Measurement of DSPACE7 with Locust - an open source load test tool

(Experiences from the University of Zurich, IT, ZORA Repository:)

To test the performance of our services (Eprints, DSpace, ...) we recommend the open source load testing tool Locust - https://locust.io

We want to test the multiple and parallel download of static and dynamic pages, the upload, various workflow steps and multiple user logins.

As I have not found any example configurations for DSpace, here is a short guide.

Installation

Instructions according to https://docs.locust.io/en/stable/installation.html

- Lokal (Win, Linux, Mac) vs. RHEL8 Server; both installed and tested. I prefer a ServerSide installation, because bigger hardware and therefore more pressure during load tests.
- python 3.11
- gcc-c++
- pip3.11
- open Port 8089

locust install: pip3.11 install locust

01 Simple scenarios ("Hello World" mode)

Configuration

Simply pass through the URL entered at the start in the input form ("Number of Users", "Spawn rate", "Host") and test it - e.g. test the homepage of your DS7 repo

vi locustfile.py

```
from locust import HttpUser, task

class HelloWorldUser(HttpUser):

@task def

   hello_world(self):

   self.client.get("/")
```

more on configs read https://docs.locust.io/en/stable/writing-a-locustfile.html#writing-a-locustfile

Start Locust

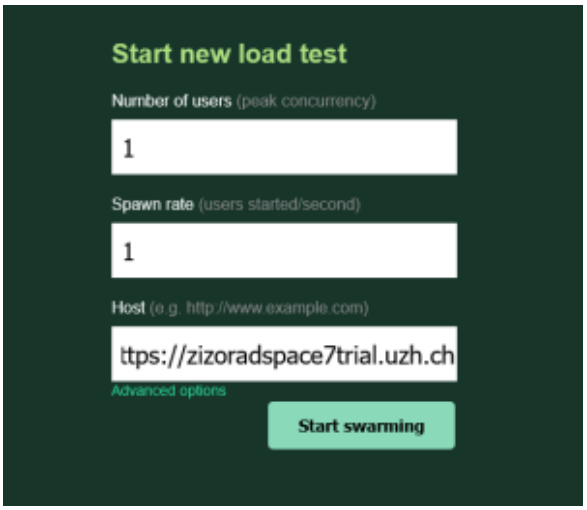
read Command Line Options

Attention: Standard http vs. https; Certs and Keys

```
[locust]$ locust --tls-cert /etc/pki/tls/certs/SERVERNAME/cert.pem --tls-key /etc/pki/tls/certs/SERVERNAME
/privkey.pem

[2024-01-31 14:40:56,153] INFO/locust.main: Starting web interface at https://0.0.0.0:8089

[2024-01-31 14:40:56,166] INFO/locust.main: Starting Locust 2.20.1
```

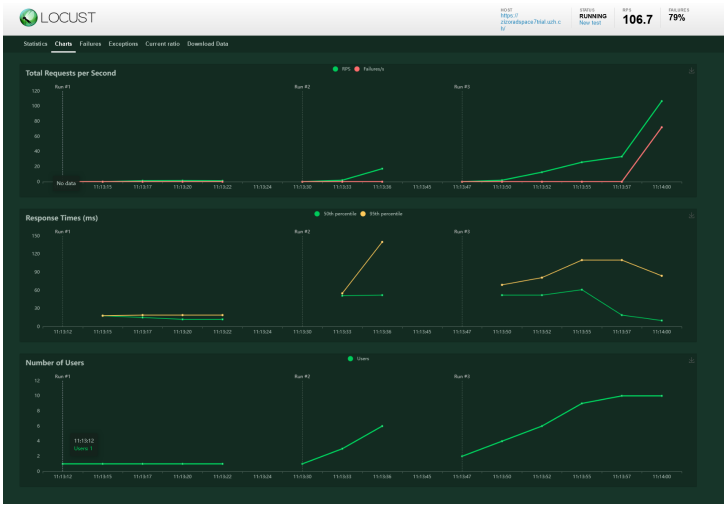
UI

call Browser https://SERVERNAME:8089/ and start loadtests. First easy with 1 user:



blocked URL
(Attention: use trailing slash at the end of 'HOST'!)

See charts, failures, statistics and many more...



blocked URL

02 Real ⚠ test scenarios (static & dynamic content)

- Homepage
- SubPage Communities
- Search
- Login Process incl. Token Autentication

and these scenarios according to the storybook in locustfile.py with

- 1 user
- 10 parallel users
- nnn parallel users

Preliminary work / tests

Before the various authentication procedures for login, session handling (credentials, tokens, cookies, etc.) and workflow steps (dashboard, upload, metadata edit) are run through with try & error, it is advisable to test the individual work steps via the APIs beforehand using CURL.

For e.g. Before login ba POST method, receive TOKEN via GET. Read (insert Link)

Configuration (“complex” Mode)

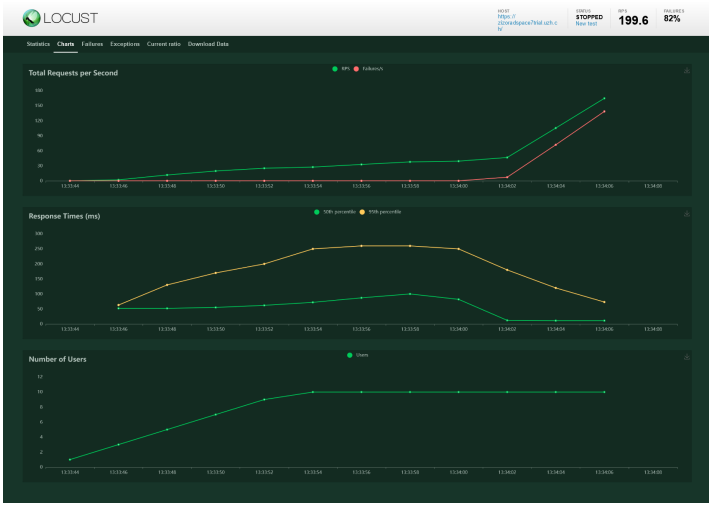
vi locustfile.py

```
# Jens Witzel, Uni Zuerich 2024/01/30

import requests

from locust import HttpUser, task, between


class HelloWorldUser(HttpUser):

    @task

    def browse_homepage(self):

        self.client.get("/")


    @task

    def browse_cummunity(self):

       self.client.get("/communities/dc75f221-6ec0-4ff2-a72b-46d11444daa4")


    @task

    def search_items(self):

       needle = "test"

       self.client.get("/search?q={needle}") # Replace with the actual endpoint for searching items


    @task

    def view_item_details(self):

        item_id = "e0da84e0-0d71-41b6-9aee-17d7b29edca6" # Replace with the actual item ID

        self.client.get(f"/items/{item_id}") # Replace with the actual endpoint for viewing item details


class DSpaceUser(HttpUser):

     wait_time = between(1, 3) # Random wait time between 1 and 3 seconds

     @task def login(self): # get token

     getauthncookies = self.client.get("/server/api/authn", headers={})

     dscookies = dict(getauthncookies.cookies.get_dict())

     csrftoken = dscookies['DSPACE-XSRF-COOKIE']

     # post login with credentials and token

     response = self.client.post("https://SERVER/server/api/authn/login", {"user": USER, "password": PASSWORD},
headers={"X-XSRF-TOKEN":csrftoken}, cookies={"DSPACE-XSRF-COOKIE":csrftoken})

     if response.status_code == 200:

      print("Login successful")

     else:

      print(f"Login failed with status code: {response.status_code}")
```

Notes

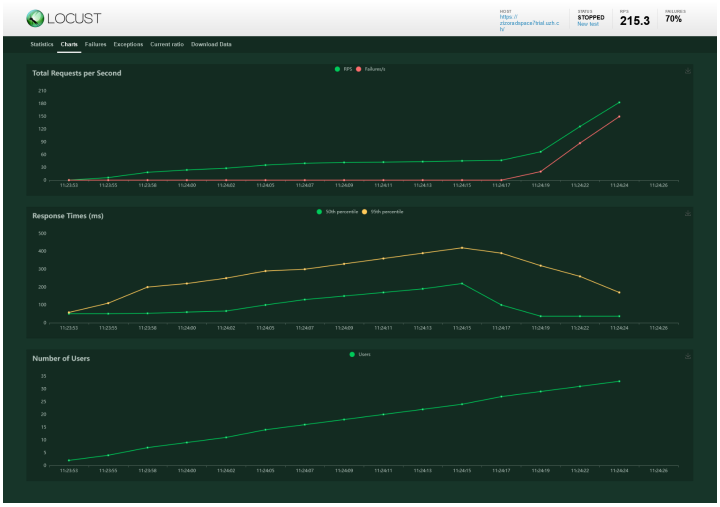
- 2 types of users (anonymous, registered user), login with token handling
- further, sequential storybook possible: login, workflow, upload, edit metadata, logout

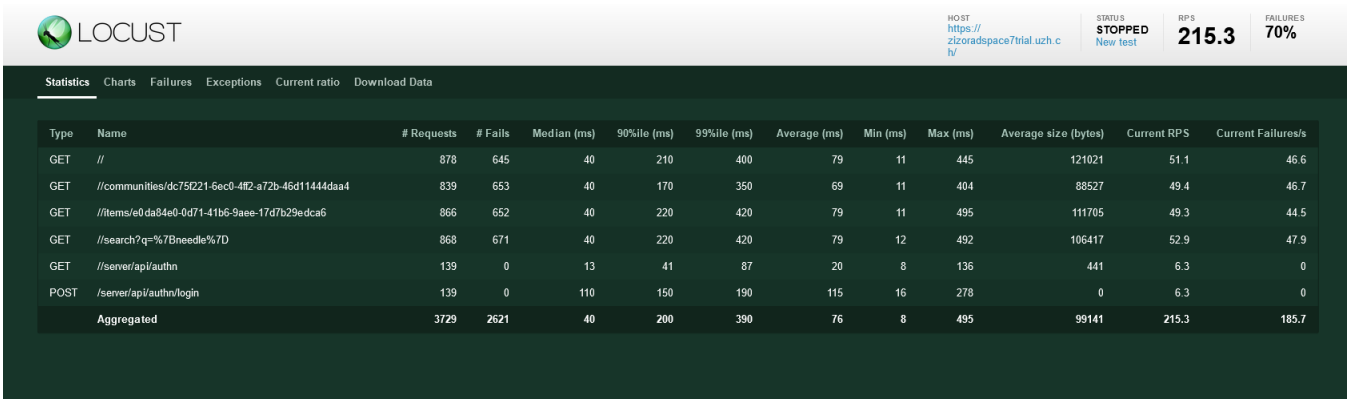
Grafic Output (10 User)



Grafic Output (100 User)

Error rate increases as soon as system is overloaded





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	//	878	645	40	210	400	79	11	445	121021	51.1	46.6
GET	//communities/dc75f221-6ec0-4ff2-a72b-46d11444daa4	839	653	40	170	350	69	11	404	88527	49.4	46.7
GET	//items/e0da84e0-0d71-41b6-9aee-17d7b29edca6	866	652	40	220	420	79	11	495	111705	49.3	44.5
GET	//search?q=%7Bneedle%7D	868	671	40	220	420	79	12	492	106417	52.9	47.9
GET	//server/api/authn	139	0	13	41	87	20	8	136	441	6.3	0
POST	/server/api/authn/login	139	0	110	150	190	115	16	278	0	6.3	0
	Aggregated	**3729**	**2621**	**40**	**200**	**390**	**76**	**8**	**495**	**99141**	**215.3**	**185.7**

03 Really hard test scenarios: Recording workflow steps in FireFox to generate script for Locust

Creating a Locust configuration file from scratch is sometimes difficult, especially when complex workflows and work steps need to be analysed.

One solution is to record typical work steps in the browser and save them in HAR format (https://en.wikipedia.org/wiki/HAR_(file_format)). This, in turn, can be created with the tool har2locust (https://github.com/SvenskaSpel/har2locust) into locustfile.py configuration files and an analysis as described above is ready to go.

The procedure:

- read the notes on https://github.com/SvenskaSpel/har2locust
- [locust]$ pip3.11 install har2locust

Record FF session:

- Open DS7 page in browser
- Menu / Tools for developers | right mouse button + Examine
- Network analysis tab, click on trash basket symbol (delete) - from then on the recording of the activities runs
- Execute DS7 actions (Login, MyDSpace, ... Logout)
- In the end goto column "File", click right mouse button, "Save as Har-File" and save the recording
- sftp Har-File => SERVER
- [locust]$ har2locust x1.har > locustfile.py
- [locust]$ locust --tls-cert /etc/pki/tls/certs/SERVERNAME/cert.pem --tls-key /etc/pki/tls/certs/SERVERNAME/privkey.pem

Links

- https://locust.io/
- https://docs.locust.io/en/stable/
- https://docs.locust.io/en/stable/writing-a-locustfile.html#writing-a-locustfile
- https://medium.com/dana-engineering/load-testing-at-web-application-api-using-locust-2b297c5771ab
- https://www.blazemeter.com/blog/locust-python