Вивід до завдання 3):
На першому скріншоті видно, що у нас є 3 підключення і ноди розподілені практично порівну. Після видалення одної з нод, дві інші взяли її результати на себе.

Map Statistics (In-Memory Format: BINARY)

Now

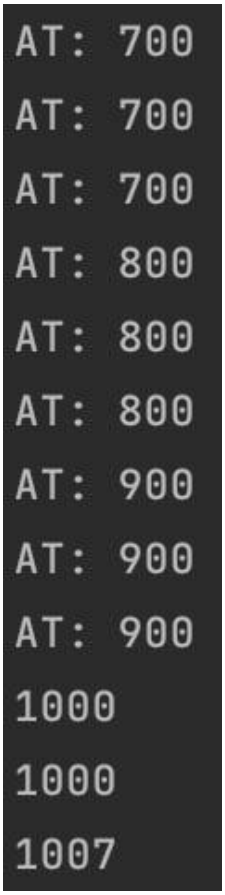
Member	Entries	Gets	Puts	Removals	Sets	Entry I
127.0.0.1:5701	320	0	320	0	0	3
127.0.0.1:5702	344	0	344	0	0	4
127.0.0.1:5703	336	0	336	0	0	4
TOTAL	1000	0	1000	0	0	12

Map Statistics (In-Memory Format: BINARY)

Now

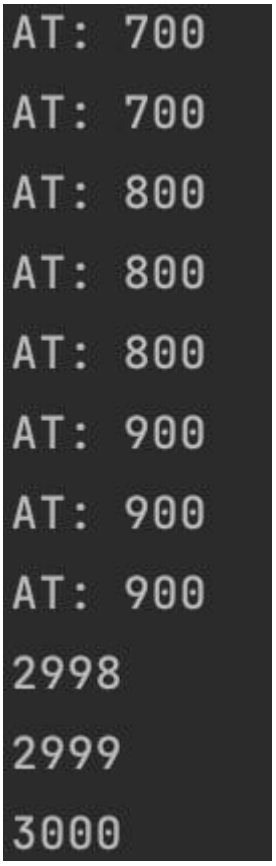
Member	Entries	Gets	Puts	Removals	Sets	Entry I
127.0.0.1:5702	504	0	344	0	0	6
127.0.0.1:5703	496	0	336	0	0	6
TOTAL	1000	0	680	0	0	12

Вивід до завдання 4):



Без блокування.

Оптимістичне блокування. У даному випадку сервери працювали у випадковому порядку. Видно велику різницю між часом, коли завершив перший сервер і третій.



Песимістичне блокування. У цьому випадку, ми бачимо, що дані поділилися практично порівну, кожен із серверів працював приблизно по черзі.

Вивід до завдання 5:

```
<multimap name="default">
    <backup-count>1</backup-count>
    <value-collection-type>SET</value-collection-type>
    <merge-policy batch-size="100">com.hazelcast.spi.merge.PutIfAbsentMergePolicy</merge-policy>
</multimap>

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

У випадку, якщо черга повністю заповнена і ми хочемо додати елемент, то сервер буде пробувати додати елемент, поки не звільниться місце.
У випадку, якщо черга пуста, то сервер буде проувати зчитати дані, поки не з’явиться новий елемент.

Якщо зчитують паралельно декілька клієнтів, то порядок визначити неможливо, проте дані не будуть повторюватися.