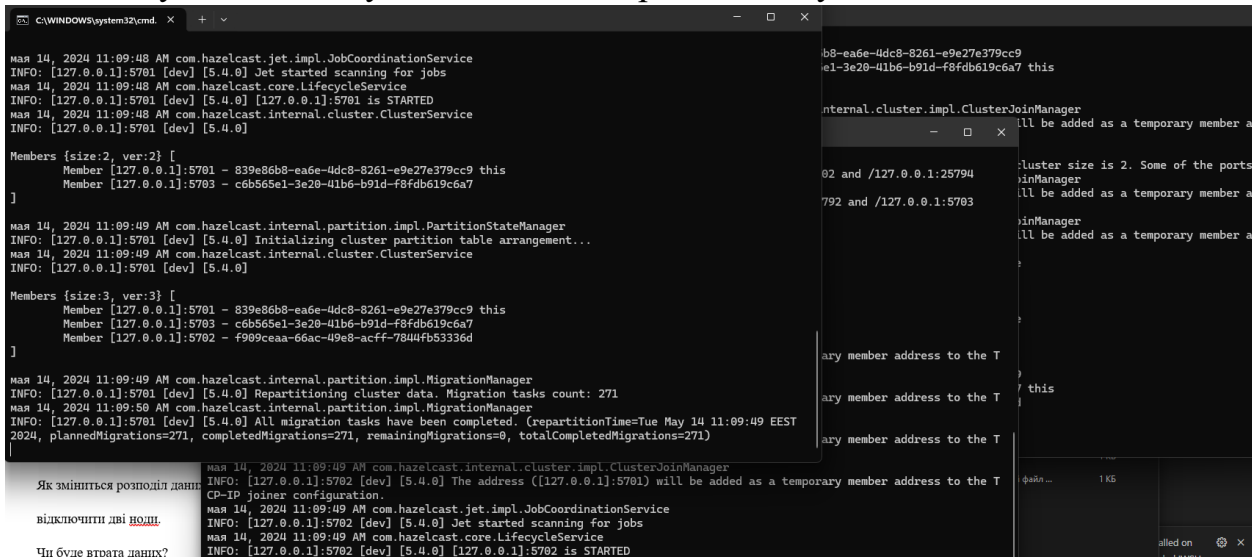


Task 2

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

Завдання:

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



The screenshot shows a terminal window with Hazelcast logs and a GUI window displaying cluster members. The logs indicate that three nodes are starting and joining the cluster. The GUI window shows the cluster members with their addresses and IDs.

```
may 14, 2024 11:09:48 AM com.hazelcast.jet.impl.JobCoordinationService
INFO: [127.0.0.1]:5701 [dev] [5.4.0] Jet started scanning for jobs
may 14, 2024 11:09:48 AM com.hazelcast.core.LifecycleService
INFO: [127.0.0.1]:5701 [dev] [5.4.0] [127.0.0.1]:5701 is STARTED
may 14, 2024 11:09:48 AM com.hazelcast.internal.cluster.ClusterService
INFO: [127.0.0.1]:5701 [dev] [5.4.0]

Members {size:2, ver:2} [
  Member [127.0.0.1]:5701 - 839e86b8-ea6e-4dc8-8261-e9e27e379cc9 this
  Member [127.0.0.1]:5703 - c6b565e1-3e20-41b6-b91d-f8fdb619c6a7
]

may 14, 2024 11:09:49 AM com.hazelcast.internal.partition.impl.PartitionStateManager
INFO: [127.0.0.1]:5701 [dev] [5.4.0] Initializing cluster partition table arrangement...
may 14, 2024 11:09:49 AM com.hazelcast.internal.cluster.ClusterService
INFO: [127.0.0.1]:5701 [dev] [5.4.0]

Members {size:3, ver:3} [
  Member [127.0.0.1]:5701 - 839e86b8-ea6e-4dc8-8261-e9e27e379cc9 this
  Member [127.0.0.1]:5703 - c6b565e1-3e20-41b6-b91d-f8fdb619c6a7
  Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb53336d
]

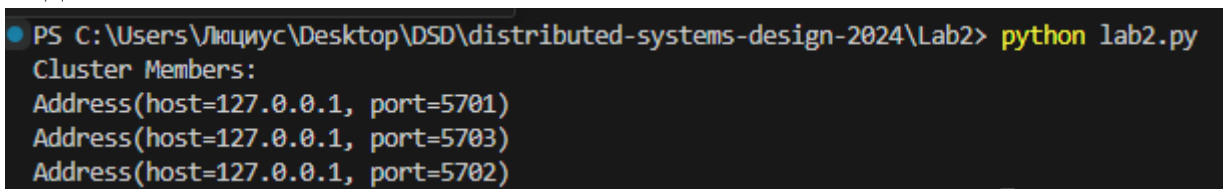
may 14, 2024 11:09:49 AM com.hazelcast.internal.partition.impl.MigrationManager
INFO: [127.0.0.1]:5701 [dev] [5.4.0] Repartitioning cluster data, Migration tasks count: 271
may 14, 2024 11:09:50 AM com.hazelcast.internal.partition.impl.MigrationManager
INFO: [127.0.0.1]:5701 [dev] [5.4.0] All migration tasks have been completed. (repartitionTime=Tue May 14 11:09:49 EEST
2024, plannedMigrations=271, completedMigrations=271, remainingMigrations=0, totalCompletedMigrations=271)

may 14, 2024 11:09:49 AM com.hazelcast.internal.cluster.impl.ClusterJoinManager
INFO: [127.0.0.1]:5702 [dev] [5.4.0] The address ([127.0.0.1]:5701) will be added as a temporary member address to the T
CP-IP joiner configuration.
may 14, 2024 11:09:49 AM com.hazelcast.jet.impl.JobCoordinationService
INFO: [127.0.0.1]:5702 [dev] [5.4.0] Jet started scanning for jobs
may 14, 2024 11:09:49 AM com.hazelcast.core.LifecycleService
INFO: [127.0.0.1]:5702 [dev] [5.4.0] [127.0.0.1]:5702 is STARTED
```

3. Продемонструйте роботу Distributed Map
Використовуючи API створіть Distributed Map запишіть в неї 1000
значень з ключем від 0 до 1

```
def task3():
    map = client.get_map("map").blocking()
    for i in range(1, 1001):
        map.put(i, random.randint(1, 1000))
```

за допомогою Management Center подивиться на розподіл значень по
нодах



The screenshot shows a terminal window with the output of the 'python lab2.py' command. The output displays the cluster members and their addresses.

```
PS C:\Users\Люциус\Desktop\DSO\distributed-systems-design-2024\Lab2> python lab2.py
Cluster Members:
Address(host=127.0.0.1, port=5701)
Address(host=127.0.0.1, port=5703)
Address(host=127.0.0.1, port=5702)
```

Map Statistics (In-Memory Format: BINARY)				RESET TIME
Member ^	^ Entries	^ Gets	^ Puts	
127.0.0.1:5701	346	346	346	
127.0.0.1:5702	314	314	314	
127.0.0.1:5703	340	340	340	
TOTAL	1 000	1 000	1 000	

подивитись як зміниться розподіл даних по нодах:
якщо відключити одну ноду

Map Statistics (In-Memory Format: BINARY)				RESET TIME
Member ^	^ Entries	^ Gets	^ Puts	
127.0.0.1:5702	488	314	314	
127.0.0.1:5703	512	340	340	
TOTAL	1 000	654	654	

відключити дві ноди.

Map Statistics (In-Memory Format: BINARY)				RESET TIME
Member ^	^ Entries	^ Gets	^ Puts	
127.0.0.1:5702	1 000	314	314	
TOTAL	1 000	314	314	

Чи буде втрата даних?

Втрат немає

4. Продемонструйте роботу з Topic

Запустіть одного клієнта який буде писати в Topic значення 1..100, а двох інших які будуть читати з Topic

Publisher

```
def task4():
    topic = client.get_topic("my_topic").blocking()

    for i in range(1, 101):
        topic.publish(i)
```

Readers

```

def on_message(message):
    print("Received message:", message)
def task4():
    print("Task4")
    topic = client.get_topic("my_topic").blocking()
    topic.add_listener(on_message)
    try:
        while True:
            pass
    except KeyboardInterrupt:
        pass

```

```

PS C:\Users\Ляцунс\Desktop\DSD\distributed-systems-design-2024\Lab2> python clients
.py
Task4
Received message: TopicMessage(message=1, publish_time=1715675358.123, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=2, publish_time=1715675358.124, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=3, publish_time=1715675358.124, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=4, publish_time=1715675358.124, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=5, publish_time=1715675358.125, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=6, publish_time=1715675358.125, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=7, publish_time=1715675358.125, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=8, publish_time=1715675358.126, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)
Received message: TopicMessage(message=9, publish_time=1715675358.126, topic_name=m
y_topic, publishing_member=Member [127.0.0.1]:5702 - f909ceaa-66ac-49e8-acff-7844fb
53336d)

```

яким чином будуть вичитуватись значення з Топіс двома клієнтами?

Коли повідомлення публікується у топіс, воно реплікується на всі вузли кластера, де є підписники на цей топіс.

якщо один з читачів буде певний час неактивний, чи отримає він повідомлення які він пропустив?

Коли повідомлення публікується у топіс, воно зберігається у внутрішній черзі на короткий час. Тому, якщо один з підписників топіс був певний час неактивний, то він не отримає повідомлення, які були опубліковані під час його простою.

5. Робота з Bounded queue

На основі Distributed Queue налаштуйте Bounded queue на 10 елементів Publisher

```
def task5():
    queue = client.get_queue("default").blocking()

    for i in range(1, 101):
        queue.offer(i, 10)
        # time.sleep(1)
```

Readers

```
def read_from_queue():
    queue = client.get_queue("default").blocking()
    try:
        i = 1
        while True:
            poll = queue.poll()
            if poll != None:
                print(f"{i}. Item: {poll}")
                i += 1
    except KeyboardInterrupt:
        pass

def task5():
    read_from_queue()
```

hazelcast.xml

```
<queue name="default">
    <!--
        Maximum size of the queue. When the size of the queue reaches this value,
        all put/offer operations will fail with an OutOfMemoryError. If the
        memory of the JVM goes down below the threshold, the queue will be
        resized. Any integer between 0 and Integer.MAX_VALUE. Default is
        Integer.MAX_VALUE.
    -->
    <max-size>10</max-size>
</queue>
```

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

```
PS C:\Users\Юлія\c\Desktop\DSD\distributed-systems-design-2024\Lab2> python clients
.py
1. Item: 1
2. Item: 3
3. Item: 5
4. Item: 6
5. Item: 8
6. Item: 10
7. Item: 12
8. Item: 14
9. Item: 15
10. Item: 18
11. Item: 19
12. Item: 21
13. Item: 22
14. Item: 25
15. Item: 27
16. Item: 28
17. Item: 29
18. Item: 31
19. Item: 33
20. Item: 34
21. Item: 36
22. Item: 39
23. Item: 41
24. Item: 42
25. Item: 44
26. Item: 45
27. Item: 49
28. Item: 50
29. Item: 52
30. Item: 53
31. Item: 54
32. Item: 56
33. Item: 57
34. Item: 60
35. Item: 61
36. Item: 63
37. Item: 65
38. Item: 67
39. Item: 68
40. Item: 69
41. Item: 72
42. Item: 75
43. Item: 77
44. Item: 79
45. Item: 81
46. Item: 83
47. Item: 86
48. Item: 89
49. Item: 91
50. Item: 93
51. Item: 96
52. Item: 98
53. Item: 100
```

```
1. Item: 2
2. Item: 4
3. Item: 7
4. Item: 9
5. Item: 11
6. Item: 13
7. Item: 16
8. Item: 17
9. Item: 20
10. Item: 23
11. Item: 24
12. Item: 26
13. Item: 30
14. Item: 32
15. Item: 35
16. Item: 37
17. Item: 38
18. Item: 40
19. Item: 43
20. Item: 46
21. Item: 47
22. Item: 48
23. Item: 51
24. Item: 55
25. Item: 58
26. Item: 59
27. Item: 62
28. Item: 64
29. Item: 66
30. Item: 70
31. Item: 71
32. Item: 73
33. Item: 74
34. Item: 76
35. Item: 78
36. Item: 80
37. Item: 82
38. Item: 84
39. Item: 85
40. Item: 87
41. Item: 88
42. Item: 90
43. Item: 92
44. Item: 94
45. Item: 95
46. Item: 97
47. Item: 99
□
```

яким чином будуть вичитуватись значення з черги двома клієнтами?

Вичитування буде відбуватися за принципом «хто встиг той з'їв», де один елемент черги забирається лише одним читачем.

перевірте яка буде поведінка на запис якщо відсутнє читання, і черга заповнена

```
PS C:\Users\Люциус\Desktop\DSD\distributed-systems-design-2024\Lab2> python lab2.py
Cluster Members:
Address(host=127.0.0.1, port=5702)
Address(host=127.0.0.1, port=5701)
Address(host=127.0.0.1, port=5703)
█
```

Запис у чергу припиняється поки не звільниться місце.

GitHub:

<https://github.com/DruzDanil/distributed-systems-design-2024/tree/main/Lab2>