КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп’ютерних наук та кібернетики

Звіт

Валідація та верифікація програмних систем

Load testing

Виконав студент
групи ІНФ-1
Веремчук Максим Романович

Опис інструментарію

В ході виконання лабораторної роботи проводилось тестування навантаження на пошукову систему(duckduckgo) великою кількістю запитів. Використовувалась мова програмування python в редакторі Visual Studio Code на системі Ubuntu 18.04. Для проведення тестування застосовувався тестувальний фреймворк locustio та bokeh - бібліотека для візуалізації даних.

Порядок роботи

При тестуванні пошукової системи створено клас для створення запиту на веб сторінку. За допомогою locustio фреймворку, ці запити виконувались паралельно з послідовним нарощуванням кількості. У даному випадку, що буде відображено на графіках, кількість штучних юзерів, що запитують ресурс зростала від 0 до 300, з додаванням нового юзера кожну секунду. Запуск фреймворку для початку тестування:

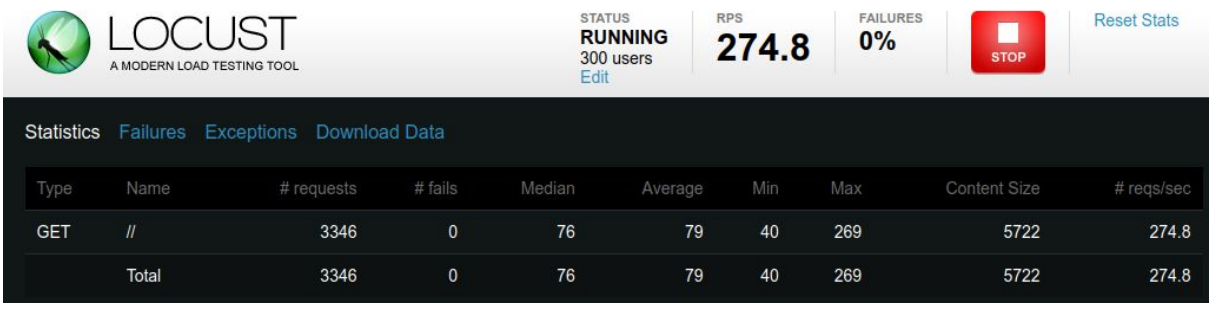
locust --host=https://duckduckgo.com/

Код класу, що викликає locustio:

```
class UserBehavior(TaskSet):
    @task(1)
    def index(self):
        self.client.get('/')

class WebsiteUser(HttpLocust):
    task_set = UserBehavior
```

Наступний скриншот демонструє одночасні запити 300 юзерів зі швидкістю приблизно 275 запитів у секунду.



Type	Name	# requests	# fails	Median	Average	Min	Max	Content Size	# reqs/sec
GET	//	3346	0	76	79	40	269	5722	274.8
	Total	3346	0	76	79	40	269	5722	274.8

Видно, що під час тестування не відбулося жодного збою(failure), та усі запити виконані коректно. Щодо швидкості, середнє значення становить близько 80 мілісекунд, що не викликає затримок у користувача. Максимальне значення 270 мілісекунд, що також є не критичним значенням.

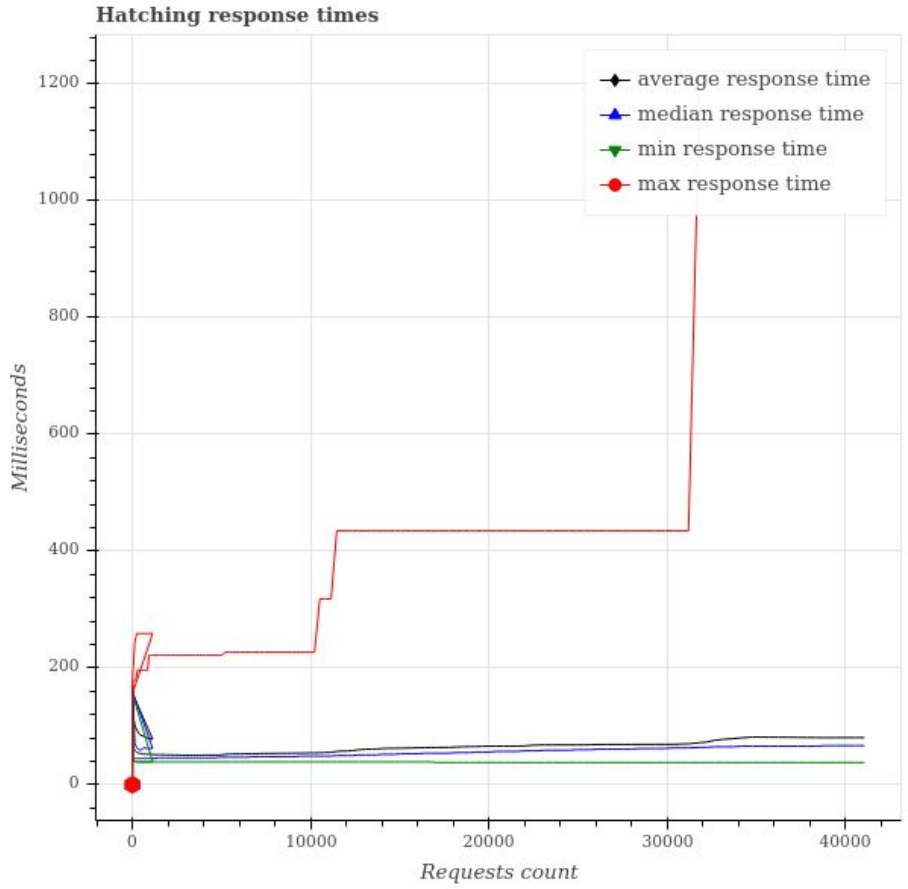
Результати тестування

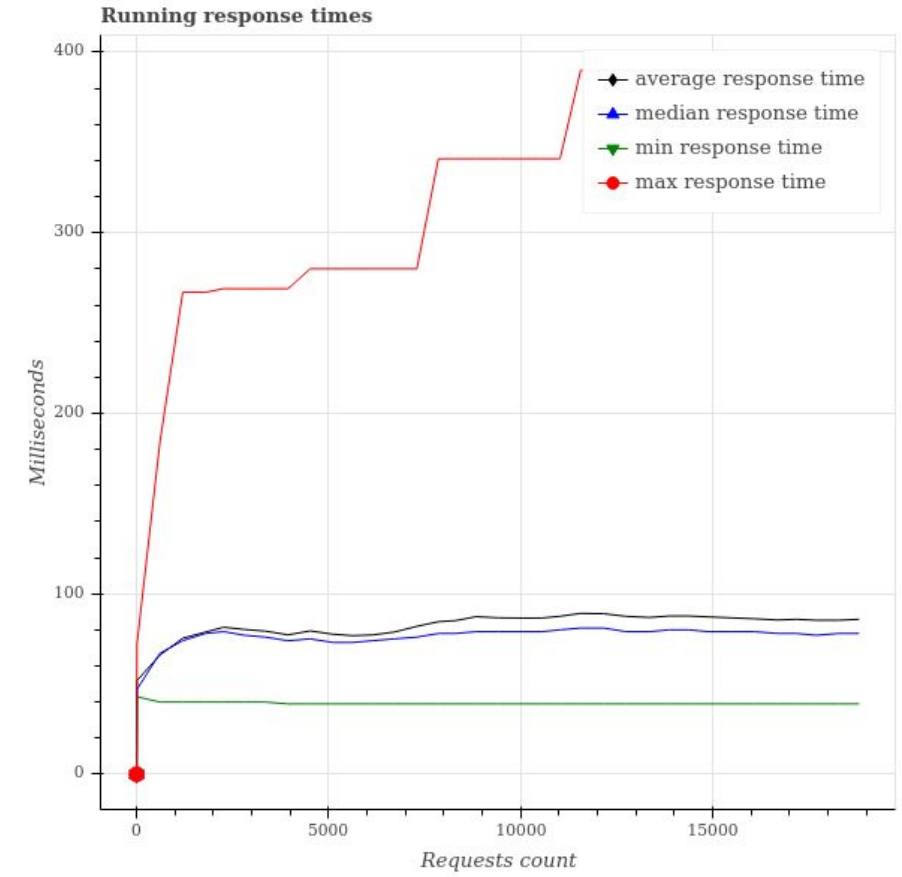
При тестування, за допомогою bokeh та наступної json структури було побудовано 4 графіки продуктивності роботи системи.

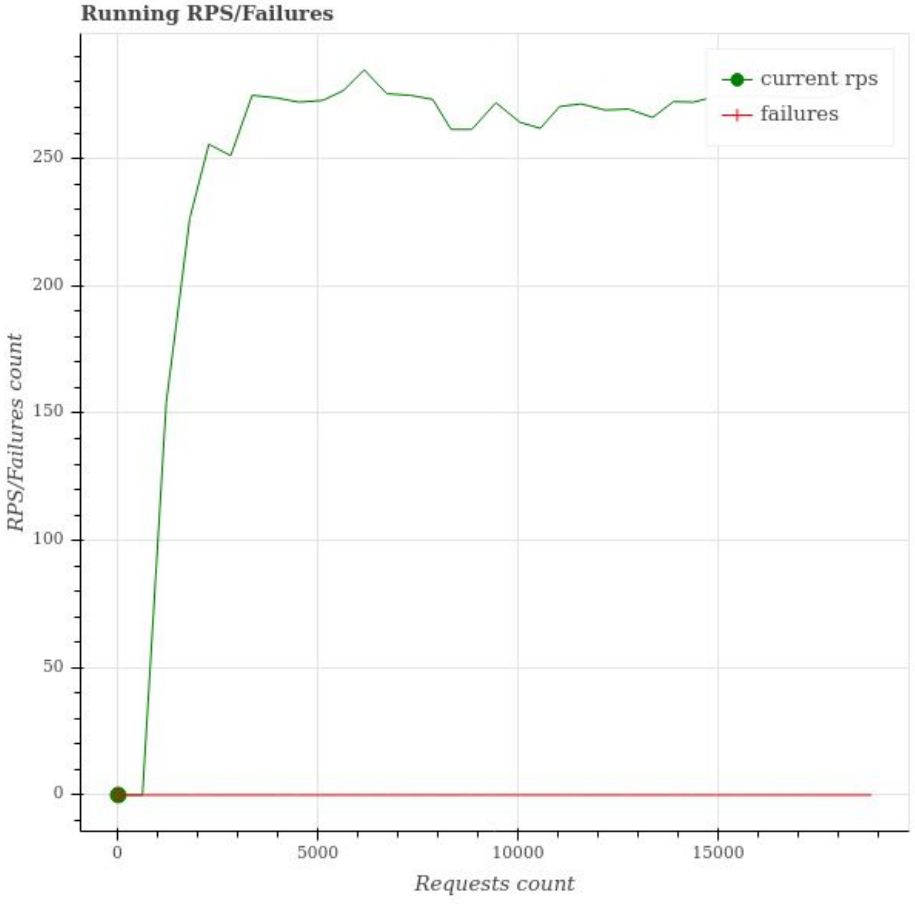
```
config = {'figures': [{'charts':
                          [{'color': 'black', 'legend': 'average
response time', 'marker': 'diamond',
                            'key': 'avg_response_time'},
                           {'color': 'blue', 'legend': 'median response
time', 'marker': 'triangle',
                            'key': 'median_response_time'},
                           {'color': 'green', 'legend': 'min response
time', 'marker': 'inverted_triangle',
                            'key': 'min_response_time'},
                           {'color': 'red', 'legend': 'max response
time', 'marker': 'circle',
                            'key': 'max_response_time'}],
                      'xlabel': 'Requests count',
                      'ylabel': 'Milliseconds',
                      'title': '{} response times'
                      },
                     {'charts': [{'color': 'green', 'legend': 'current
rps', 'marker': 'circle',
                                  'key': 'current_rps'},
                                 {'color': 'red', 'legend': 'failures',
'marker': 'cross',
                                  'key': 'num_failures', 'skip_null':
True}],
                      'xlabel': 'Requests count',
                      'ylabel': 'RPS/Failures count',
                      'title': '{} RPS/Failures'
                      }],
          'url': 'http://localhost:8089/stats/requests',
          'states': ['hatching', 'running'],
```

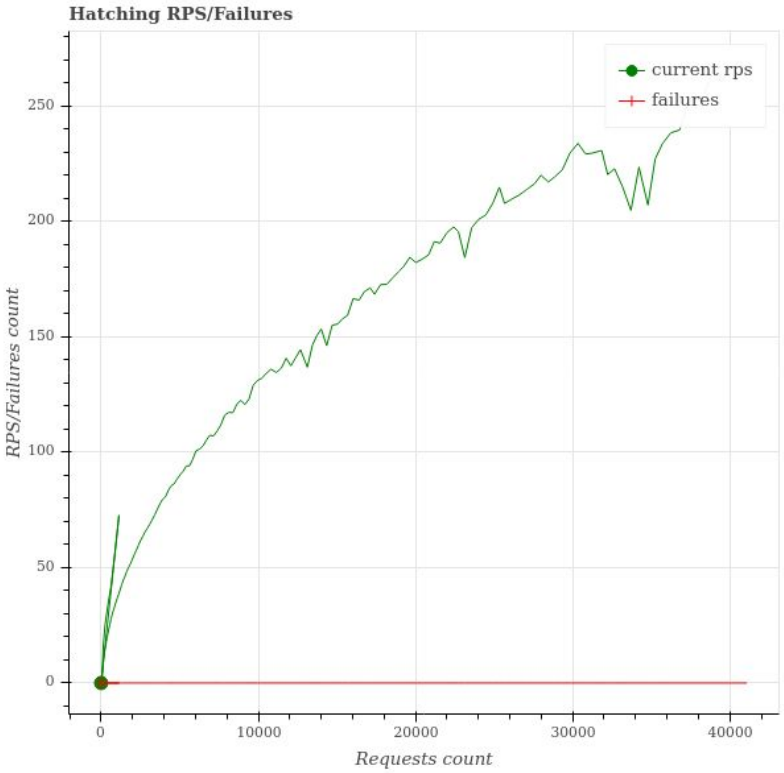
```
        'requests_key': 'num_requests'
        }
```

Наступні графіки є результатом роботи тесту та показують затримку запиту в залежності від кількості запитів, та кількість помилок в залежності від швидкості запитів(помилок завжди 0), та аналогічні графіки які показують ті ж величини відносно приросту юзерів. На графіках показано мінімальні, максимальні, середні та медіанні значення.




Running response times
average response time
median response time
min response time
max response time
Milliseconds
400
300
200
100
0
0
5000
10000
15000
Requests count


Running RPS/Failures
current rps
failures
RPS/Failures count
250
200
150
100
50
0
0
5000
10000
15000
Requests count



Висновки

У ході виконання лабораторної роботи було виконано тестування навантаження пошукової системи за допомогою паралельного надсилання запитів. Якщо брати що пошукові запити надсилаються кожним окремим юзером не дуже часто то система показала себе доволі стійкою, оскільки не було помічено ніяких збоїв.