Q1 Problem Motivation

Currently, Serverless architecture is rapidly emerging as a popular technology in the cloud computing domain. It enables developers to build and run applications without the need to manage underlying infrastructure. This means that developers can concentrate on the development of business and algorithm logic while being liberated from tedious operational tasks. This architecture enhances development speed, elasticity, and scalability while reducing costs. Mainstream cloud service providers, such as Azure, Alibaba Cloud, and AWS, have launched their own Serverless services, with Azure Function being a prime example. Furthermore, active open-source communities have developed open-source Serverless platforms such as OpenFaaS.

At present, the performance research of Serverless platforms primarily focuses on several aspects, including startup time, execution time, memory usage, and concurrent requests. These metrics are crucial for the performance of Serverless functions. The objective of this experiment is to compare the performance of the Azure Function with the open-source platform OpenFaaS. However, due to time constraints, it is challenging to conduct a comprehensive comparison in all aspects. Therefore, this experiment will compare the performance of the two platforms in the following areas:

- Matrix multiplication
- HTTP requests
- Kafka consumption speed

Q2 Related Work

Currently, performance testing for serverless platforms is mainly focused on the comparison between open-source and commercial platforms. Among them, Li et al. [1] provided me with a lot of inspiration through their performance testing comparison of open-source serverless platforms, with a focus on http throughput performance testing. In the experimental design, I also referred to some of their methods.

Li, J. et al. (2021) "Analyzing Open-Source Serverless Platforms: Characteristics and Performance (S)," Proceedings [Preprint]. Available at: https://doi.org/10.18293/seke2021-129.

Q3 Experimental Design and Implementation

Q3.1 Experimental Design

I evaluate the commercial serverless platforms on the Azure Functions and the OpenFaas. The OpenFaas was deployed on the Azure VM(Standard B2ms), with 2 vcpu， 8 GiB Memory, running ubuntu 18.04. And the Azure Function is set to use the Consumption (Serverless) Plan.

To evaluate the HTTP performance, I implement a HTTP work-load function to send a 5-Bytes Data. I also use wrk to generate HTTP workloads for invoking serverless functions.

Simultaneously, in order to measure the computational performance of the platform, I decided to use matrix multiplication as a test for its computational capabilities. I created a function that takes the matrix size for matrix multiplication and the desired number of tests as input, and outputs the average time required for each multiplication calculation. This serves as a measure of the platform's computational performance.

In practice, Serverless Functions are often employed as Kafka consumers triggered by Kafka messages, consuming Kafka data. Consequently, to better align with real-world scenarios, I decided to test the consumption speed of Kafka consumers. This metric reflects the comprehensive performance of throughput, computation, and I/O. Thus, I created a function that takes the number of Kafka messages to be consumed as input and returns the time consumed. Additionally, I used Python to develop a Kafka producer, which writes a large number of messages to

Kafka in preparation for testing.

Q3.2 Implementation

I implemented all the functions using the Java language.

For HTTP performance testing, I implemented an HTTP work-load function that sends 5 bytes of data. I wrote a Lua script and used `wrk` to perform stress testing on the interface. The command is as follows:

```
./wrk -c 500 -t 20 -d 60 -s test.lua https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
```

The testing method referred to Li et al., involving a 60-second warm-up followed by a 60-second test. The test was repeated five times, obtaining the throughput (RPS) values, which served as the performance test results.

For matrix multiplication, I used PostMan to call the `HttpTrigger-average-matrix-multiplication-test` interface. This interface accepts two parameters: `generation` and `size`, which represent the number of test runs and the matrix size for multiplication, respectively. The interface returns the average time required for matrix multiplication calculations after multiple tests. In this experiment, I set it to perform 20 tests, each time multiplying 400×400 matrices. The matrices for calculation were randomly generated, and the matrix calculations utilized Apache Commons Math3.

For testing Kafka consumption speed, for convenience, I still created an interface: `kafka-consume-test`. The input parameter `totalMessage` represents the number of Kafka messages planned for consumption. The return value is the time required to consume the specified number of messages. The code depends on the Kafka-client library. First, I wrote 250,000 messages to Kafka using a Kafka producer. Then, I called the Kafka consumers deployed on OpenFaaS and Azure Function to consume 100,000 Kafka messages each and returned the required time.

Result

[HTTP TEST]

Openfaas

```
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   537.93ms   94.01ms 844.09ms   73.03%
    Req/Sec    49.35     35.79   242.00     70.90%
  55300 requests in 1.00m, 34.78MB read
  Non-2xx or 3xx responses: 55300
Requests/sec:    920.19
Transfer/sec:    592.65KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://openfaas.halden.top/function/http-test
Running 1m test @ https://openfaas.halden.top/function/http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   468.98ms   75.19ms 994.37ms   66.76%
    Req/Sec    54.63     33.86   242.00     63.31%
  63463 requests in 1.00m, 39.91MB read
  Non-2xx or 3xx responses: 63463
Requests/sec:   1056.01
Transfer/sec:    680.11KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://openfaas.halden.top/function/http-test
Running 1m test @ https://openfaas.halden.top/function/http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   509.57ms  118.83ms   1.21s    83.92%
    Req/Sec    49.59     25.52   212.00     69.37%
  58530 requests in 1.00m, 36.81MB read
  Non-2xx or 3xx responses: 58530
Requests/sec:    973.89
Transfer/sec:    627.24KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://openfaas.halden.top/function/http-test
Running 1m test @ https://openfaas.halden.top/function/http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   466.72ms   77.21ms 960.84ms   69.58%
    Req/Sec    54.11     28.74   242.00     72.70%
  63865 requests in 1.00m, 40.17MB read
  Non-2xx or 3xx responses: 63865
Requests/sec:   1062.68
Transfer/sec:    684.40KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://openfaas.halden.top/function/http-test
Running 1m test @ https://openfaas.halden.top/function/http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   489.33ms   74.74ms   1.62s    73.38%
    Req/Sec    54.50     38.59   284.00     73.28%
  60935 requests in 1.00m, 38.33MB read
  Non-2xx or 3xx responses: 60935
Requests/sec:   1013.94
Transfer/sec:    653.03KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://openfaas.halden.top/function/http-test
Running 1m test @ https://openfaas.halden.top/function/http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   498.15ms   89.56ms   1.58s    72.99%
    Req/Sec    51.50     31.92   242.00     76.30%
  59961 requests in 1.00m, 37.71MB read
  Non-2xx or 3xx responses: 59961
Requests/sec:    997.99
Transfer/sec:    642.73KB
```

Azure Functions

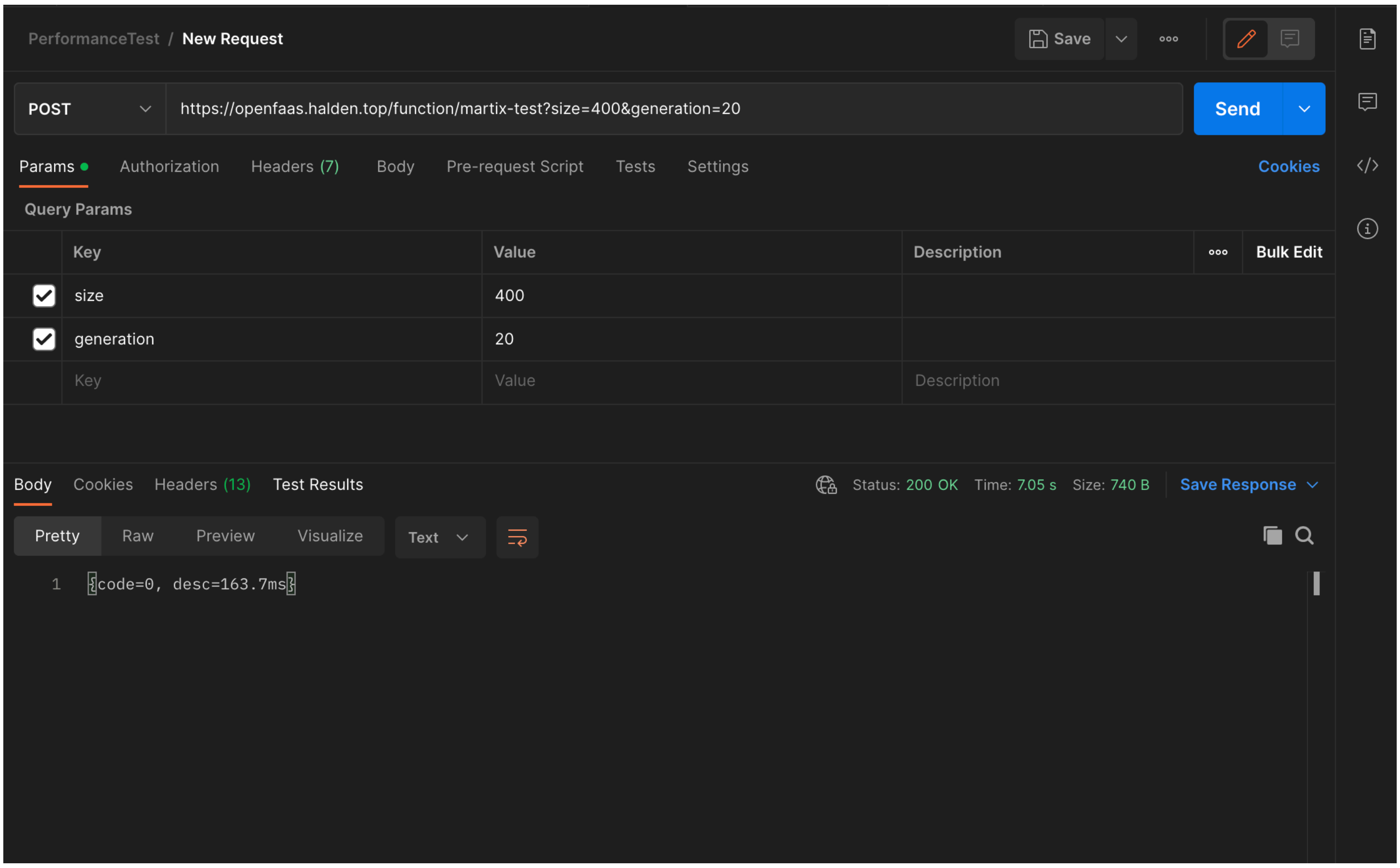
```
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
Running 1m test @ https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   240.34ms  242.89ms   1.99s    88.75%
    Req/Sec   129.08     41.56   450.00     69.30%
  151961 requests in 1.00m, 15.53MB read
  Socket errors: connect 0, read 0, write 0, timeout 59
  Non-2xx or 3xx responses: 151961
Requests/sec:   2528.64
Transfer/sec:    264.54KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
Running 1m test @ https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   185.63ms  199.68ms   1.53s    86.76%
    Req/Sec   179.86     55.13   540.00     69.77%
  212159 requests in 1.00m, 20.04MB read
  Non-2xx or 3xx responses: 212159
Requests/sec:   3530.50
Transfer/sec:    341.45KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
Running 1m test @ https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   211.24ms  255.47ms   1.96s    85.14%
    Req/Sec   192.19     67.42   790.00     67.61%
  226099 requests in 1.00m, 21.35MB read
  Non-2xx or 3xx responses: 226099
Requests/sec:   3762.45
Transfer/sec:    363.75KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
Running 1m test @ https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   195.35ms  240.33ms   2.00s    84.45%
    Req/Sec   206.95     81.84   565.00     67.15%
  243545 requests in 1.00m, 22.99MB read
  Socket errors: connect 0, read 0, write 0, timeout 221
  Non-2xx or 3xx responses: 243545
Requests/sec:   4052.31
Transfer/sec:    391.78KB
root@yfliu:/home/jasondennis12138/wrk# ./wrk -c 500 -t 20 -d 60 -s test.lua https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
Running 1m test @ https://performance-test-01.azurewebsites.net/api/HttpTrigger-http-test
  20 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   150.62ms  165.25ms   1.56s    87.10%
    Req/Sec   219.91     56.73   700.00     68.01%
  258701 requests in 1.00m, 24.42MB read
  Non-2xx or 3xx responses: 258701
Requests/sec:   4304.62
Transfer/sec:    416.17KB
```

Results

	1st	2ed	3th	4th	5th	Average
Azure Function Throughput (RPS)	2528.64	3530.50	3762.45	4052.31	4304.62	3635.704
OpenFaas Throughput (RPS)	1056.01	973.89	1062.68	1013.94	997.99	1020.902

Matrix multiplication

OpenFaas



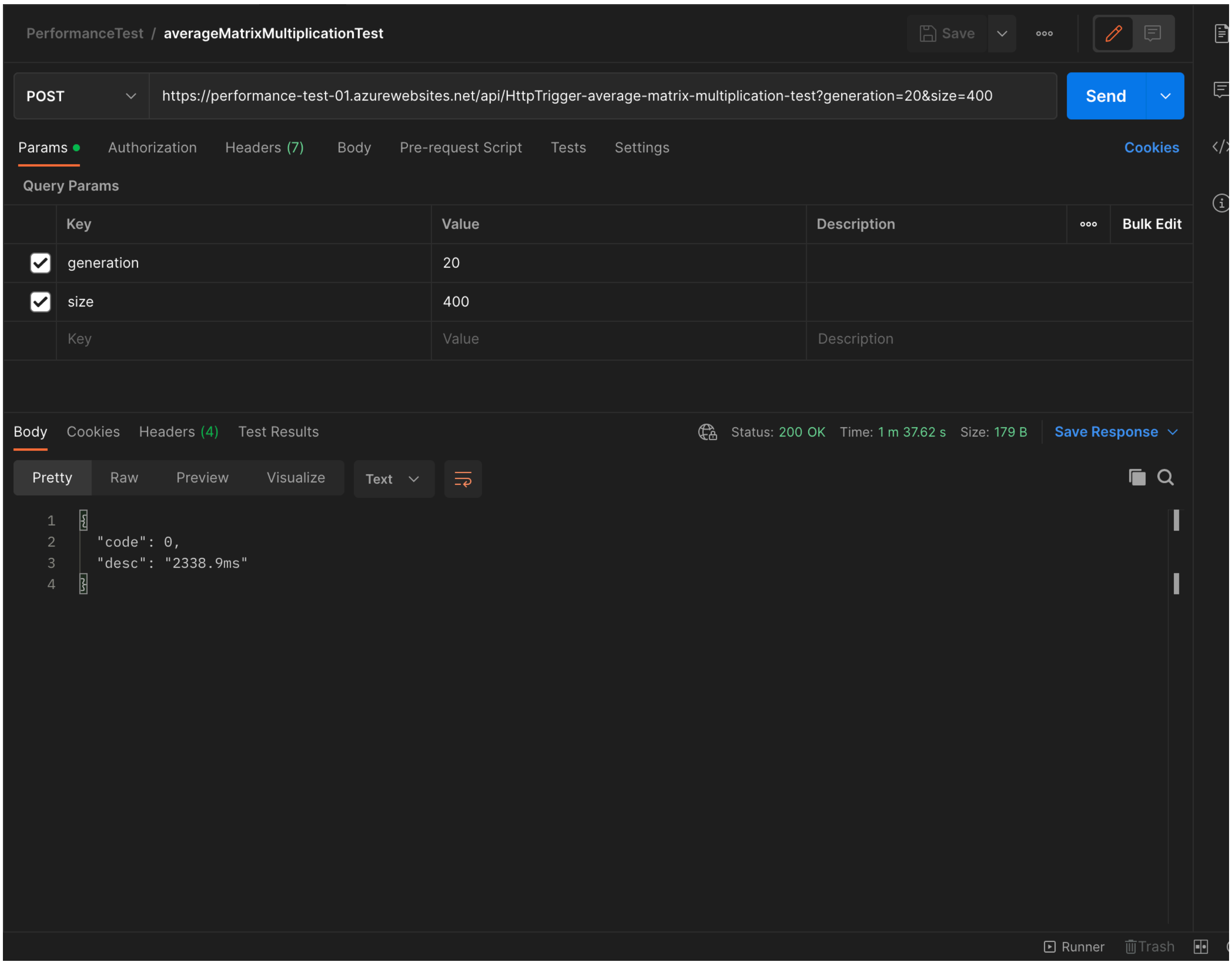
```
https://openfaas.halden.top/function/martix-test?size=400&generation=20
```

	Key	Value	Description
✓	size	400	
✓	generation	20	

Status: 200 OK Time: 7.05 s Size: 740 B

```
{code=0, desc=163.7ms}
```

Azure Function



POST https://performance-test-01.azurewebsites.net/api/HttpTrigger-average-matrix-multiplication-test?generation=20&size=400

Query Params

Key	Value	Description
generation	20	
size	400	

Status: 200 OK Time: 1 m 37.62 s Size: 179 B

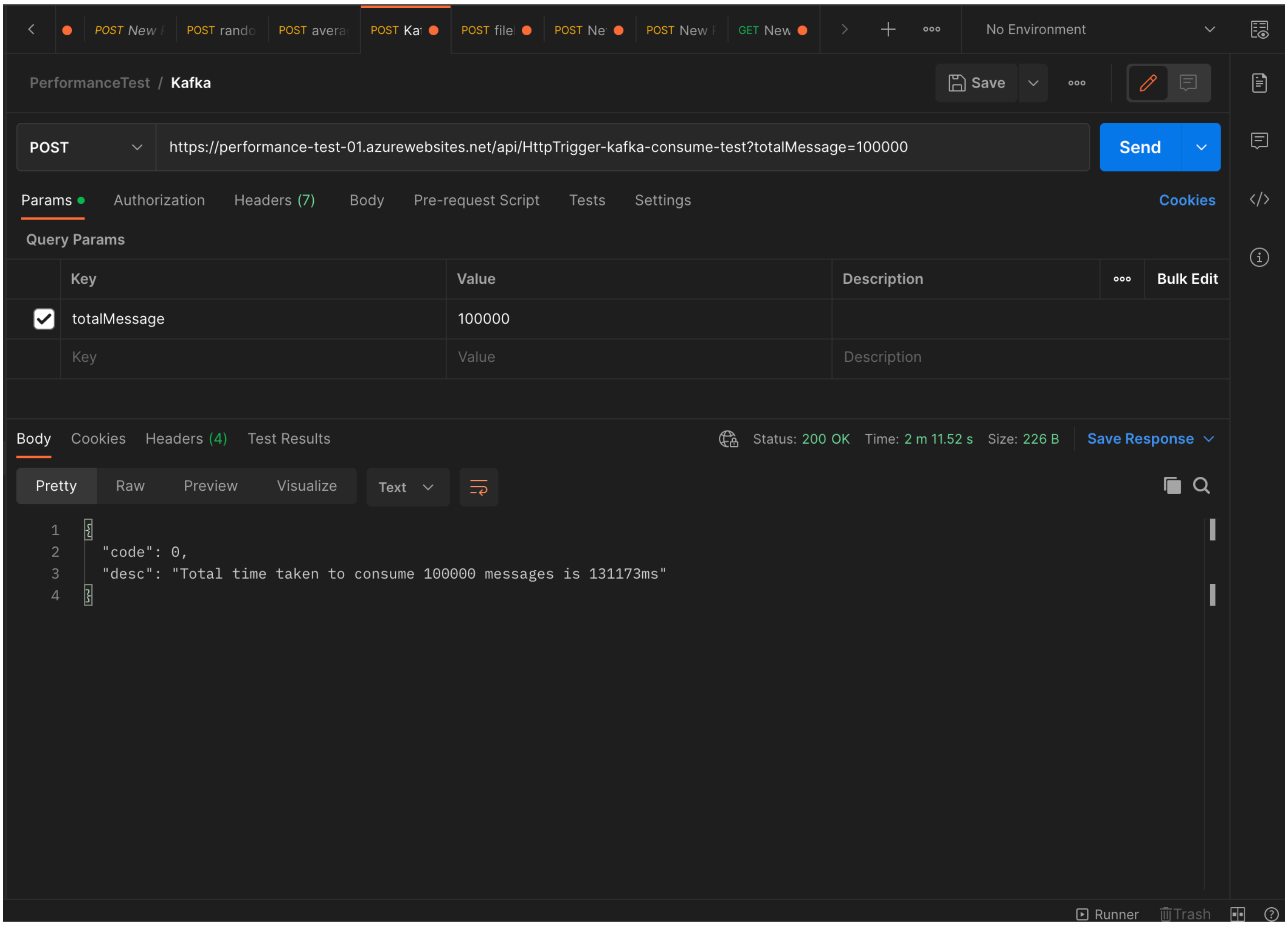
```
{
  "code": 0,
  "desc": "2338.9ms"
}
```

Result

	Size	Generation	Time Used(ms)
OpenFaas	400	20	163.7
Azure Function	400	20	2338.9

Kafka consume Test

Azure Function



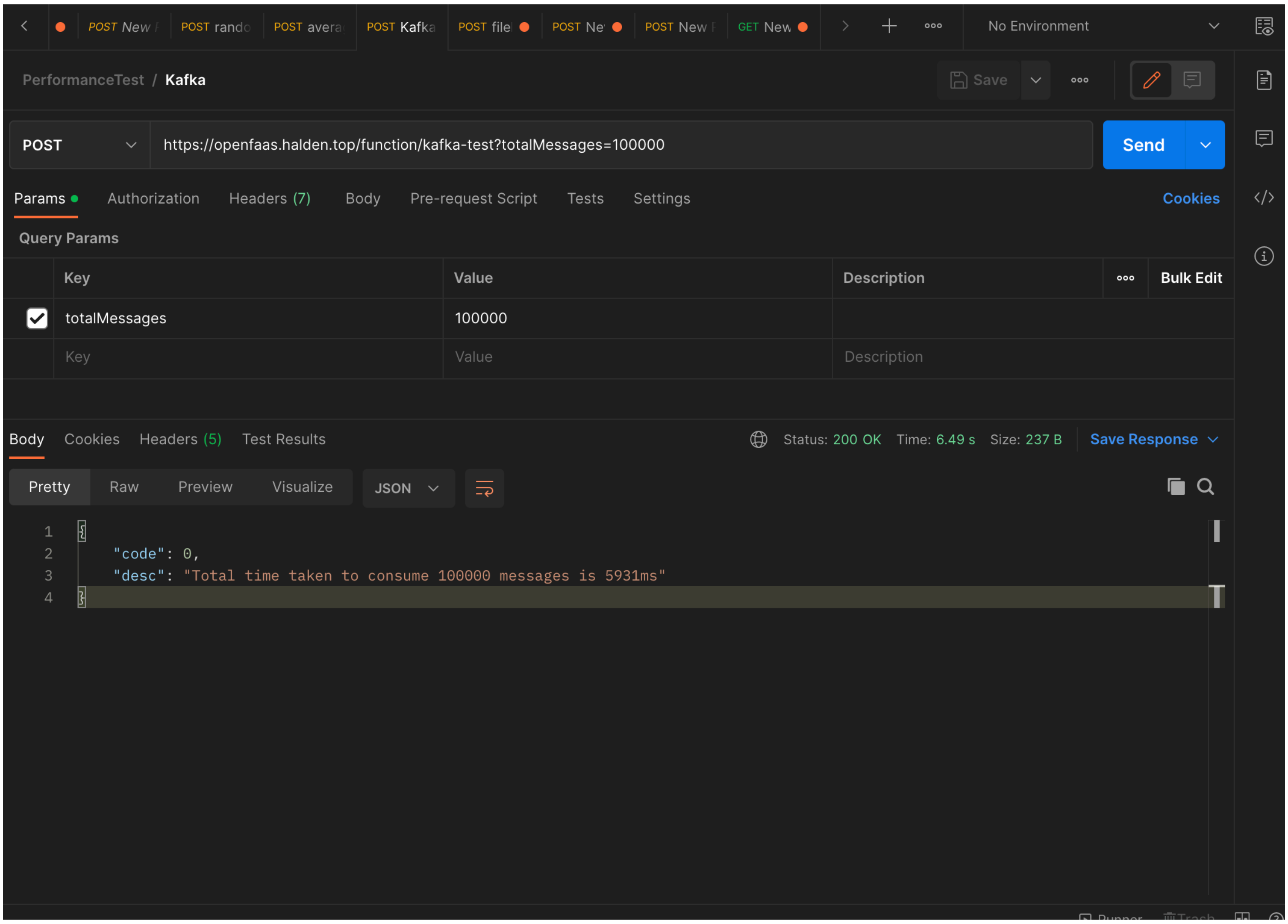
POST https://performance-test-01.azurewebsites.net/api/HttpTrigger-kafka-consume-test?totalMessage=100000

Key	Value	Description
✓ totalMessage	100000	

Status: 200 OK Time: 2 m 11.52 s Size: 226 B

```
{
  "code": 0,
  "desc": "Total time taken to consume 100000 messages is 131173ms"
}
```

OpenFaas



POST https://openfaas.halden.top/function/kafka-test?totalMessages=100000

Query Params

Key	Value	Description
totalMessages	100000	

Status: 200 OK Time: 6.49 s Size: 237 B

```
{
    "code": 0,
    "desc": "Total time taken to consume 100000 messages is 5931ms"
}
```

Result

	Message Consumed	Time Used(ms)
OpenFaas	100000	131173
Azure Function	100000	5931

Evaluation

For the HTTP performance test, we observed that the function deployed on OpenFaaS had a relatively stable throughput of around 1000 RPS. In contrast, the function deployed on Azure Function exhibited a significantly higher throughput compared to OpenFaaS. Additionally, we noticed that the achievable throughput steadily increased with continuous invocations. We believe that this is due to the auto-scaling capabilities of Azure Function. When experiencing continuous performance pressure, Azure Function scales the function to enhance its concurrency performance, achieving better concurrency performance. In this regard, Azure Function outperforms the open-source platform OpenFaaS in terms of concurrent performance.

In the matrix multiplication performance test, OpenFaaS demonstrated outstanding performance. The time required for a 400x400 matrix multiplication was merely 7% of that needed for Azure Function. This considerable performance gap indicates that the strength of Azure Function does not lie in computation-intensive tasks.

Similarly, for the Kafka consumption performance test, we found that Azure Function's performance was far inferior to OpenFaaS. Consuming 100,000 messages took as long as 2 minutes. As we know that consuming Kafka messages poses a stringent test for computational performance, this result was not surprising.

In summary, Azure Function's performance advantages lie in its auto-scaling and high concurrency capabilities. Azure Function is a mature platform, backed by a server cluster and boasting a robust scaling mechanism. When faced with a large number of concurrent requests, Azure Function promptly scales the function. Although OpenFaaS can also auto-scale, it is often constrained by the resource limitations of the cluster where OpenFaaS is installed, making it more prone to reach performance ceilings.

Admittedly, due to time constraints, many tasks were not completed satisfactorily. For example, in the Kafka consumption performance test, Azure Function could have been set up as a Kafka-message-triggered function with auto-scaling and auto-shrinking based on the backlog of Kafka messages. If this work had been completed, it might have better reflected the performance of Azure Function in Kafka message consumption.