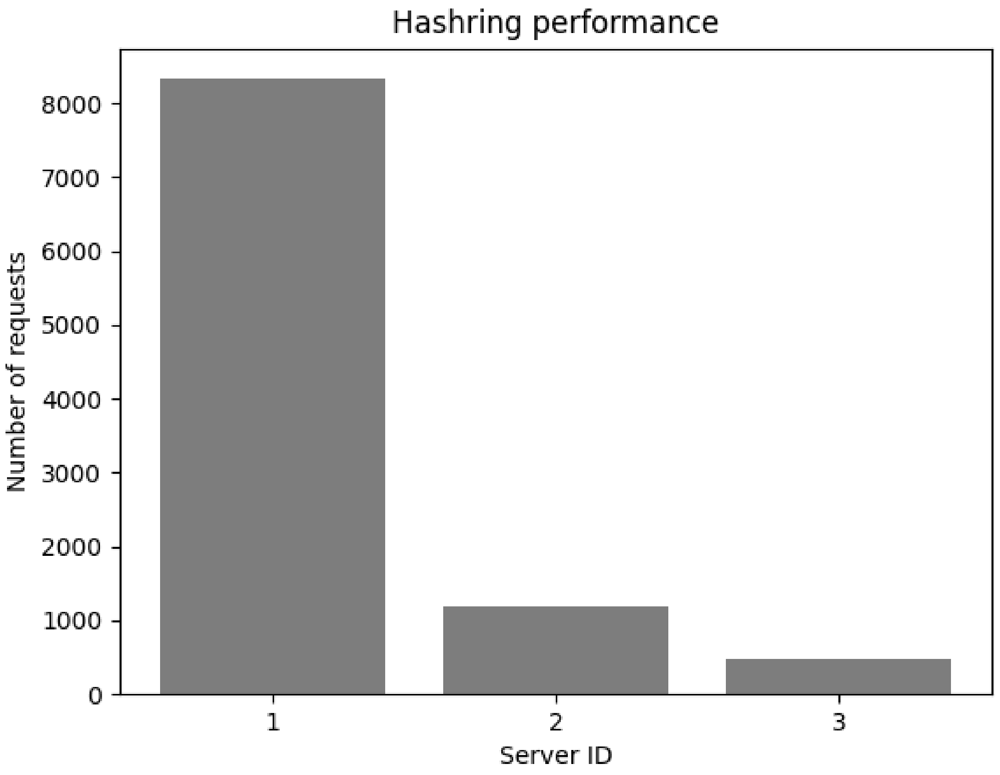
Analysis

A-1: Launch 10,000 Async Requests on N = 3 Server Containers

In this test, we launched 10,000 asynchronous requests on a load balancer with 3 server containers. The bar chart below illustrates the request count handled by each server instance.

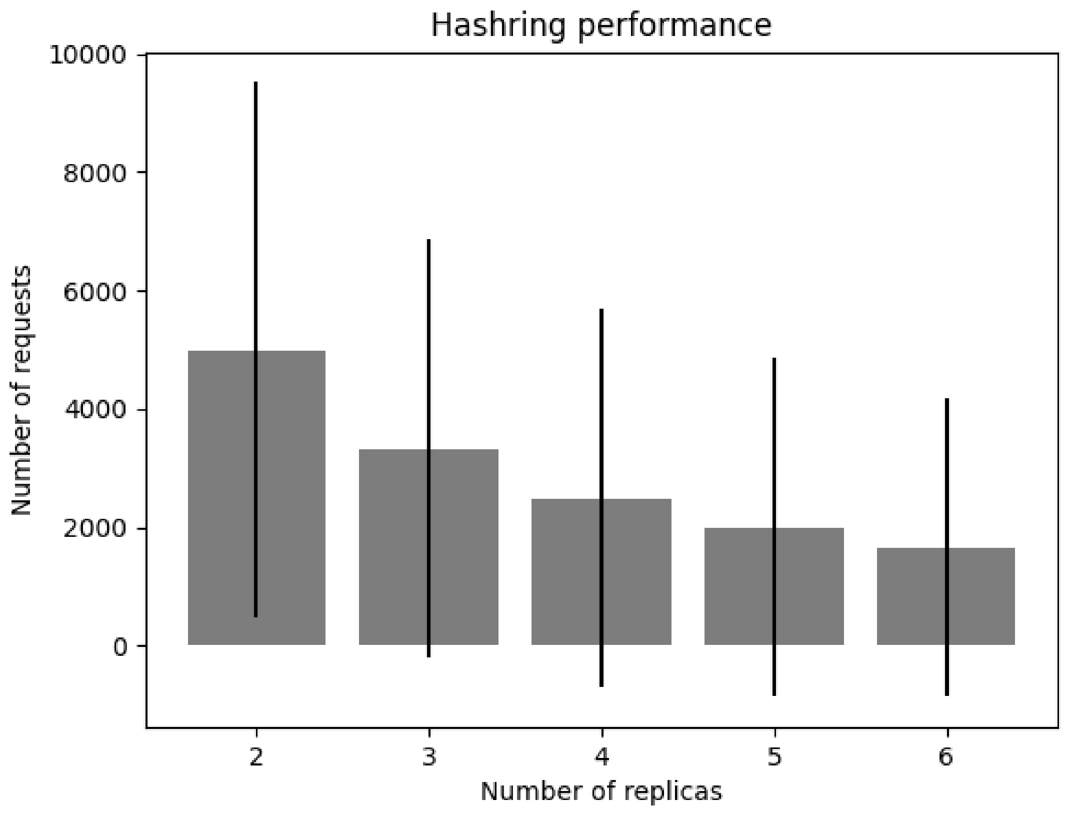


Time taken: 18.75167202949524 seconds
Throughput: 533.2857776240224 requests/second

Observations:
The load distribution is very uneven, server 1 gets most of the requests. This is because the hash function is not evenly spread out in the range [0, 512]

A-2: Increment N from 2 to 6 and Launch 10,000 Requests on Each Increment

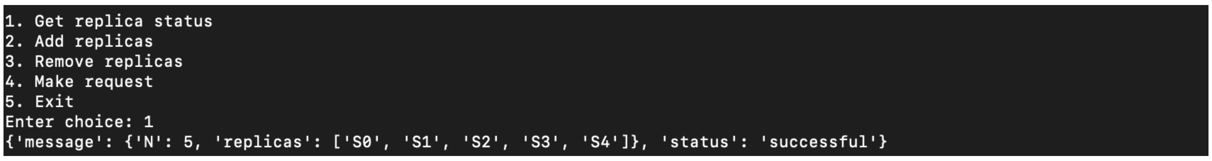
We increased the number of server containers from 2 to 6 and launched 10,000 requests on each increment. The line chart below shows the average load on the servers at each run.



Observations:
As the number of servers increased, the average load per server decreased.
The load balancer did not balance the load effectively, as the standard deviations of the number of requests is very high

A3: Test All Endpoints of the Load Balancer

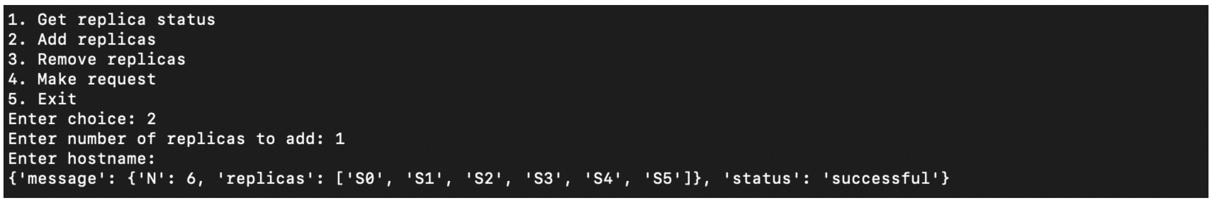
/rep Endpoint (Method: GET)



```
1. Get replica status
2. Add replicas
3. Remove replicas
4. Make request
5. Exit
Enter choice: 1
{'message': {'N': 5, 'replicas': ['S0', 'S1', 'S2', 'S3', 'S4']}, 'status': 'successful'}
```

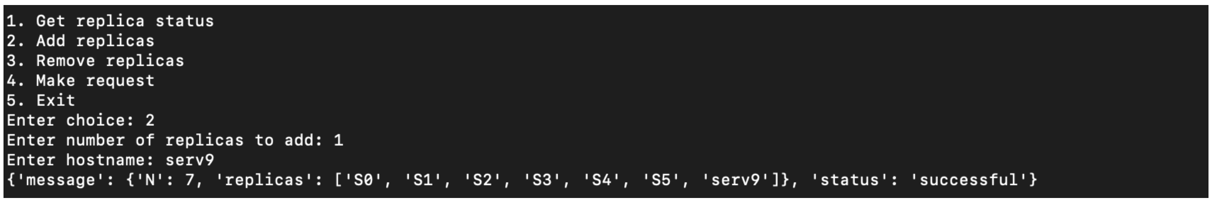
Results:
The response successfully provided information about the number of replicas and their hostnames.
The status of the operation was reported as "successful."

/add Endpoint (Method: POST)



```
1. Get replica status
2. Add replicas
3. Remove replicas
4. Make request
5. Exit
Enter choice: 2
Enter number of replicas to add: 1
Enter hostname:
{'message': {'N': 6, 'replicas': ['S0', 'S1', 'S2', 'S3', 'S4', 'S5']}, 'status': 'successful'}
```

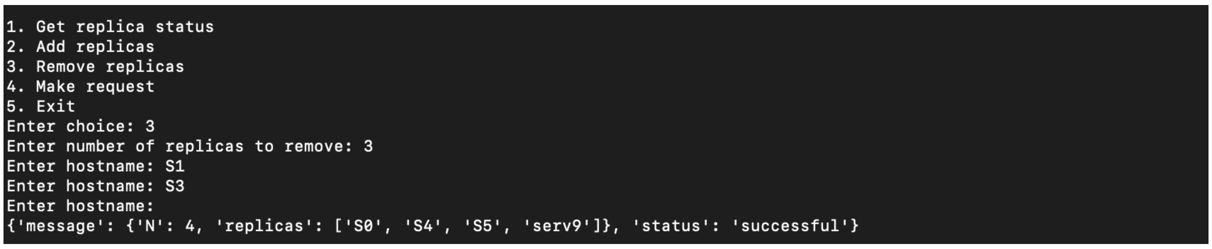
If we do not assign a name, then the load balancer assigns a unique name to the new server



```
1. Get replica status
2. Add replicas
3. Remove replicas
4. Make request
5. Exit
Enter choice: 2
Enter number of replicas to add: 1
Enter hostname: serv9
{'message': {'N': 7, 'replicas': ['S0', 'S1', 'S2', 'S3', 'S4', 'S5', 'serv9']}, 'status': 'successful'}
```

Results:
The response successfully confirmed the addition of new server instances.
The status of the operation was reported as "successful."

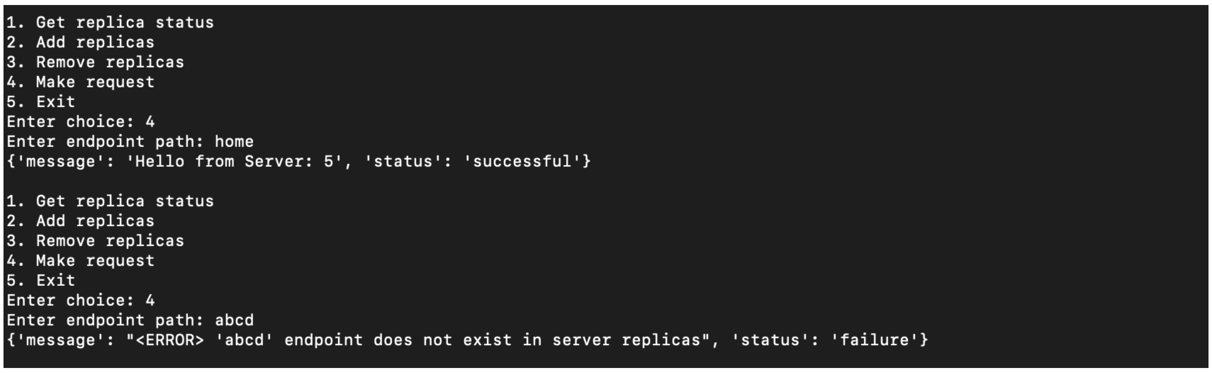
/rm Endpoint (Method: DELETE)



```
1. Get replica status
2. Add replicas
3. Remove replicas
4. Make request
5. Exit
Enter choice: 3
Enter number of replicas to remove: 3
Enter hostname: S1
Enter hostname: S3
Enter hostname:
{'message': {'N': 4, 'replicas': ['S0', 'S4', 'S5', 'serv9']}, 'status': 'successful'}
```

Results:
The response confirmed the removal of server instances from the load balancer.
The status of the operation was reported as "successful."

/<path> Endpoint (Method: GET)



```
1. Get replica status
2. Add replicas
3. Remove replicas
4. Make request
5. Exit
Enter choice: 4
Enter endpoint path: home
{'message': 'Hello from Server: 5', 'status': 'successful'}

1. Get replica status
2. Add replicas
3. Remove replicas
4. Make request
5. Exit
Enter choice: 4
Enter endpoint path: abcd
{'message': "<ERROR> 'abcd' endpoint does not exist in server replicas", 'status': 'failure'}
```

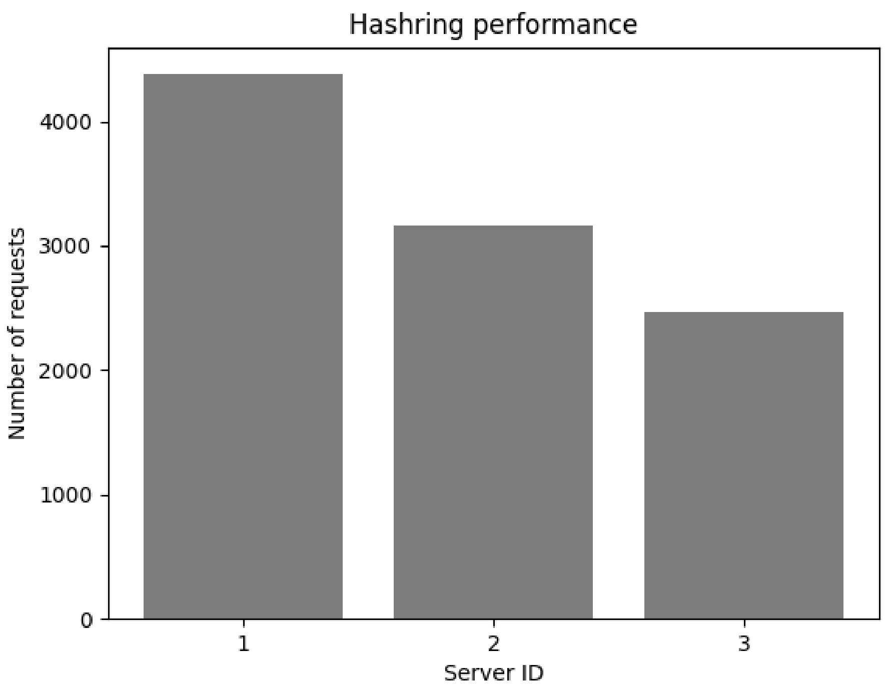
Results:
For defined endpoints, the response provided the expected message and reported a "successful" status.
For undefined endpoints, the response correctly indicated that the requested endpoint does not exist in server replicas.

Server Failure

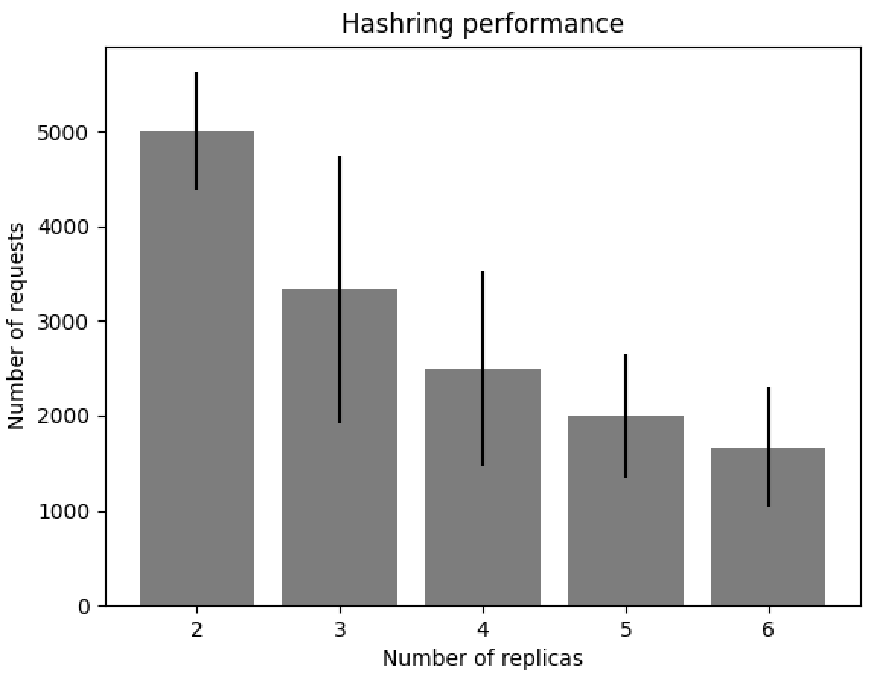
The load balancer keeps checking the servers every 10 seconds. In case of a server failure, the load balancer spawns a new server within 10 seconds.

A4: Modify Hash Functions and Report Observations

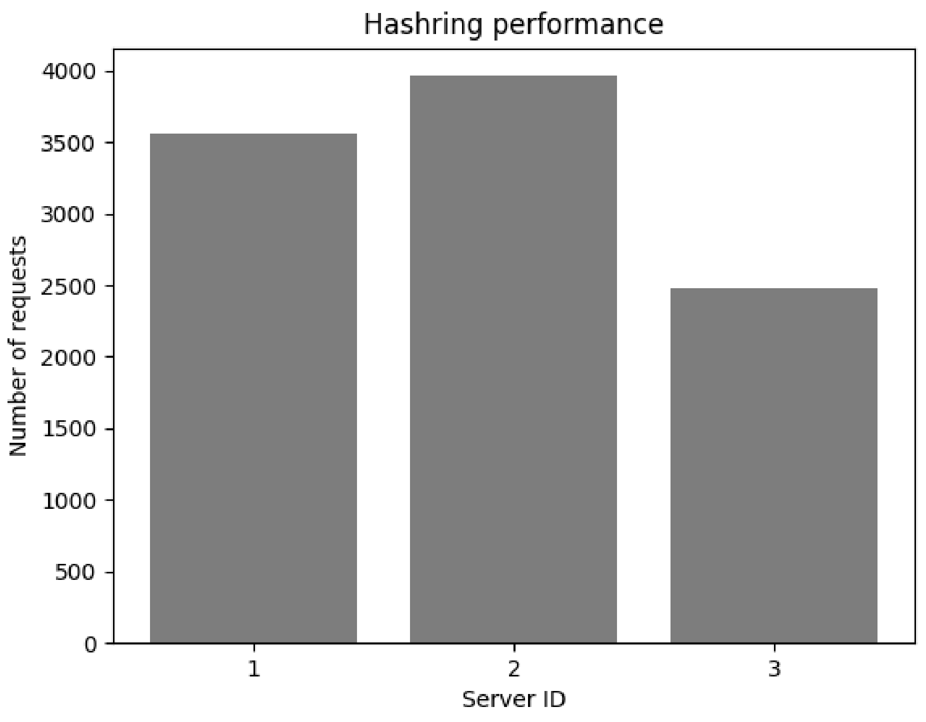
SHA256:



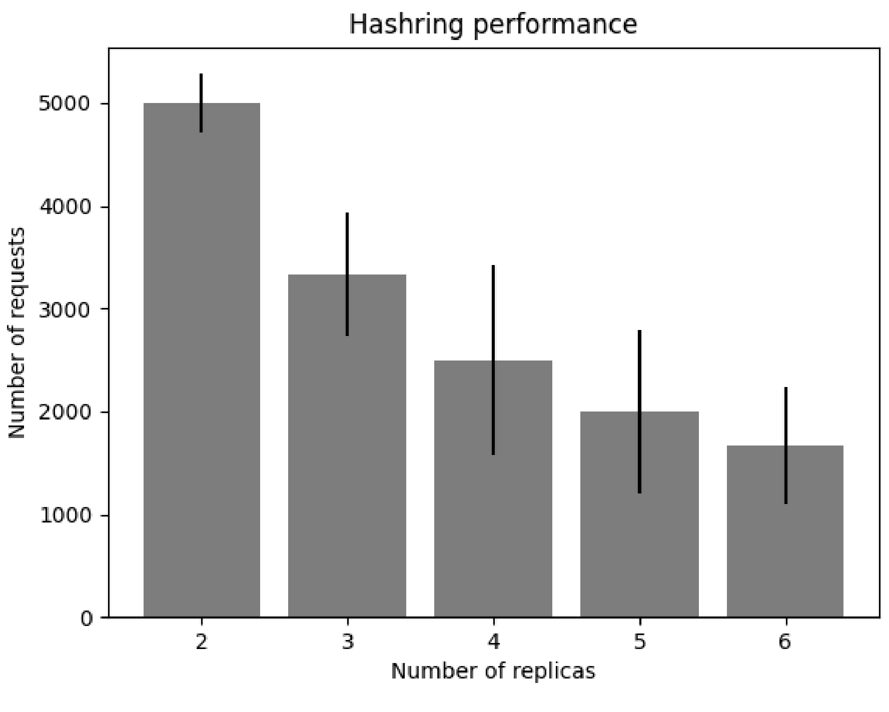
Time Taken - 21.67357087135315 seconds
Throughput: 461.39143657298354 requests/second



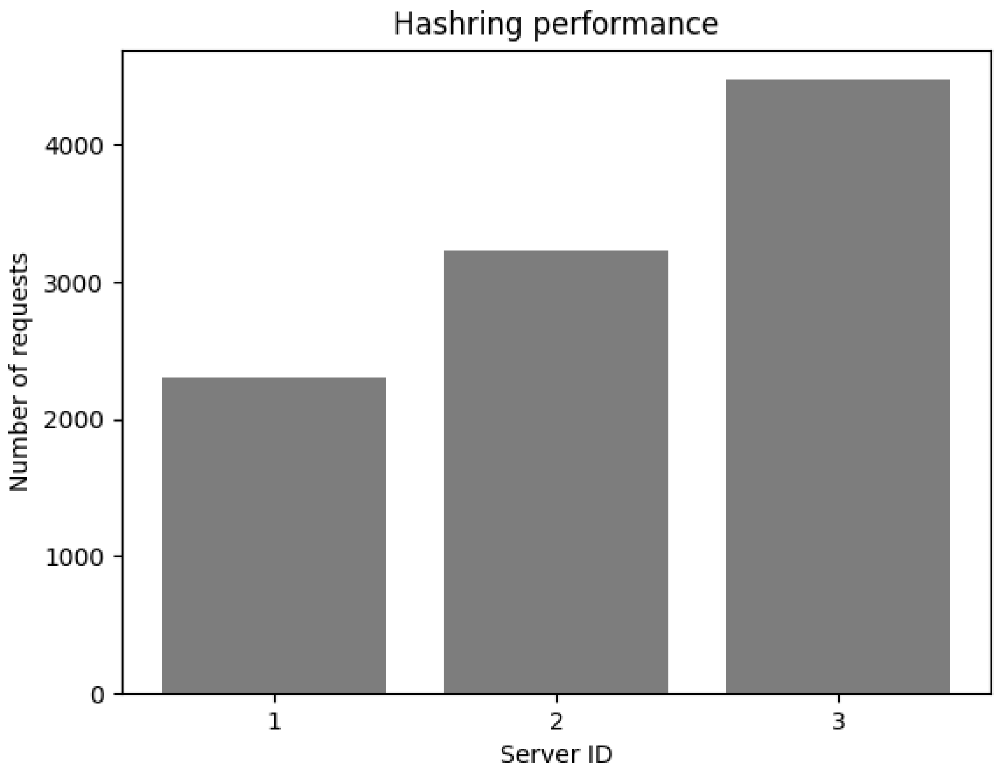
MD5:



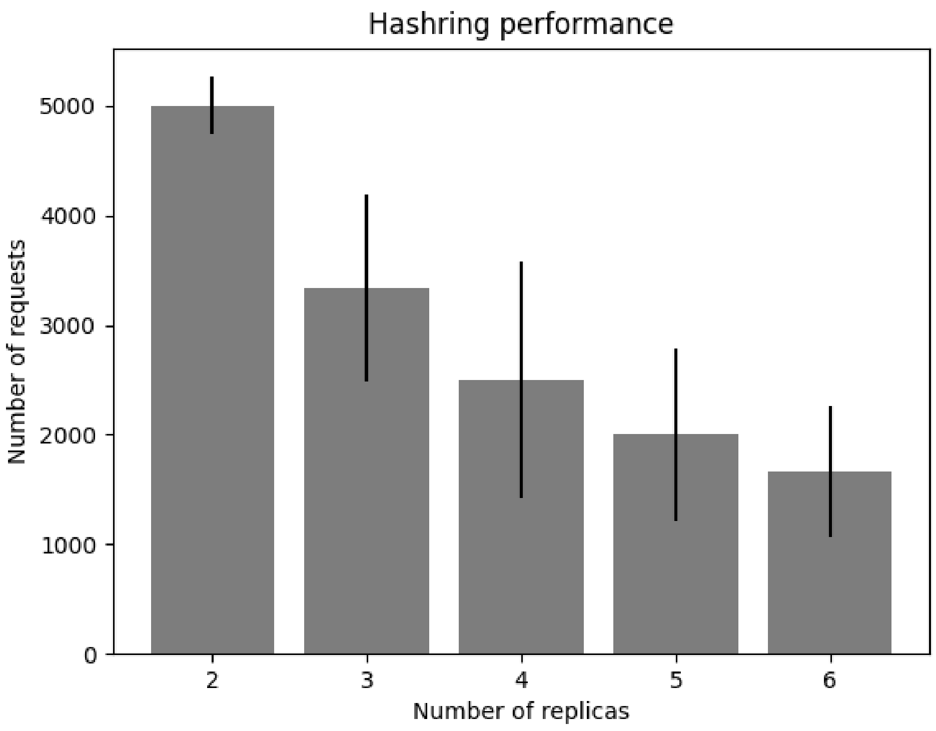
Time taken: 19.958230018615723 seconds
Throughput: 501.0464350131579 requests/second



Custom ($i^2 * j^2 + 2j + 25$)



Time taken: 19.058347940444946 seconds
Throughput: 524.7044513642421 requests/second



Observations:

- Hash functions like MD5, SHA256 and a custom hash function which is more spread out performs much better than the given hash function which allocates most servers in lower values