

Министерство цифрового развития, связи и массовых коммуникаций
Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«Сибирский государственный университет телекоммуникаций и
информатики» (СибГУТИ)

Лабораторная работа №6
«Разработка сценария нагрузочного тестирования с использованием
Locust»

Выполнил студент:
Абрамов М.В.
группа ИП-311

Проверил:
Сентюров С.А.

Новосибирск 2025 г.

СОДЕРЖАНИЕ

1. Введение.....	3
2. Теоретические сведения	3
3. Ход выполнения работы	4
3.1. Настройка окружения	4
3.2. Разработка Locust-скрипта (locustfile.py).....	4
3.3. Запуск нагрузочного тестирования	5
4. Результаты тестирования.....	5
4.1. Сводная таблица результатов (Пример запуска)	5
4.2. Анализ результатов	6
5. Заключение	6
6. Приложение (Скриншоты)	7

1. Введение

Цель работы: Научиться разрабатывать сценарии нагрузочного тестирования с использованием **Locust**, тестируя **API OpenBMC** и открытое публичное **API**. Оценить производительность системы, проанализировать задержки и возможные точки отказа.

Задачи:

1. Установить и настроить инструмент нагрузочного тестирования **Locust**.
2. Разработать **Locust-скрипт** (`locustfile.py`) на **Python**, включающий сценарии для тестирования **OpenBMC API** (получение информации о системе и состояния питания).
3. Включить в сценарий тестирование публичного API (например, `wttr.in`).
4. Провести нагрузочное тестирование с заданными параметрами (пользователи, скорость запуска).
5. Проанализировать и представить полученные метрики (среднее время отклика, процент ошибок).

2. Теоретические сведения

Locust

Locust — это инструмент для нагрузочного тестирования, который позволяет моделировать поведение множества виртуальных пользователей (написанных на **Python**) и отправлять **HTTP-запросы** к тестируемому **API** или веб-приложению. Основные возможности включают: создание кастомных сценариев, запуск тестов с заданным числом пользователей и мониторинг времени отклика в реальном времени.

OpenBMC API (Redfish)

OpenBMC API предоставляет **Redfish-интерфейс** для удаленного управления серверным оборудованием. В рамках работы использовались запросы типа GET для получения данных о системе и ее состоянии питания.

3. Ход выполнения работы

3.1. Настройка окружения

1. Установлен инструмент **Locust** (версия 3.12.3 на Python).
2. Проверена доступность **OpenBMC API** (<https://localhost:2443/redfish/v1/>) с помощью **curl** с использованием учетных данных root:OpenBmc. Запрос выполнен успешно.
3. Для публичного тестирования выбран сервис **wttr.in**, доступность которого также подтверждена **curl**.
4. Проверка **JSONPlaceholder** завершилась ошибкой соединения, поэтому тестирование проводилось только с **OpenBMC** и **wttr.in**.

3.2. Разработка Locust-скрипта (locustfile.py)

Разработан скрипт с использованием двух классов для моделирования нагрузки:

1. **OpenBMCTest** (Host: <https://localhost:2443>):
 - о **Задача:** Получение информации о системе (GET /redfish/v1/Systems/system).
 - о **Задача:** Получение состояния питания (GET /redfish/v1/Systems/system), с извлечением поля PowerState из JSON-ответа.
 - о **Особенность:** Для работы с **OpenBMC API** использовалась базовая аутентификация (auth=("root", "OpenBmc")) и отключение проверки SSL-сертификата (verify=False).
2. **WeatherTest** (Host: <https://wttr.in>):

- **Задача:** Получение прогноза погоды для Новосибирска (GET /Novosibirsk?format=j1).
- Ожидание между запросами (wait_time) для обоих классов установлено в диапазоне от **1 до 3 секунд**.

3.3. Запуск нагрузочного тестирования

Запуск **Locust** выполнен через команду locust, после чего тест инициирован через веб-интерфейс:

- **Количество пользователей (Number of users):** 10
- **Скорость запуска (Ramp Up):** 2 пользователя/сек
- **Общее количество пользователей:** 10, распределенных поровну (5 OpenBMCTest, 5 WeatherTest).

4. Результаты тестирования

4.1. Сводная таблица результатов (Пример запуска)

Сводные результаты, полученные в ходе нагрузочного тестирования (фрагмент статистики):

Тип	Имя запроса	Запросов	Ошибка	% Ошибок	Медиана (мс)	95% перцентиль (мс)	Среднее (мс)
GET	/Novosibirsk?format=j1	7	7	100%	73	110	72.67
GET	/redfish/v1/Systems/system	6	0	0%	410	660	443.42
Агрегировано	-	13	7	53.8%	110	660	243.78
38							

4.2. Анализ результатов

1. Тестирование OpenBMC API (<https://localhost:2443>)

- Показана **отличная стабильность с 0% ошибок**³⁹. Все 6 запросов на получение информации о системе были выполнены успешно.
- Среднее время отклика составило **443.42 мс**⁴⁰. **Медиана (50% запросов)** составила **410 мс**, а **95% запросов** были обработаны быстрее, чем за **660 мс**⁴¹. Такие показатели свидетельствуют о приемлемой производительности системы под данной нагрузкой.

2. Тестирование публичного API (<https://wttr.in>)

- Зафиксирован **100% процент ошибок** (7 из 7 запросов)⁴².
- Низкое время отклика (**72.67 мс** в среднем) ⁴³ при полном отсутствии успешных запросов указывает на то, что ошибка, скорее всего, связана не с производительностью, а с **некорректной обработкой ответа** или его формата скриптом **Locust**, который ожидает успешный статус-код, отличный от полученного, либо некорректно парсит ответ, не являющийся **JSON**.

3. Общий вывод по нагрузке

- Из-за проблем с публичным API, общий процент ошибок составил **53.8%**⁴⁴.
- Однако целевой объект тестирования (**OpenBMC API**) полностью справился с заданной нагрузкой (10 пользователей, 2 пользователя/сек) с **0% ошибок**, продемонстрировав хорошую доступность и скорость отклика.

5. Заключение

В ходе лабораторной работы №6 были успешно освоены навыки разработки сценариев нагрузочного тестирования с использованием фреймворка **Locust**.

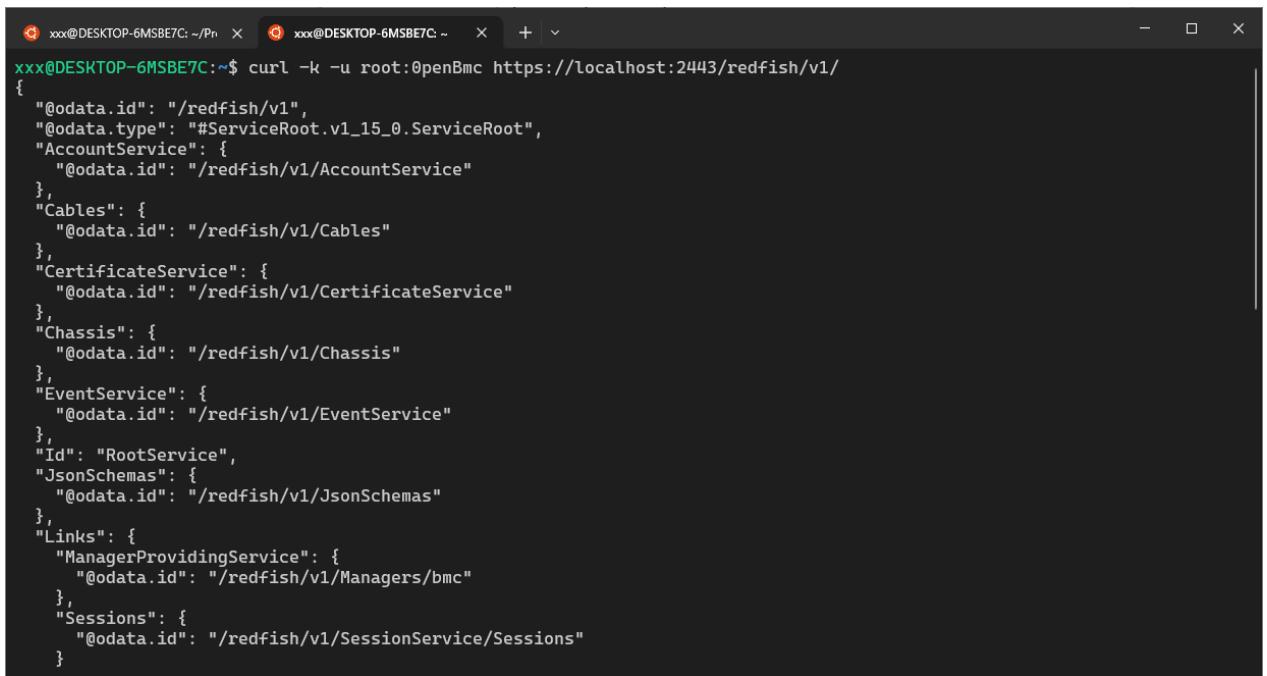
1. Был разработан скрипт на **Python** (locustfile.py), имитирующий нагрузку на две различные системы: **OpenBMC API** и открытый сервис **wttr.in**.
2. Нагрузочное тестирование **OpenBMC API** показало **высокую надежность (0% ошибок)** и **приемлемое среднее время отклика (443.42 мс)**, что подтверждает его готовность к эксплуатации.
3. Была выявлена проблема при тестировании публичного API (wttr.in), что привело к **100% ошибок** в этом сегменте.

Цели работы достигнуты: сценарий нагрузочного тестирования разработан, запуск произведен, а результаты проанализированы и представлены в отчете.

6. Приложение (Скриншоты)

(В данном разделе должны быть представлены скриншоты, подтверждающие выполнение работы)

```
(venv) xxx@DESKTOP-6MSBE7C:~/Projects/QTStudy/testPO/lab6$ locust -V
locust 2.42.4 from /home/xxx/Projects/QTStudy/testPO/lab6/venv/lib/python3.12/site-packages/locust (Python 3.12.3)
```



```
curl -k -u root:OpenBmc https://localhost:2443/redfish/v1/
{
  "@odata.id": "/redfish/v1",
  "@odata.type": "#ServiceRoot.v1_15_0.ServiceRoot",
  "AccountService": {
    "@odata.id": "/redfish/v1/AccountService"
  },
  "Cables": {
    "@odata.id": "/redfish/v1/Cables"
  },
  "CertificateService": {
    "@odata.id": "/redfish/v1/CertificateService"
  },
  "Chassis": {
    "@odata.id": "/redfish/v1/Chassis"
  },
  "EventService": {
    "@odata.id": "/redfish/v1/EventService"
  },
  "Id": "RootService",
  "JsonSchemas": {
    "@odata.id": "/redfish/v1/JsonSchemas"
  },
  "Links": {
    "ManagerProvidingService": {
      "@odata.id": "/redfish/v1/Managers/bmc"
    },
    "Sessions": {
      "@odata.id": "/redfish/v1/SessionService/Sessions"
    }
}
```

```
xxx@DESKTOP-6MSBE7C:~/Pn > xxx@DESKTOP-6MSBE7C:~ > + <
}
}xxx@DESKTOP-6MSBE7C:~$ curl -k -u root:@openBmc https://wttr.in/
Weather report: Novosibirsk, Russia

      .-.   Snow
     (   ) . +1(-3) °C
    (_____)  ↗ 18 km/h
   * * * *  1 km
* * * *  1.2 mm

[Thu 20 Nov]
Morning | Noon | Evening | Night
\ / Partly Cloudy | .-. Light freezing | .-. Light sleet | .-. Heavy snow
/-"-.-. +1(-5) °C | (   ). 0(-5) °C | (   ). 0(-5) °C | (   ). 0(-4) °C
\_(   ) . ↗ 23-36 km/h | (_____) ↗ 22-36 km/h | (_____) ↗ 19-30 km/h | (_____) ↗ 17-26 km/h
/(_____) 10 km | * * * * 10 km | * * * * 9 km | * * * * 2 km
0.0 mm | 0% | * * * * 0.1 mm | 100% | * * * * 2.1 mm | 100% | * * * * 0.7 mm | 100%
0% |
```

```
Follow @igor_chubin for wttr.in updates
xxx@DESKTOP-6MSBE7C:~/Pn  x  xxx@DESKTOP-6MSBE7C:~  x  +  v

[{"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit", "body": "quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\n reprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"}, {"userId": 1, "id": 2, "title": "qui est esse", "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla"}, {"userId": 1, "id": 3, "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut", "body": "et iusto sed quo iure\\nvolutatem occaecati omnis eligendi aut ad\\nvolutatem doloribus vel accusantium quis pariatur\\nmolestiae porro eius odio et labore et velit aut"}, {"userId": 1, "id": 4, "title": "eum et est occaecati", "body": "ullam et saepe reiciendis voluptatem adipisci\\nsit amet autem assumenda provident rerum culpa\\nquis hic omnis\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"}]
```

1. Установка Locust и проверка доступности API (Curl запросы к OpenBMC и wttr.in).

```
locustfile.py X
Ubuntu > home > xxx > Projects > QTStudy > testPO > lab6 > locustfile.py > OpenBMCTest > get_system_info_and_power_state
1  from locust import HttpUser, task, between
2  import json
3
4
5  OPENBMC_HOST = "https://localhost:2443"
6  OPENBMC_AUTH = ("root", "OpenBmc")
7
8
9  class OpenBMCTest(HttpUser):
10     host = OPENBMC_HOST
11     wait_time = between(1, 3)
12     disable_known_hosts = True
13
14     def on_start(self):
15         self.client.auth = OPENBMC_AUTH
16         self.client.verify = False
17
18     @task(1)
19     def get_system_info_and_power_state(self):
20         """Запрос информации о системе и проверка состояния питания."""
21         response = self.client.get(
22             "/redfish/v1/Systems/system", name="GET /Systems/system"
23         )
24
25         response.raise_for_status()
26
27         try:
28             system_info = response.json()
29
30             power_state = system_info.get("PowerState", "Unknown")
31             print(f"System info: {response.status_code}. Power state: {power_state}")
32
33             valid_power_states = ["On", "Off", "PoweringOn", "PoweringOff", "Unknown"]
34
35             if power_state not in valid_power_states:
36
37                 raise Exception(
38                     f"Получено недопустимое состояние питания: {power_state}"
39                 )
40
41         except json.JSONDecodeError:
42
43             raise Exception("Ошибка декодирования JSON в ответе OpenBMC.")
44
45
46     class WeatherTest(HttpUser):
47         host = "https://wttr.in"
48         wait_time = between(1, 3)
49
50         @task(1)
51         def get_novosibirsk_weather(self):
52             response = self.client.get("/Novosibirsk?format=j1", name="GET /Novosibirsk")
53
54             response.raise_for_status()
55
56             try:
57                 weather_data = response.json()
58                 current_temp = weather_data["current_condition"][0]["temp_C"]
59                 print(f"Weather: {response.status_code}. Temp: {current_temp}°C")
60             except (KeyError, json.JSONDecodeError) as e:
61
62                 raise Exception(f"Ошибка парсинга или структуры ответа wttr.in: {e}")
63
64
65     class MyLoadTest(OpenBMCTest, WeatherTest):
66         pass
67
```

```
xxx@DESKTOP-6MSBE7C:~/Projects/QTStudy/testPO/lab6$ locust
[2025-11-20 20:57:40,403] DESKTOP-6MSBE7C/INFO/locust.main: Starting web interface at http://0.0.0.0:8089
[2025-11-20 20:57:40,414] DESKTOP-6MSBE7C/INFO/locust.main: Starting Locust
```

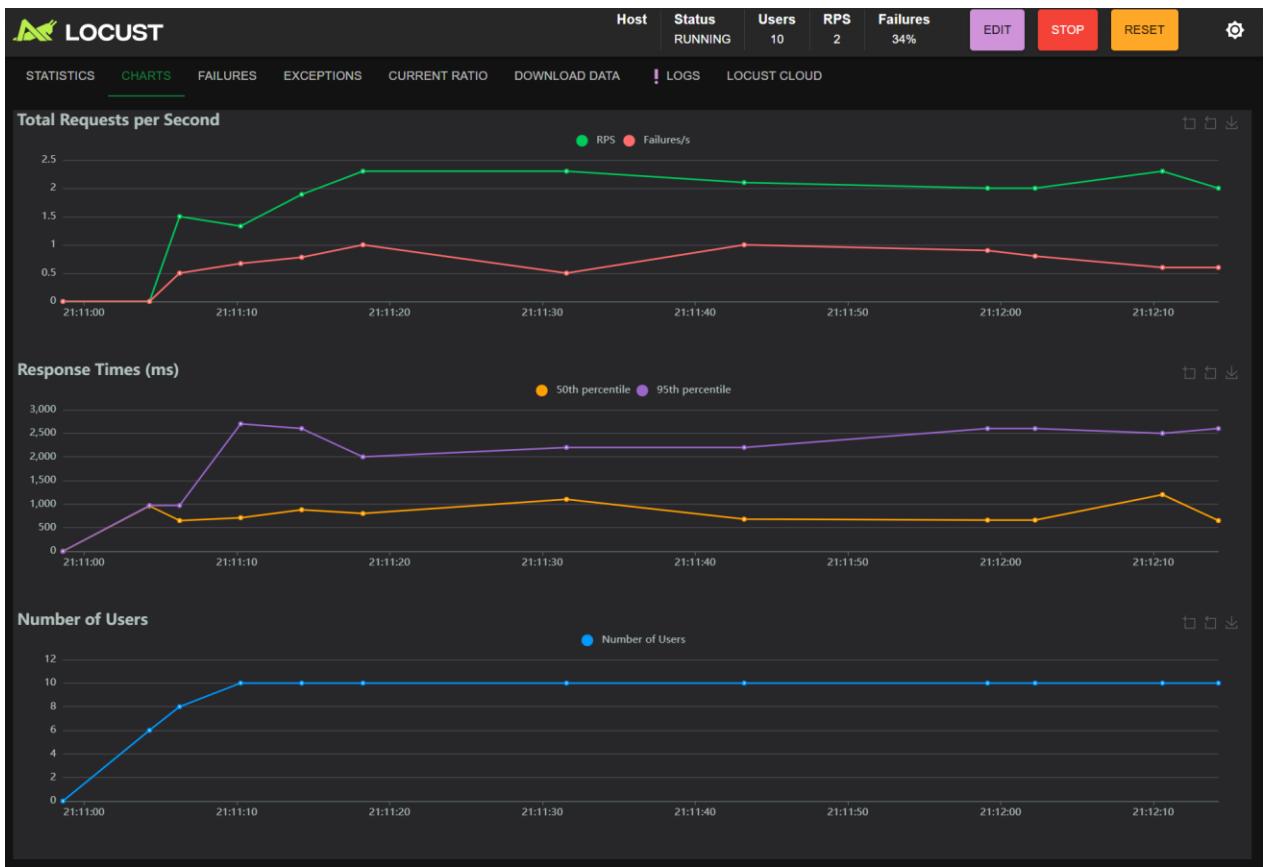
2. Листинг кода locustfile.py (Определение классов OpenBMCTest и WeatherTest).

The screenshot shows the Locust web interface for starting a new load test. At the top, there are status indicators: HOST READY, RPS 0, and FAILURES 0%. Below these are input fields for 'Number of users (peak concurrency)' set to 10 and 'Ramp Up (users started/second)' set to 2. A dropdown menu labeled 'Advanced options' is partially visible. At the bottom is a large green button labeled 'START SWARM'.

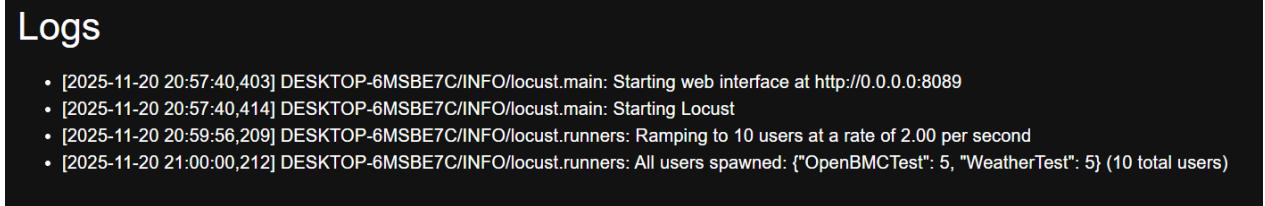
3. Веб-интерфейс Locust (Начало теста: 10 пользователей, Ramp Up 2).

The screenshot shows the Locust web interface on the 'STATISTICS' tab during a test run. The top bar displays 'Host RUNNING', 'Users 10', 'RPS 2', and 'Failures 35%'. Below the bar are buttons for 'EDIT', 'STOP', and 'RESET'. The main area shows a table of test results:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	GET /Novosibirsk	46	46	300	710	900	383.57	190	903	0	0.8	0.8
GET	GET /Systems/system	84	0	1300	2600	2700	1390.02	385	2678	2614	1.2	0
	Aggregated	130	46	800	2300	2600	1033.89	190	2678	1689.05	2	0.8



4. Сводная статистика Locust (Таблица с метриками RPS, Fails, Median, Average).



5. Логи работы Locust (Вывод в терминале с System info: 200 и Power state: Off).