

Homework 2

Protocol

Task 1

Regarding the settings, it was necessary to change in `hazelcast.xml` the `cluster-name` :

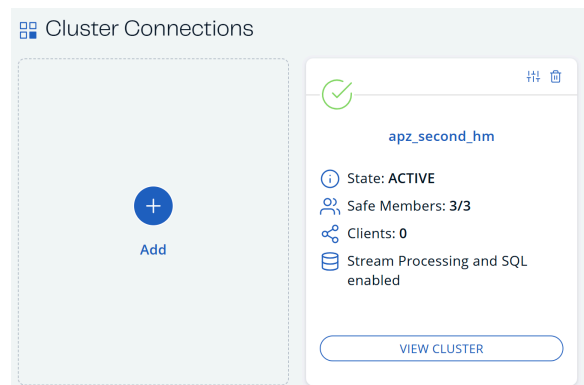
```
<cluster-name>apz_second_hm</cluster-name>
```

That's all 🙄

Task 2

To launch 3 instances, I ran `bin/hz-start.bat` in three terminals and it added them to one cluster due to the same cluster name in the configuration.

```
Members {size:3, ver:3} [
  Member [127.0.0.1]:5701 - 1126bbbd-c517-4a50-935a-af6cb9878b28
  Member [127.0.0.1]:5702 - f7f91aac-31b9-4745-8b44-610784850f5f
  Member [127.0.0.1]:5703 - a92678b8-a17f-4ec1-b430-a1d026507fad this
]
```





Task 3


To write 1000 values, you need to run the command:




```
python3 ./task3/distributed_map.py
```

It will create a map, which can be visible in the **Storage/Maps** section:


Maps 

Search 

Default View 


   Show System Maps ☐ OFF


Name ^	Persistence ^	In Memory F... ^	Entries ^	Entry Memory ^	Backup Mem ^
distributed_example	Disabled	BINARY	1,000	128.65 kB	128.65 k




1 - 1 of 1 Rows 10 

This is the distribution (they have almost the same number of entries):


Map Statistics (In-Memory Format: BINARY)

 1 minute ago → now

Default View 

Member ^	Entries ^	Gets ^	Puts ^	Removals ^	Sets ^
127.0.0.1:5701	338	0	0	0	338
127.0.0.1:5702	319	0	0	0	319
127.0.0.1:5703	343	0	0	0	343
TOTAL	1,000	0	0	0	1,000

1 - 3 of 3 Rows 10 

When one node is down, we can see that the distribution has changed, but there has been no loss:

Map Statistics (In-Memory Format: BINARY)

RESET TIME

1 minute ago

→ now

Default View

Member ^	Entries ^	Gets ^	Puts ^	Removals ^	Sets ^
127.0.0.1:5701	514	0	0	0	338
127.0.0.1:5702	486	0	0	0	319
TOTAL	1,000	0	0	0	657

1 - 2 of 2 Rows 10

When two nodes are disconnected in sequence, the values are also preserved, i.e. without loss:

Map Statistics (In-Memory Format: BINARY)

RESET TIME

1 minute ago

→ now

Default View

Member ^	Entries ^	Gets ^	Puts ^	Removals ^	Sets ^
127.0.0.1:5701	1,000	0	0	0	338
TOTAL	1,000	0	0	0	338

1 - 1 of 1 Rows 10

In order to stop two at the same time, I decided to stop their processes. For this I looked at which processes were running related to **hazelcast** and selected two random nodes (more precisely, the id of their processes) and forcibly stopped them:

```
PS C:\Users\Irunu> Get-WmiObject Win32_Process | Where-Object { $_.CommandLine -match "hazelcast" } | Select-Object ProcessId, CommandLine
ProcessId CommandLine
-----
18608 C:\WINDOWS\system32\cmd.exe /c ""C:\Users\Irunu\AppData\Local\hazelcast-5.5\bin\hz-start.bat""
9600 C:\Program Files\Java\jdk-20\bin\java --add-modules java.se --add-exports java.base/jdk.internal.ref=ALL-UNNAMED --add-opens java.b...
9620 C:\WINDOWS\system32\cmd.exe /c ""C:\Users\Irunu\AppData\Local\hazelcast-5.5\management-center\bin\start.bat""
4280 java -cp "hazelcast-management-center-5.5.jar org.springframework.boot.loader.Launcher Launch PropertiesLauncher
7072 C:\Program Files\Java\jdk-20\bin\java.exe -cp hazelcast-management-center-5.5.jar org.springframework.boot.loader.Launcher Launch Properties...
9136 C:\WINDOWS\system32\cmd.exe /c ""C:\Users\Irunu\AppData\Local\hazelcast-5.5\bin\hz-start.bat""
7136 C:\Program Files\Java\jdk-20\bin\java --add-modules java.se --add-exports java.base/jdk.internal.ref=ALL-UNNAMED --add-opens java.b...
7228 C:\WINDOWS\system32\cmd.exe /c ""C:\Users\Irunu\AppData\Local\hazelcast-5.5\bin\hz-start.bat""
7960 C:\Program Files\Java\jdk-20\bin\java --add-modules java.se --add-exports java.base/jdk.internal.ref=ALL-UNNAMED --add-opens java.b...
```

```
PS C:\Users\Irunu> Stop-Process -Id 9600, 74136 -Force
PS C:\Users\Irunu> Get-WmiObject Win32_Process | Where-Object { $_.CommandLine -match "hazelcast" } | Select-Object ProcessId, CommandLine
ProcessId CommandLine
-----
9620 C:\WINDOWS\system32\cmd.exe /c ""C:\Users\Irunu\AppData\Local\hazelcast-5.5\management-center\bin\start.bat""
4280 java -cp "hazelcast-management-center-5.5.jar org.springframework.boot.loader.Launcher Launch PropertiesLauncher
7072 C:\Program Files\Java\jdk-20\bin\java.exe -cp hazelcast-management-center-5.5.jar org.springframework.boot.loader.Launcher Launch Properties...
7228 C:\WINDOWS\system32\cmd.exe /c ""C:\Users\Irunu\AppData\Local\hazelcast-5.5\bin\hz-start.bat""
7960 C:\Program Files\Java\jdk-20\bin\java --add-modules java.se --add-exports java.base/jdk.internal.ref=ALL-UNNAMED --add-opens java.b...
PS C:\Users\Irunu>
```

As a result, we can see that a loss has occurred:

Map Statistics (In-Memory Format: BINARY)

RESET TIME 1 minute ago → now

Default View

Member ^	Entries ^	Gets ^	Puts ^	Removals ^	Sets ^
127.0.0.1:5703	663	0	0	0	0
TOTAL	663	0	0	0	0

1 - 1 of 1 Rows 10

#TODO пояснити чому та як це можна виправити

Task 4-7

For this task, I made a launch from one file, where you can pass arguments to specify which type to run/test.

To run **without locks**:

```
python3 .\run_incrementing_together.py --test_type=without_locks.py
```

The result:

Key	Key Type
key	String
<i>i</i> Need to enable per entry stats of the map to see all the values. Please see the documentation	
Value:	12405
Memory Cost:	64 B
Expiration Time:	N/A
Last Access Time:	N/A
Last Stored Time:	N/A
Time to Live:	Unlimited
Key Owner Member:	127.0.0.1:5701
Class:	java.lang.Integer
Creation Time:	N/A
Hits:	N/A
Last Update Time:	N/A
Version:	30000
Max Idle:	Unlimited

As a result, a very strong loss is noticeable. This is due to the fact that they work at the same time reading an outdated value (i.e., an outdated one). A race condition occurs, that is, part of the increments is simply lost.

For example, one process updated the value, did not have time to read the new one, as another process updated the outdated value and wrote the old one again.

To run with the **pessimistic blocking**:

```
python3 .\run_incrementing_together.py --test_type=pessimistic_locks.py
```

Key	Key Type
key	String
<i>Need to enable per entry stats of the map to see all the values. Please see the documentation</i>	
Value:	30000
Memory Cost:	64 B
Expiration Time:	N/A
Last Access Time:	N/A
Last Stored Time:	N/A
Time to Live:	Unlimited
Key Owner Member:	127.0.0.1:5701
Class:	java.lang.Integer
Creation Time:	N/A
Hits:	N/A
Last Update Time:	N/A
Version:	30000
Max Idle:	Unlimited

Pessimistic locking means that before updating the value of "key", we "lock" it so that other processes cannot change it at the same time. This ensures that no increments are lost, because only one process at a time updates the "key". However, this can be slow if many processes want to change the value at the same time.

As we can see, there were no losses (value is 30 000), but compared to the previous launch, it worked significantly longer.

(Below a table with the exact recorded time)

To run with the **optimistic blocking**:

```
python3 .\run_incrementing_together.py --test_type=optimistic_locks.py
```

Key	Key Type
key	String
Need to enable per entry stats of the map to see all the values. Please see the documentation	
Value:	30000
Memory Cost:	64 B
Expiration Time:	N/A
Last Access Time:	N/A
Last Stored Time:	N/A
Time to Live:	Unlimited
Key Owner Member:	127.0.0.1:5701
Class:	java.lang.Integer
Creation Time:	N/A
Hits:	N/A
Last Update Time:	N/A
Version:	30000
Max Idle:	Unlimited

Optimistic locking means that we don't lock the "key", but simply check if it has changed during the update. This method is faster than pessimistic locking because it doesn't make other processes wait. However, if many processes frequently change the "key", there may be many retries (`while True`).

But the result also successfully converged to 30,000.

Note: between tests I used a command `python3 ./clean.py` to remove the key and make testing valid.

Comparison of results:

Test Type	Total Time (s)	Value
Without Locks	7.8668	12405
Pessimistic Locks	54.2212	30000
Optimistic Locks	22.7039	30000

- Without locks, values are lost a lot (12405 instead of 30000) because threads simultaneously read and modify the same value, overwriting it.
 - Pessimistic locking guarantees the correct result (30000), but is slow (54.22 s) because threads wait for others to release the resource.
 - Optimistic locking also gives the correct result (30000), but is faster (22.70 s) because threads check if the value has changed and try again without waiting for a lock.
-

Task 8

First of all, it was needed to add to the configuration of the `hazelcast` such a part:

```
<queue name="ten-bounded-version-queue">  
  <max-size>10</max-size>  
</queue>
```

I have a common code, I run subprocesses, where two clients in the cluster are responsible for reading, and the other for writing. In general, I thought about making a counter through shared memory to know when to do `client.shutdown()`, but initially I did it through `subprocess.Popen`, which slightly prevents it from doing it normally, so I decided to count on the fact that they +- count the same number of numbers.

As a result we get such output:


```
[client write 127.0.0.1:5701] added value 51
[client read 127.0.0.1:5702] read value 51
[client write 127.0.0.1:5701] added value 52
[client read 127.0.0.1:5703] read value 52
[client write 127.0.0.1:5701] added value 53
[client read 127.0.0.1:5702] read value 53
[client write 127.0.0.1:5701] added value 54
[client read 127.0.0.1:5703] read value 54
[client write 127.0.0.1:5701] added value 55
[client read 127.0.0.1:5702] read value 55
```

Sometimes it might happen such a thing:

```
[client read 127.0.0.1:5702] read value 29
[client write 127.0.0.1:5701] added value 29
```

But that's okay, since it asynchronously writes the outputs of three processes to the terminal, there may be cases where it wrote a value, read it, wrote what it read, wrote what it wrote.

The values will be read in such a way that each queue element can only be read by one client. Therefore, if two readers are working at the same time, they will distribute the values between them randomly depending on who executes `take()` first.

I added a feature that if the queue is full, they will retry, writing about it to the terminal:

```
[client write 127.0.0.1:5701] added value 1
[client write 127.0.0.1:5701] added value 2
[client write 127.0.0.1:5701] added value 3
[client write 127.0.0.1:5701] added value 4
[client write 127.0.0.1:5701] added value 5
[client write 127.0.0.1:5701] added value 6
[client write 127.0.0.1:5701] added value 7
[client write 127.0.0.1:5701] added value 8
[client write 127.0.0.1:5701] added value 9
[client write 127.0.0.1:5701] added value 10
Full queue, retrying in 2 seconds (Total Tetries 0)
Full queue, retrying in 2 seconds (Total Tetries 1)
Full queue, retrying in 2 seconds (Total Tetries 2)
```

That's it, the end 😊