Task 1: installed hazelcast ✅

Task 2:

Member	CPU	Used…	Com…	Heap…	Used…	Com…	GC …	GC Major…	GC …	GC Minor…
127.0.0.1:5701	30.74 %	87.91 MB	209.00 MB	42.06 %	0 B	0 B	0	0ms	6	42ms
127.0.0.1:5702	34.22 %	81.50 MB	221.00 MB	36.88 %	0 B	0 B	0	0ms	6	59ms
127.0.0.1:5703	32.37 %	78.32 MB	224.00 MB	34.96 %	0 B	0 B	0	0ms	6	67ms

1 – 3 of 3 Rows 10

Task 3:

1 node:

Map Statistics (In-Memory Format: BINARY) Now

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Backups
127.0.0.1:5701	1000	0	0	0	2795	129.78 kB	0
TOTAL	1000	0	0	0	2795	129.78 kB	0

1 – 1 of 1 Rows 10

2 nodes:

Map Statistics (In-Memory Format: BINARY) Now

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Backups
127.0.0.1:5701	489	0	0	0	1467	63.46 kB	511
127.0.0.1:5702	511	0	0	0	1533	66.31 kB	489
TOTAL	1000	0	0	0	3000	129.78 kB	1000

1 – 2 of 2 Rows 10

3 nodes:

Map Statistics (In-Memory Format: BINARY) Now

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Backups
127.0.0.1:5701	328	0	0	0	1795	42.57 kB	340
127.0.0.1:5702	350	0	0	0	1883	45.43 kB	327
127.0.0.1:5703	322	0	0	0	322	41.78 kB	333
TOTAL	1000	0	0	0	4000	129.78 kB	1000

1 – 3 of 3 Rows 10

As we can see, there are no data losses, and data is distributed almost equally.

Task 4:

No locks:

```
sophmintaii@tardis:~/Documents/microservises-architecture$ python3 map_no_locks.py
result = 1000
time taken = 13.938880681991577
```

Map Browser

Key: 1

Key Type: String

Value:	1000	**Class:**	java.lang.String
Memory Cost:	68.00 B	**Creation Time:**	N/A
Expiration Time:	N/A	**Hits:**	-1
Last Access Time:	N/A	**Last Update Time:**	N/A
Last Stored Time:	N/A	**Version:**	1000
Time to Live:	292471208 years and 8 months	**Max Idle:**	Unlimited
Key Owner Member:	127.0.0.1:5703		

Cancel BROWSE

(no race condition)

Optimistic locks:

```
sophmintaii@tardis:~/Documents/microservises-architecture$ python3 map_optimistic.py
result = 1000
time taken = 14.660603046417236
```

Pessimistic locks:

```
sophmintaii@tardis:~/Documents/microservises-architecture$ python3 map_pessimistic.py
result = 1000
time taken = 16.02294158935547
```

The map without locks is the fastest.

Task 5:

In case there is no reading, and the space in queue ended, the writer will wait until it frees to write all the values:

```
sophmintaii@tardis:~/Documents/microservises-architecture$ python3 write.py
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
waiting for free space
^CTraceback (most recent call last):
  File "/home/sophmintaii/Documents/microservises-architecture/write.py", line 20, in
<module>
    time.sleep(1)
KeyboardInterrupt
```

for two readers and one writer, the writer is going to write all the values because readers take them, and readers are going to take different values: (screenshot the part where there are already two readers running)

```
97_value
99_value
101_value
103_value
105_value
107_value
109_value
111_value
113_value
115_value
117_value
119_value
121_value
123_value
125_value
127_value
129_value
131_value
133_value
135_value
137_value
139_value
141_value
143_value
145_value
147_value
```

```
96_value
98_value
100_value
102_value
104_value
106_value
108_value
110_value
112_value
114_value
116_value
118_value
120_value
122_value
124_value
126_value
128_value
130_value
132_value
134_value
```