Sprint Project 3 : Flask, ML, Api

Name: José Alejandro León Andrade

Stress Test Report

Stress tests were performed with locust and different parameters and the following results were obtained:

TEST 1.0:

Instances : 1	Users : 10	SpawnRate : 1s

Results obtained in locust:

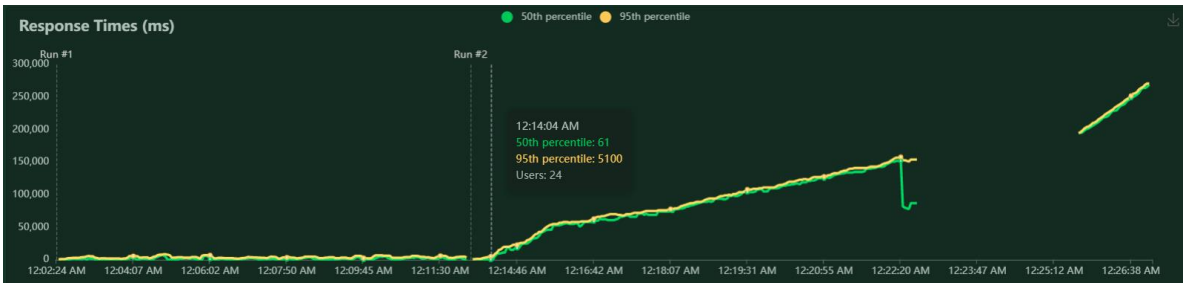
Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	621	0	29	51	240	37	16	471	548	1.4	0
POST	/predict	694	0	2300	4900	7100	2682	583	8429	85	1	0
	Aggregated	**1315**	**0**	**800**	**3900**	**6700**	**1433**	**16**	**8429**	**304**	**2.4**	**0**

TEST 2.0:

Instances : 1	Users : 500	SpawnRate : 1s

Results obtained in locust:

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	898	258	88000	206000	262000	94831	18	270479	391	0.9	0
POST	/predict	917	251	94000	159000	264000	99595	649	271200	62	1.1	0
	Aggregated	**1815**	**509**	**91000**	**201000**	**263000**	**97238**	**18**	**271200**	**224**	**2**	**0**



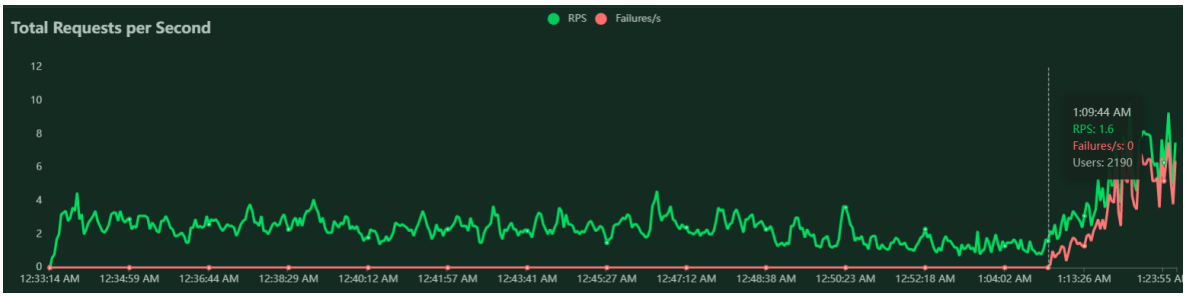
TEST 3:

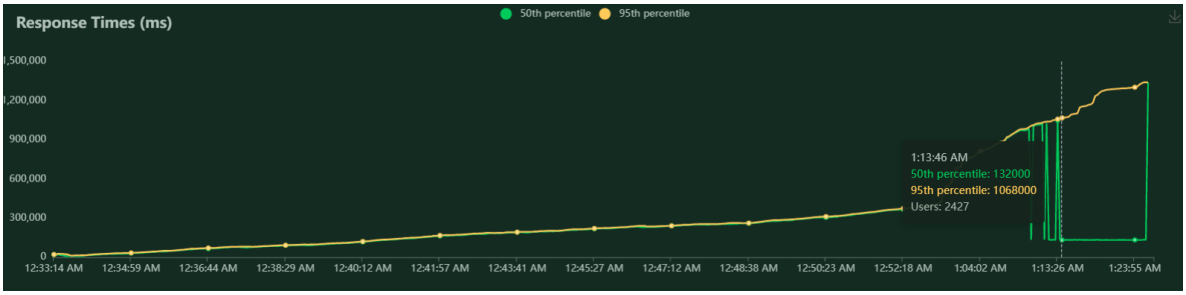
Instances : 1	Users : 3000	SpawnRate : 1s

Results obtained in locust:

Statistics | Charts | Failures | Exceptions | Current ratio | Download Data

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	4004	1377	145000	1079000	1268000	379126	5135	1289019	360	3.7	2.9
POST	/predict	4027	1425	145000	1058000	1274000	374787	10428	1294460	55	4.3	3.6
	Aggregated	**8031**	**2802**	**145000**	**1068000**	**1269000**	**376950**	**5135**	**1294460**	**207**	**8**	**6.5**





TEST, BY USING 2 INSTANCES:

Using the following command to add two instances of the model yields the following results:

docker-compose up --build -d --scale model=2

```
[+] Running 4/4
 - Container assignment-redis-1  Running
 - Container assignment-model-2  Started
 - Container assignment-model-1  Started
 - Container ml_api              Started
```

TEST 1.1:

Instances : 2	Users : 10	SpawnRate : 1s

Results obtained in locust:

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	648	0	35	120	380	70	19	4776	548	1.5	0
POST	/predict	625	8	1200	3900	31000	2446	555	52056	84	1.5	0
	Aggregated	**1273**	**8**	**280**	**2400**	**21000**	**1237**	**19**	**52056**	**320**	**3**	**0**

TEST 2.1:

Instances : 2	Users : 500	SpawnRate : 1s

Results obtained in locust:

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	975	0	40000	77000	91000	40227	20	94348	548	1.1	0
POST	/predict	948	0	43000	80000	95000	43509	614	96746	85	1.6	0
	Aggregated	**1923**	**0**	**41000**	**78000**	**94000**	**41845**	**20**	**96746**	**320**	**2.7**	**0**

TEST 3.1:

Instances : 2	Users : 3000	SpawnRate : 1s

Results obtained in locust:

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	19551	18708	30000	30000	134000	32918	19	170794	24	23.1	23.1
POST	/predict	19365	18461	30000	30000	141000	33425	865	176737	4	23	23
	Aggregated	**38916**	**37169**	**30000**	**30000**	**138000**	**33170**	**19**	**176737**	**14**	**46.1**	**46.1**

By running the command “docker exec containerID cat /proc/cpuinfo” it is possible to get the hardware specifications.

Hardware specifications:

Description	Values
model name	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
cpu MHz	3407.999
cache size	8192 KB
cpu cores	4
MemTotal	8114112 Kb
address sizes	39 bits physical, 48 bits virtual
Mem Limit	7.738GiB

When executing the following commands in windows, it gets the same values as those obtained in the docker containers.

- Get-WmiObject win32_processor | Format-List Name,NumberOfCores,NumberOfLogicalProcessors
- Get-WmiObject -Class win32_physicalmemory | Format-Table Capacity,Manufacturer,PartNumber,Speed -AutoSize
- Get-WmiObject win32_diskdrive | Format-Table Model,Size,InterfaceType

Analysis

It is observed that for 1 single instance and 10 users there is an average of 29ms per response for “get” while for a “post” an average of 2300 ms per response is obtained; in this case no failures were obtained.

On the other hand, when testing with 2 instances, we obtain responses in the “post” in almost half the time, however, we begin to notice failures.

Then tests were performed with 500 and 3000 users respectively. In this case for 1 instance, we obtained averages of 94000 ms and 145000 ms respectively. For these cases several failures are seen (27% for 500 concurrent users and 35% for 3000 users).

For the case of 2 instances, we obtain averages of 43000 ms and 19365 ms respectively. For these cases even more failures are seen for the case of 3000 concurrent users (almost 95%).

As a conclusion, it is observed that the more instances there are, the faster the servers will respond to requests, however, the percentage of failures is higher; this may be due to the fact that when load tests are performed with Locust, a large number of users performing different actions on the system are being simulated, which can lead to the system being overloaded and presenting failures or errors.

It must also be taken into account that the hardware resources supported by the servers are not designed for this type of loads or architectures.