Raport z Testów

SUT *(System under test):*

Aplikacja Django z dwoma endpointami:
- */*
- */hello*

Parametry:

1. Liczba użytkowników (zgodnie z tabelą)
2. Stosunek requestów
   - 25% */hello*
   - 75% */index*

Warunki:

Lokalny setup zgodny z *README.md*

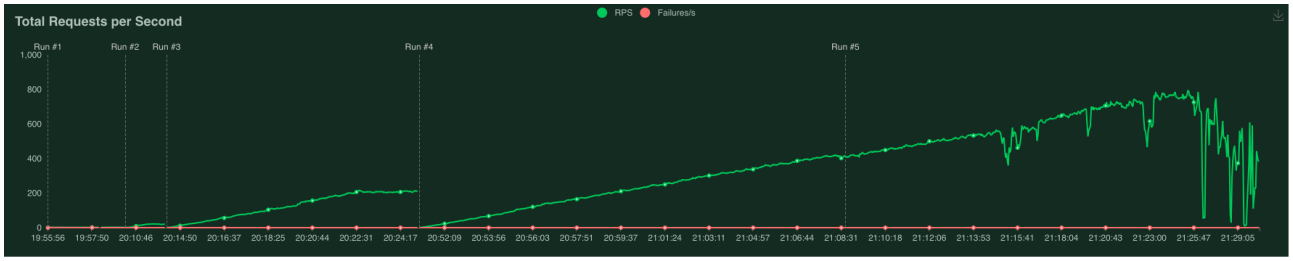
Cel testu:

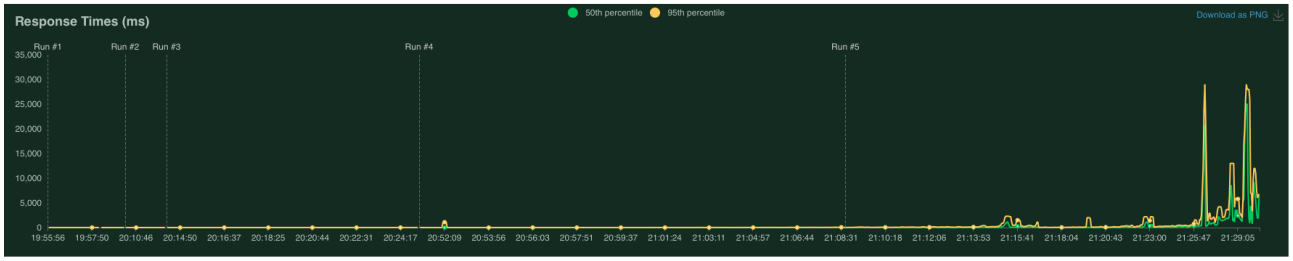
Weryfikacja jak serwer Django radzi sobie z obsługą wielu requestów na raz.
Ponadto na wyniki mogą mieć wpływ: karta sieciowa, implementacja enepointów, możliwości maszyny/obłożenie CPU.

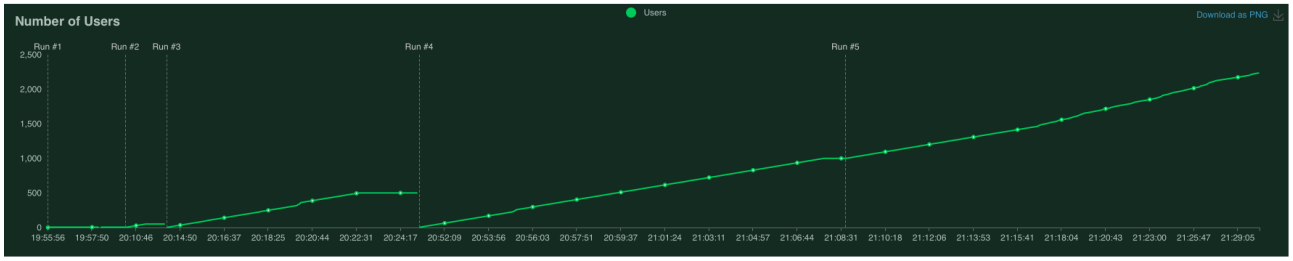
Wyniki:

Nr przypadku testowego	Liczba użytkowników	Spawn rate	Host	Failures
1	5	1	http://127.0.0.1:8000	0%
2	50	1	http://127.0.0.1:8000	0%
3	500	1	http://127.0.0.1:8000	0%
4	1000	1	http://127.0.0.1:8000	0%

Nr przypadku testowego	Liczba użytkowników	Spawn rate	Host	Failures
5	5000	1	http://127.0.0.1:8000	Brak danych







Interpretacja wyników i wnioski

- Testy zostały zakończone przy 2240 użytkownikach przez możliwości komputera, na którym przeprowadzane były testy.
- We wszystkich przypadkach Locust zaraportował 0% nieudanych zapytań.
- Czasy oczekiwania na odpowiedz znacząco wzrosły do **nawet** 25 sekund przy około 2000 użytkowników.
  Istotne jest aby zaznaczyć, że na taki wynik ma wpływ użyta fixtura - lokalny komputer, podejrzewam, że to nie możliwości Django są ograniczone, a właśnie zasoby CPU nie pozwoliły na obsługę tak dużej ilości zapytań.
- Użyteczność takich testów uważam za *mocno limitowaną*. Przy dobrej randomizacji danych można uzyskać efekt fuzz-testów - pozwoliłoby to wykryć potencjalne błędy w logice, natomiast swarm testing na lokalnej maszynie efektywnie testuje możliwości lokalnej maszyny, nie serwera.