Locust stress test

By fellow Alberto Rosales

Specs of my PC

MOTHERBOARD: GIGABYTE B500M AORUS ELITE

- Socket AM4
- Maximum RAM: 128GB
- RAM slots: 4
- Channel architecture: Dual-channel

CPU: RYZEN 5 5600X

- Cores: 6
- Threads: 12
- Frequency: 3.7 GHz up to 4.6 GHz
- Integrated Graphics: N/A
- Memory Support: DDR4-3200MHz Dual-channel

RAM: XPG GAMMIX D10

- Memory Type: DDR4
- Speed: 3200MHz (XMP 2.0)
- Capacity: 8GB x 4 (32GB Total)

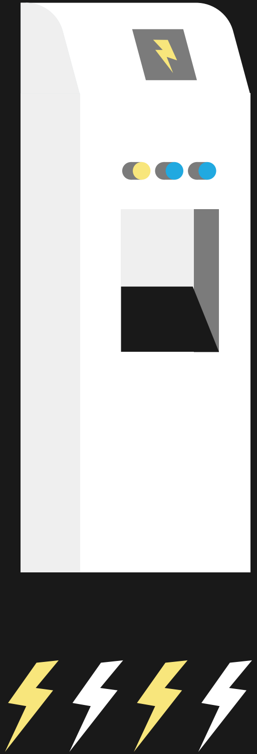
GPU: MSI VENTUS 2X - GeForce RTX 3060 Ti 8GB

- Interface: PCI Express Gen 4
- Core Clocks Boost: 1695 MHz
- Memory Speed: 14 Gbps

STORAGE: XPG SPECTRIX S40G

- Capacity: 1TB
- Factor: M.2
- Interface
- PCIe Gen3x4
- Sequential Read (Max): Up to 3500 MB/s
- Sequential Write (Max): Up to 3000 MB/s

Test execution plan

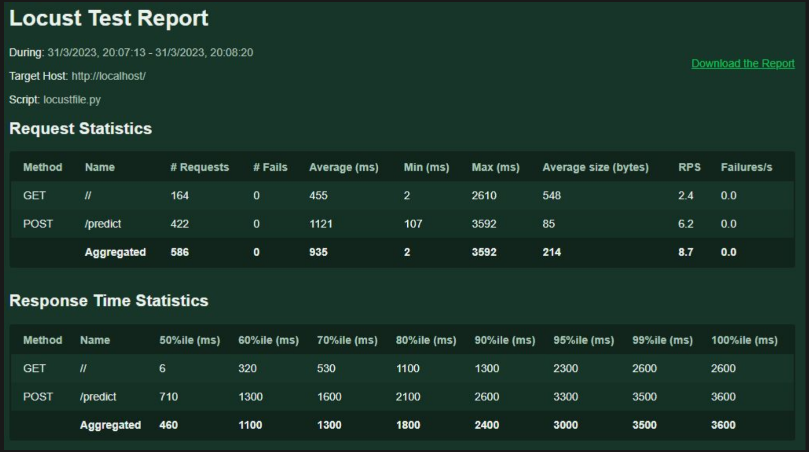


First with 1 instance of the model I'll be making iterations of the test, the limit will be 1000 users in each iteration, I'll be increasing the spawn rate of users randomly in every test until failure.

After that, I will repeat the same conditions, but this time with 2 instances of the model.

During this test I'll try to keep my RAM usage from other apps as low as possible, currently using 2 Chrome tabs, VS Code, and Docker.

1 spawn rate



Locust Test Report

During: 31/3/2023, 20:07:13 - 31/3/2023, 20:08:20

Target Host: http://localhost/

Script: locustfile.py

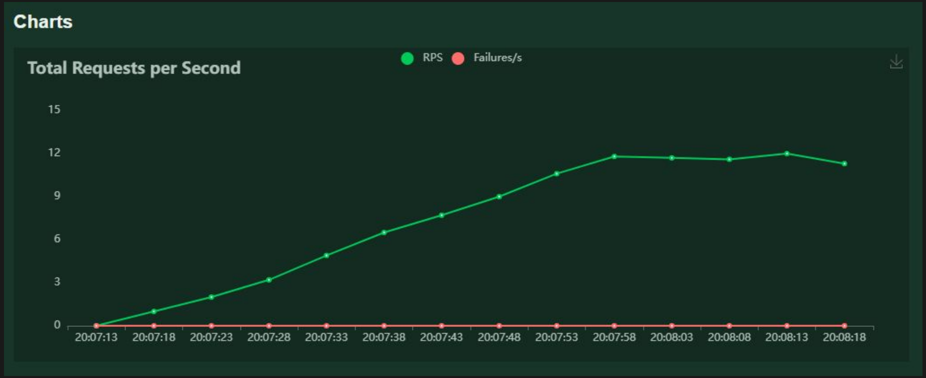
Download the Report

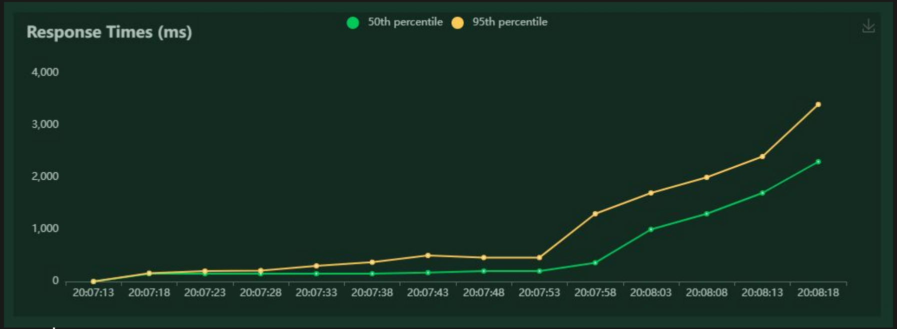
Request Statistics

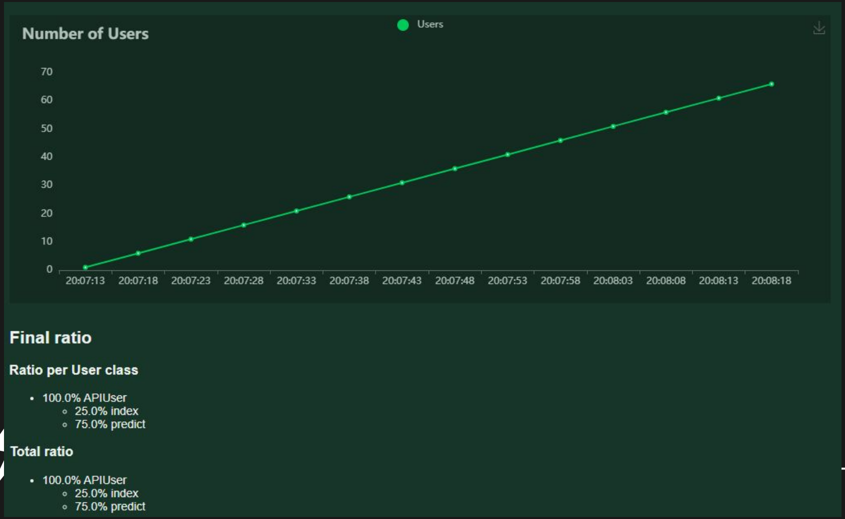
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	164	0	455	2	2610	548	2.4	0.0
POST	/predict	422	0	1121	107	3592	85	6.2	0.0
	Aggregated	**586**	**0**	**935**	**2**	**3592**	**214**	**8.7**	**0.0**

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	6	320	530	1100	1300	2300	2600	2600
POST	/predict	710	1300	1600	2100	2600	3300	3500	3600
	Aggregated	**460**	**1100**	**1300**	**1800**	**2400**	**3000**	**3500**	**3600**







Final ratio

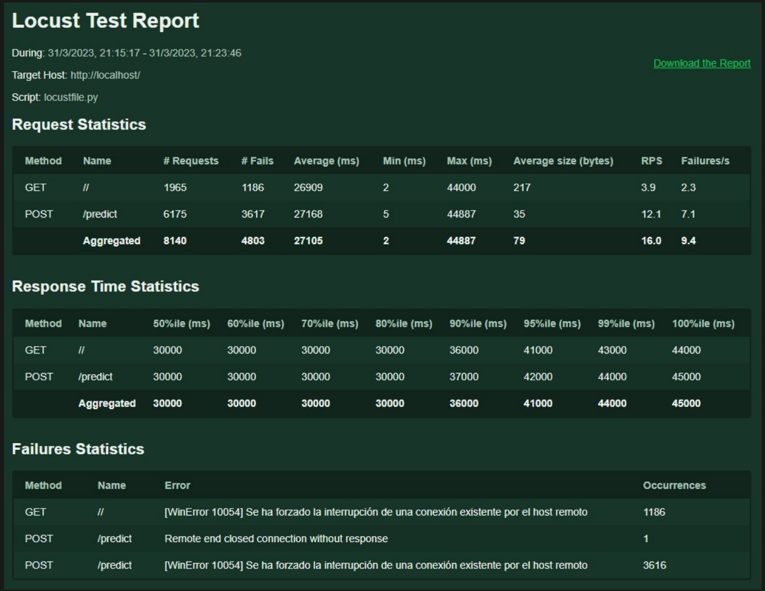
Ratio per User class

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Total ratio

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

2 spawn rate



Locust Test Report

During: 31/3/2023, 21:15:17 - 31/3/2023, 21:23:46

Target Host: http://localhost/

Script: locustfile.py

Download the Report

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	1965	1186	26909	2	44000	217	3.9	2.3
POST	/predict	6175	3617	27168	5	44887	35	12.1	7.1
	Aggregated	**8140**	**4803**	**27105**	**2**	**44887**	**79**	**16.0**	**9.4**

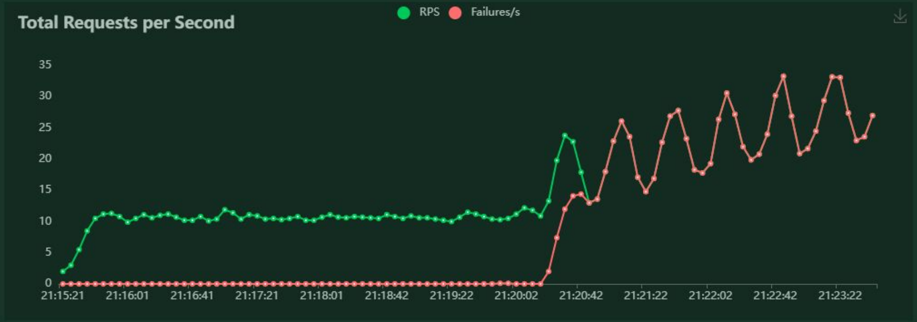
Response Time Statistics

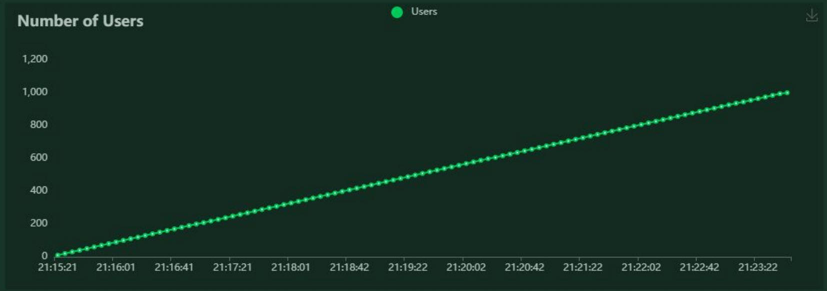
Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	30000	30000	30000	30000	36000	41000	43000	44000
POST	/predict	30000	30000	30000	30000	37000	42000	44000	45000
	Aggregated	**30000**	**30000**	**30000**	**30000**	**36000**	**41000**	**44000**	**45000**

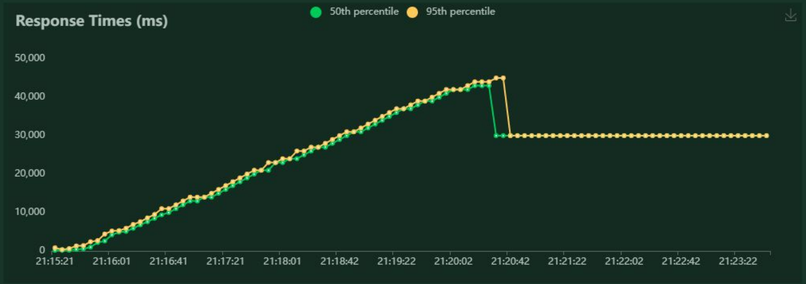
Failures Statistics

Method	Name	Error	Occurrences
GET	//	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	1186
POST	/predict	Remote end closed connection without response	1
POST	/predict	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	3616

Charts







Final ratio

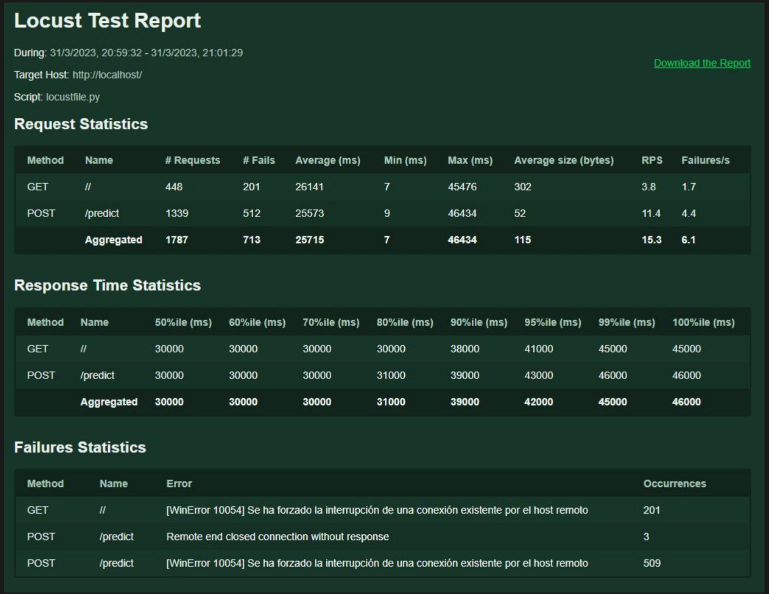
Ratio per User class

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Total ratio

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

10 spawn rate



Locust Test Report

During: 31/3/2023, 20:59:32 - 31/3/2023, 21:01:29

Target Host: http://localhost/

Script: locustfile.py

Download the Report

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	448	201	26141	7	45476	302	3.8	1.7
POST	/predict	1339	512	25573	9	46434	52	11.4	4.4
	Aggregated	**1787**	**713**	**25715**	**7**	**46434**	**115**	**15.3**	**6.1**

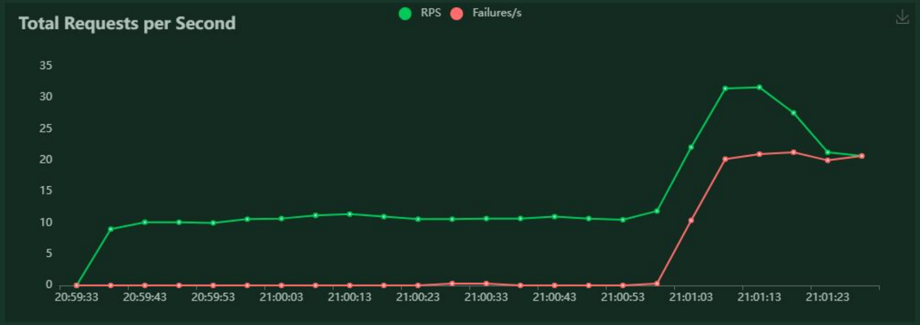
Response Time Statistics

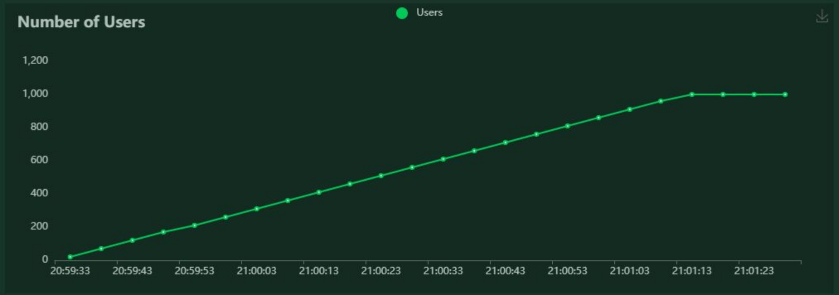
Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	30000	30000	30000	30000	38000	41000	45000	45000
POST	/predict	30000	30000	30000	31000	39000	43000	46000	46000
	Aggregated	**30000**	**30000**	**30000**	**31000**	**39000**	**42000**	**45000**	**46000**

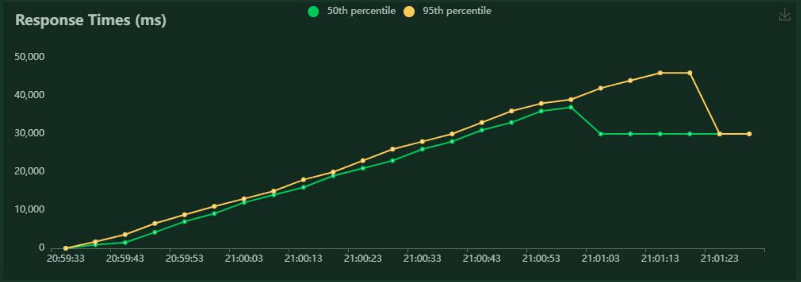
Failures Statistics

Method	Name	Error	Occurrences
GET	//	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	201
POST	/predict	Remote end closed connection without response	3
POST	/predict	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	509

Charts







Final ratio

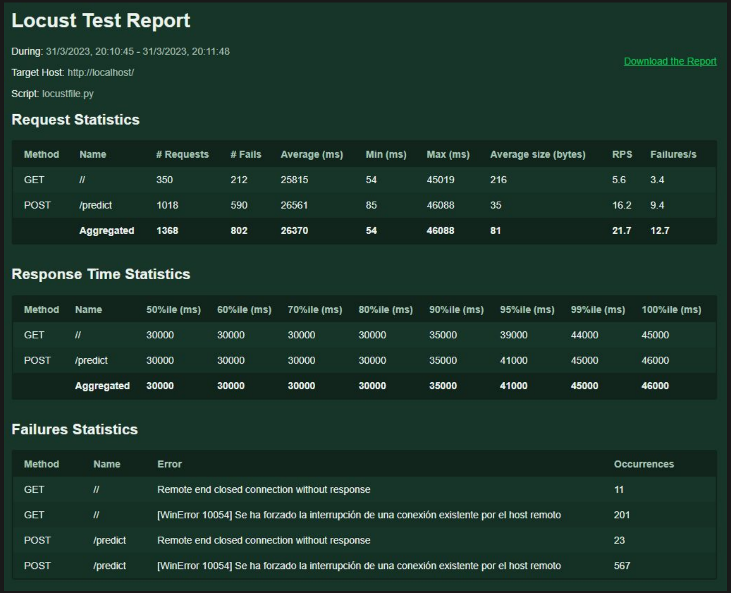
Ratio per User class

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Total ratio

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

100 spawn rate



Locust Test Report

During: 31/3/2023, 20:10:45 - 31/3/2023, 20:11:48

Target Host: http://localhost/

Script: locustfile.py

Download the Report

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	350	212	25815	54	45019	216	5.6	3.4
POST	/predict	1018	590	26561	85	46088	35	16.2	9.4
	Aggregated	**1368**	**802**	**26370**	**54**	**46088**	**81**	**21.7**	**12.7**

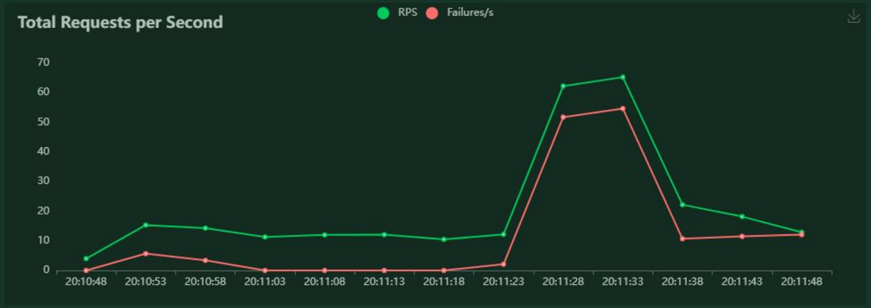
Response Time Statistics

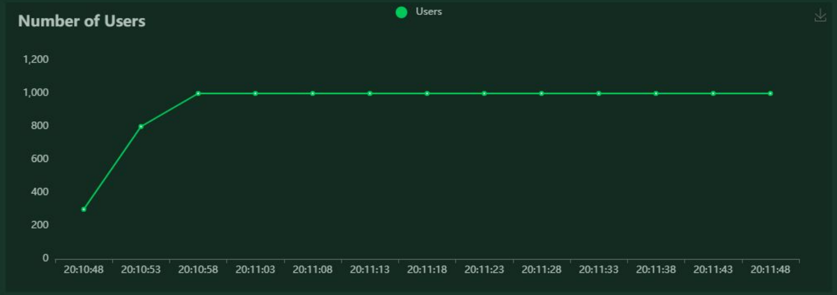
Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	30000	30000	30000	30000	35000	39000	44000	45000
POST	/predict	30000	30000	30000	30000	35000	41000	45000	46000
	Aggregated	**30000**	**30000**	**30000**	**30000**	**35000**	**41000**	**45000**	**46000**

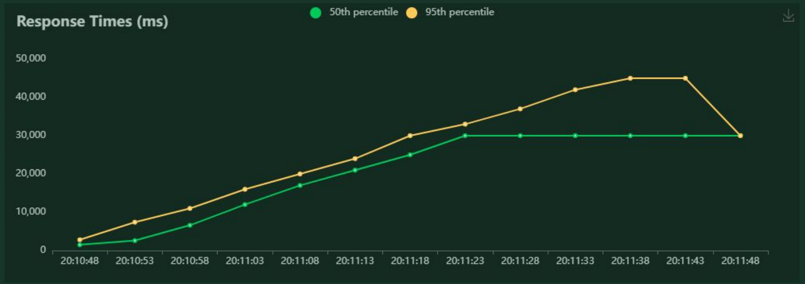
Failures Statistics

Method	Name	Error	Occurrences
GET	//	Remote end closed connection without response	11
GET	//	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	201
POST	/predict	Remote end closed connection without response	23
POST	/predict	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	567

Charts







Final ratio

Ratio per User class

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Total ratio

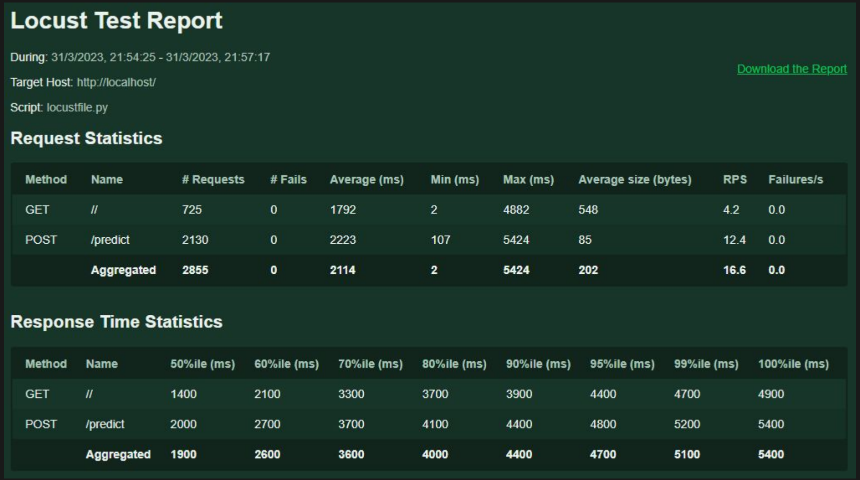
- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

First results

We can see that the computer could hold up to around 600 users before it started to fail, if we let it run a bit longer it fails tremendously, the response time gets low to a certain level and is maintained at the same level the whole time after the failure and practically every request fails.

Now let’s go ahead with 2 instances of the model.

2 instances - 1 spawn rate



Locust Test Report

During: 31/3/2023, 21:54:25 - 31/3/2023, 21:57:17

Target Host: http://localhost/

Script: locustfile.py

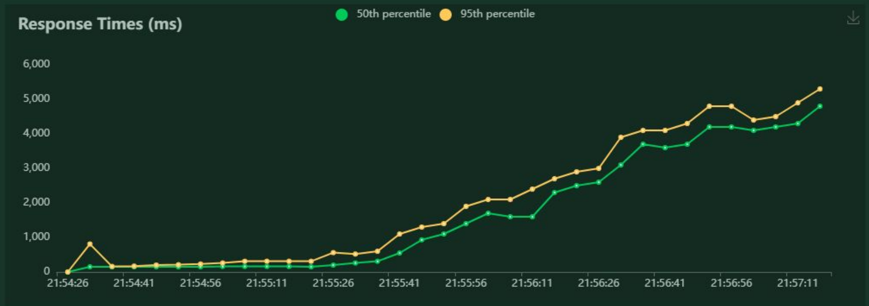
Download the Report

Request Statistics

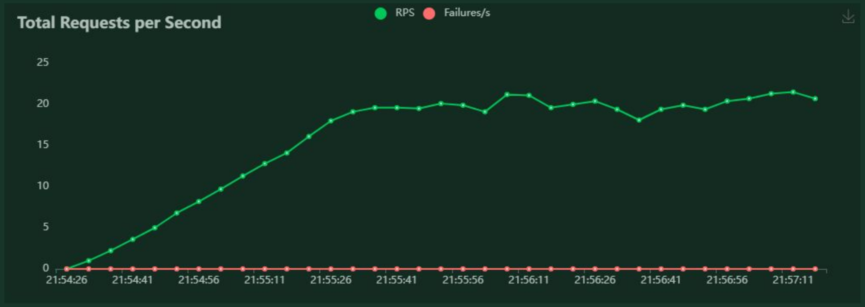
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	725	0	1792	2	4882	548	4.2	0.0
POST	/predict	2130	0	2223	107	5424	85	12.4	0.0
	Aggregated	**2855**	**0**	**2114**	**2**	**5424**	**202**	**16.6**	**0.0**

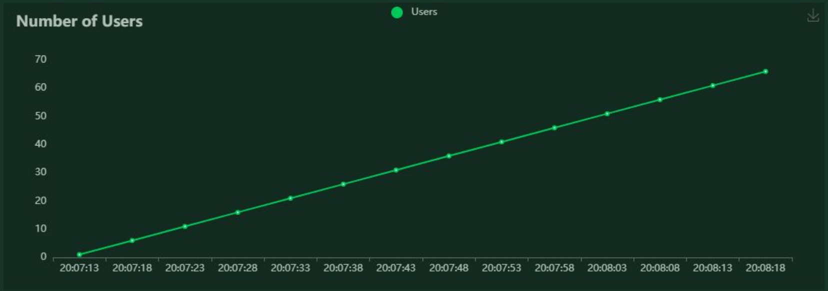
Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	1400	2100	3300	3700	3900	4400	4700	4900
POST	/predict	2000	2700	3700	4100	4400	4800	5200	5400
	Aggregated	**1900**	**2600**	**3600**	**4000**	**4400**	**4700**	**5100**	**5400**



Charts





Final ratio

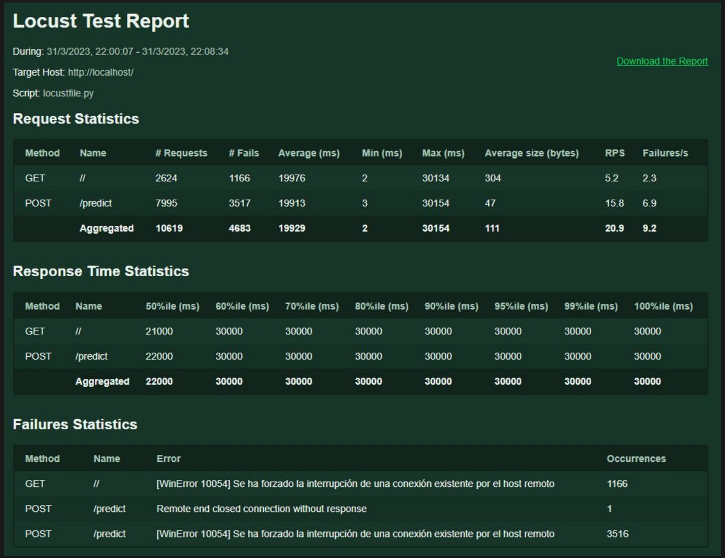
Ratio per User class

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Total ratio

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

2 instances - 2 spawn rate



Locust Test Report

During: 31/3/2023, 22:00:07 - 31/3/2023, 22:08:34

Target Host: http://localhost/

Script: locustfile.py

Download the Report

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	2624	1166	19976	2	30134	304	5.2	2.3
POST	/predict	7995	3517	19913	3	30154	47	15.8	6.9
	Aggregated	**10619**	**4683**	**19929**	**2**	**30154**	**111**	**20.9**	**9.2**

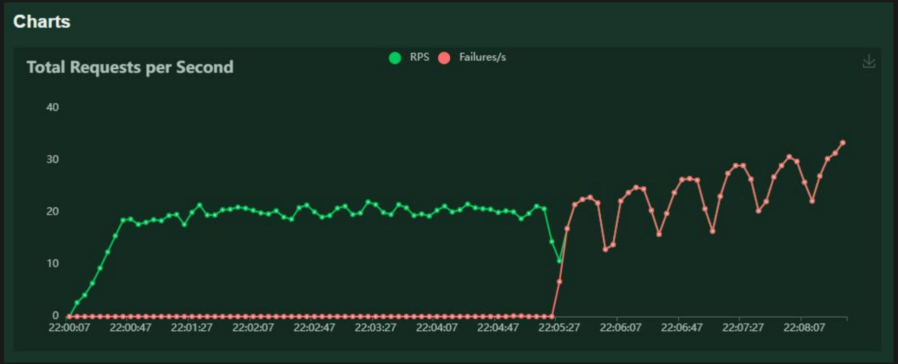
Response Time Statistics

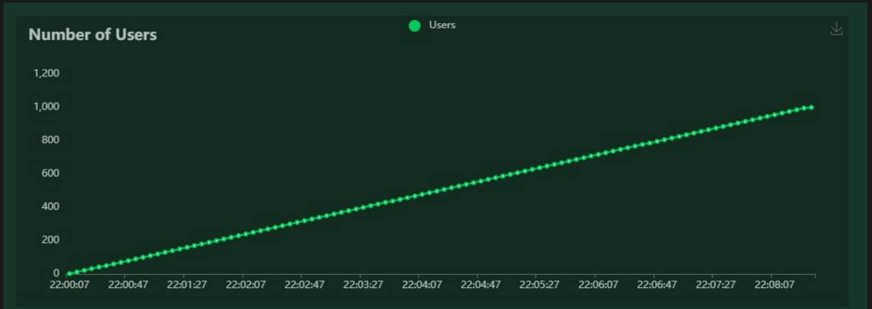
Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	21000	30000	30000	30000	30000	30000	30000	30000
POST	/predict	22000	30000	30000	30000	30000	30000	30000	30000
	Aggregated	**22000**	**30000**	**30000**	**30000**	**30000**	**30000**	**30000**	**30000**

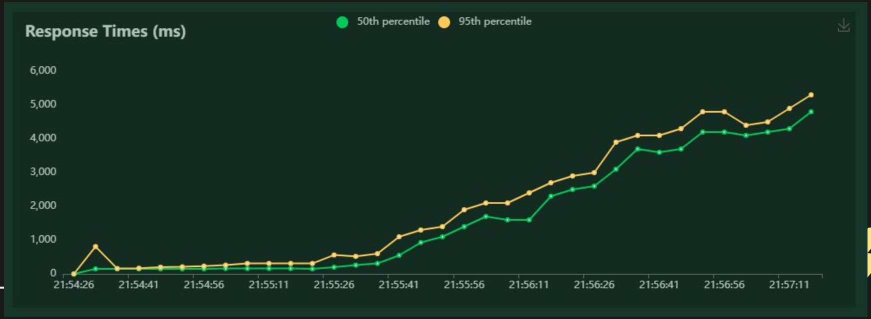
Failures Statistics

Method	Name	Error	Occurrences
GET	//	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	1166
POST	/predict	Remote end closed connection without response	1
POST	/predict	[WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto	3516

Charts







Final ratio

Ratio per User class

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Total ratio

- 100.0% APIUser
  - 25.0% index
  - 75.0% predict

Final results

As a final result we can also see that with 2 instances of the model, the test still fails at around 600 users, just as intense as with 1 instance.