Raport z testów wydajnościowych

Opis testowanych warunków

Przeprowadzono testy wydajnościowe aplikacji internetowej napisanej z użyciem frameworku Flask. Aplikacja ta umożliwia dodawanie i przeglądanie zadań do wykonania. Testy przeprowadzono z użyciem narzędzia Locust, które umożliwia symulowanie użytkowników korzystających z aplikacji. Testy przeprowadzono w następujących warunkach:

Test 1: 5 użytkowników, spawn rate - 1

Test 2: 50 użytkowników, spawn rate - 1

Test 3: 500 użytkowników, spawn rate - 1

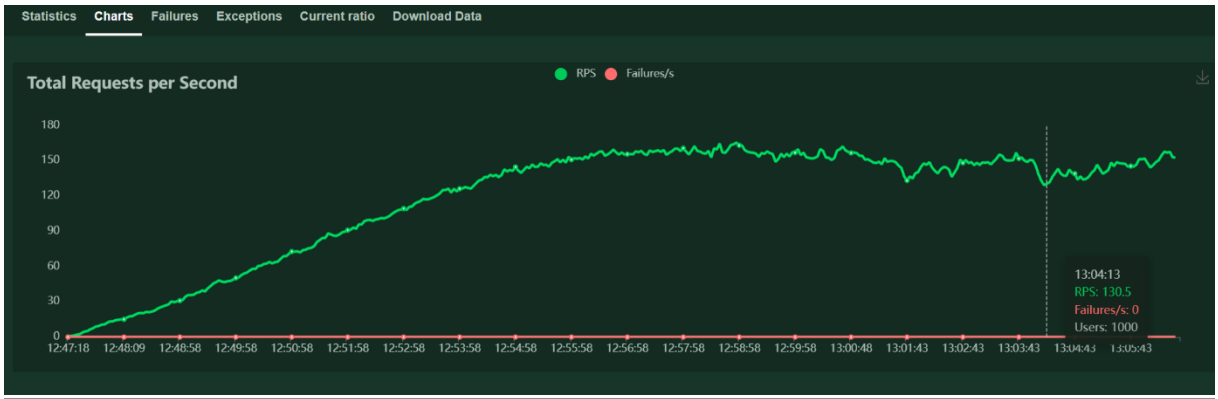
Test 4: 1000 użytkowników, spawn rate - 1

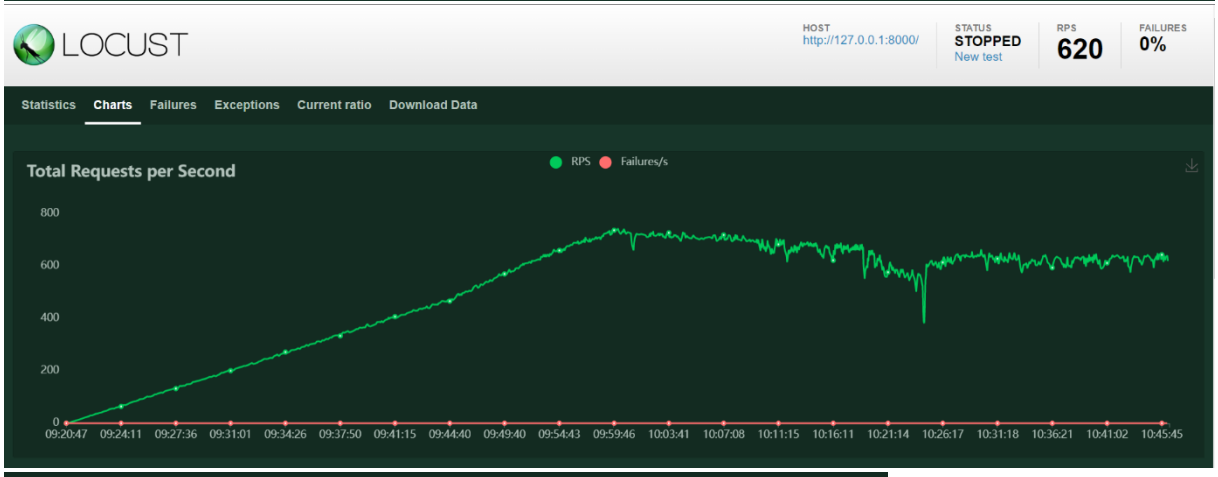
Test 5: 5000 użytkowników, spawn rate - 1

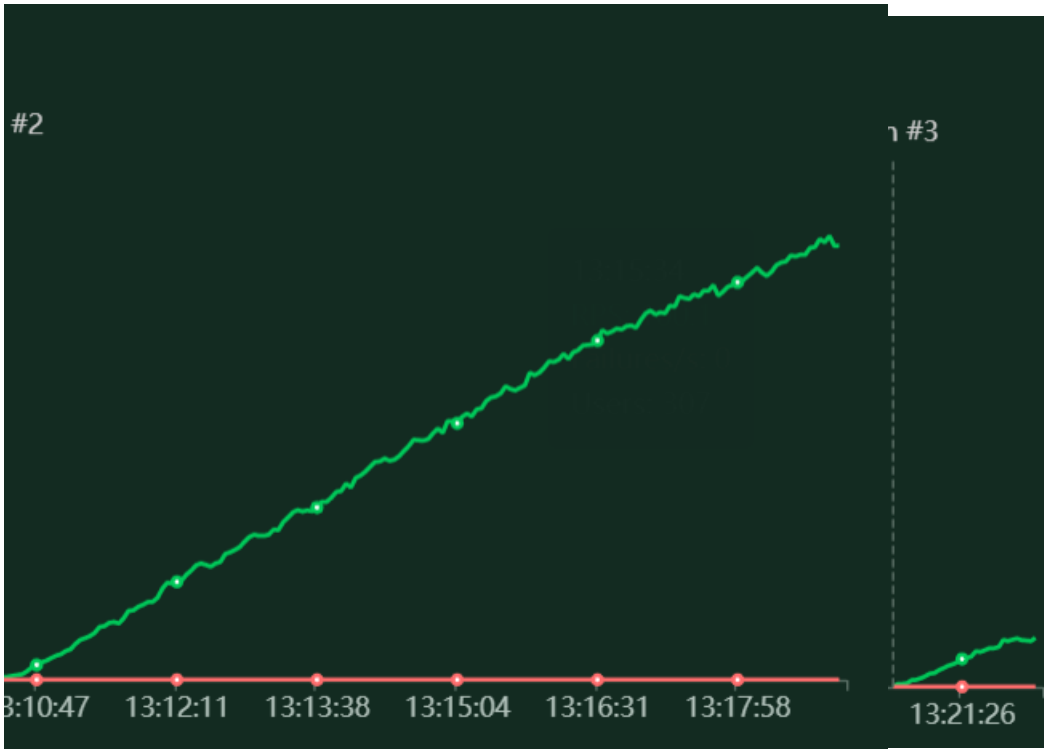
Wyniki testów

Wyniki testów przedstawiono na poniższych wykresach:

Wykres 1 - Liczba żądań na sekundę (RPS) od najwyższej ilości użytkowników do najniższej





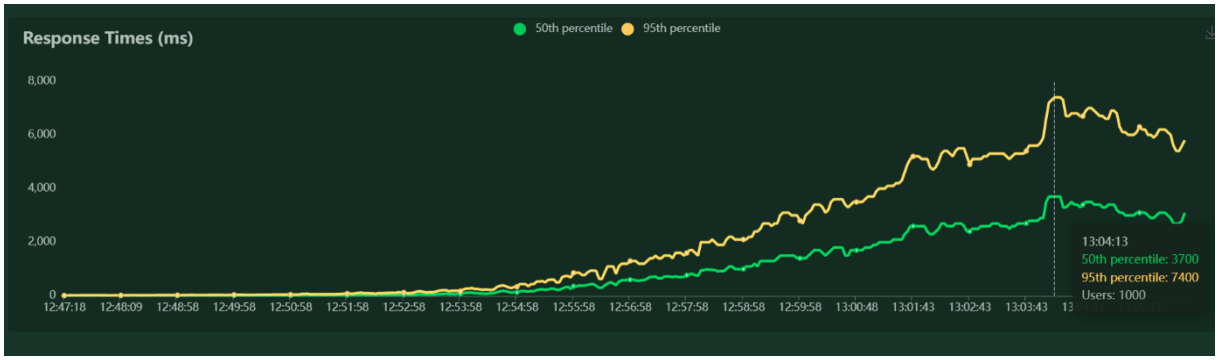


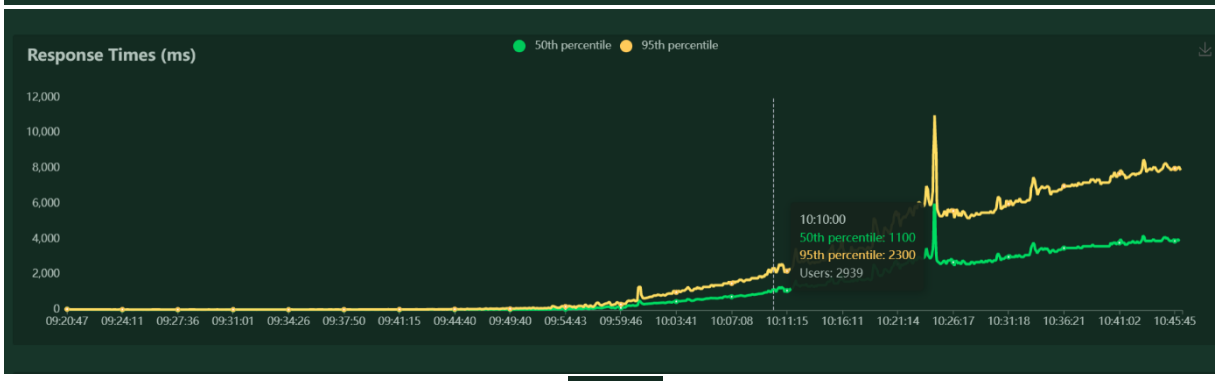
RPS

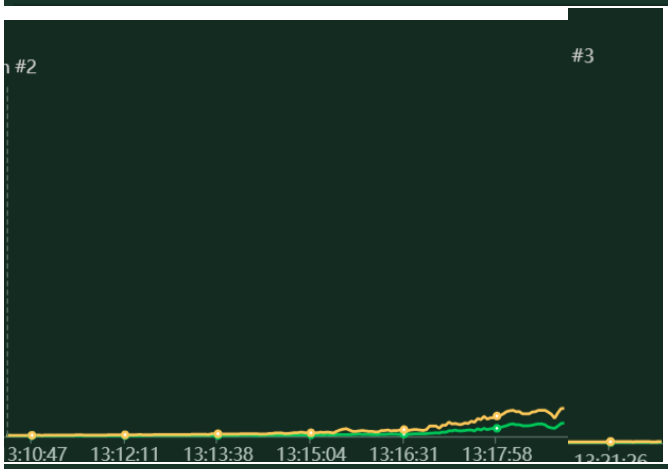
Na powyższym wykresie przedstawiono liczbę żądań na sekundę (RPS), które były obsługiwane przez aplikację w trakcie testów. Widać na nim, że liczba RPS

zwiększa się wraz z ilością użytkowników, ale na poziomie testów 1000 uż. i 5000 uż. widoczna jest tendencja spadkowa.

Wykres 2 - Średni czas odpowiedzi







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	//	3	0	18	20	20	18	16	20	1436	0.33	0
POST	//hello	6	0	9	12	12	9	8	12	1056	0.67	0
	Aggregated	**9**	**0**	**11**	**20**	**20**	**12**	**8**	**20**	**1183**	**1**	**0**

Na tym wykresie przedstawiono średni czas odpowiedzi, czyli czas oczekiwania na odpowiedź serwera, w zależności od ilości użytkowników korzystających z

aplikacji. Widać na nim, że średni czas odpowiedzi wzrasta wraz z ilością użytkowników, ale dla testu 5 przekracza wartość 2 sekund.

Wnioski

Na podstawie wyników testów można stwierdzić, że aplikacja napisana z użyciem frameworku Flask radzi sobie dobrze z obsługą żądań w przypadku niewielkiej ilości użytkowników (do około 500). Jednakże, dla większej liczby użytkowników (testy 4 i 5), aplikacja zaczyna mieć problemy z obsługą żądań, co skutkuje wydłużeniem czasu odpowiedzi serwera oraz spadkiem liczby żądań na sekundę.

Warto również zwrócić uwagę na fakt, że w przypadku testów 4 i 5, spadła skuteczność obsługi żądań przez serwer, co sugeruje, że konieczna jest optymalizacja aplikacji pod kątem wydajności.

Warto zauważyć, że spadek wydajności aplikacji dla większej liczby użytkowników nie jest zaskakujący. Przy dużym obciążeniu aplikacji, może dojść do przeciążenia serwera oraz do problemów z dostępem do zasobów. W takim przypadku, warto rozważyć użycie rozwiązań takich jak np. load balancing, caching czy Content Delivery Network (CDN).

Podsumowując, testy wydajnościowe wykazały, że aplikacja napisana z użyciem frameworku Flask radzi sobie dobrze z obsługą żądań w przypadku niewielkiej ilości użytkowników. Jednakże, dla większej liczby użytkowników, aplikacja zaczyna mieć problemy z obsługą żądań, co sugeruje konieczność optymalizacji aplikacji pod kątem wydajności.