



FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Programa de Pós-Graduação Informática Aplicada - PPGIA

Doutorado em Ciência de Dados e Inteligência Artificial

Disciplina: M907 - Sistemas Distribuídos | Professor Dr. Nabor Mendonça
Antonio Marcos Aires Barbosa | Matrícula: 2016397

TRABALHO 03:

- Realização de Testes de Carga com Múltiplas Instâncias do Wordpress Utilizando o Locust

Introdução:

O objetivo do trabalho é realizar testes de carga com múltiplas instâncias do Wordpress, tal como configuradas no Trabalho 2. Para isso, você deverá instalar e configurar o Locust, uma ferramenta para geração de carga para aplicações web.

Atividade:

Realizar testes de carga utilizando o gerador de carga Locust para avaliar o desempenho de diversos cenários de uso do Wordpress, variando a arquitetura da aplicação (número de instâncias do Wordpress) e variando a quantidade de usuários gerados pelo Locust. O contêiner do Locust deve ser definido e adicionado ao docker-compose criado no Trabalho 2.

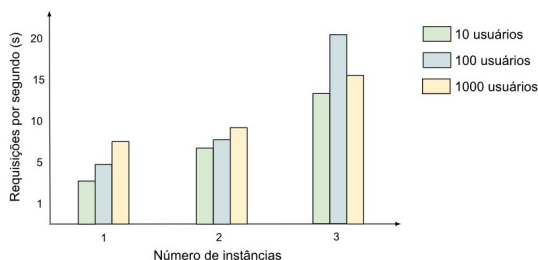
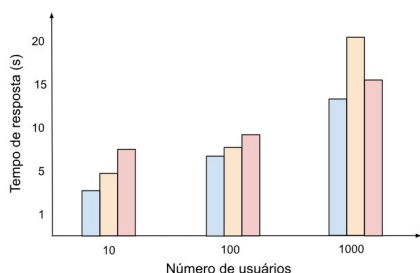
Cenários de teste:

- Blog post com uma imagem de aproximadamente 1mb;
- Blog post com um texto de aproximadamente 400kb;
- Blog post com uma imagem de 300kb;

Entregáveis:

Cada cenário de teste deve ser executado com pelo menos três crescentes números de usuários gerados pelo Locust (por exemplo, 10, 100, e 1000), e pelo menos três crescentes números de instâncias do WordPress (por exemplo, 1, 2, e 3). Os resultados de cada cenário, representados pelos valores das métricas coletadas pelo Locust (por exemplo, tempo de resposta, requisições por segundo, etc.) deverão ser visualizados na forma de gráficos, com o número de usuários ou a quantidade de instâncias do WordPress representados no eixo X, e os valores das métricas no eixo Y.

As figuras abaixo ilustram os estilos de gráficos que deverão ser entregues.



Respostas

Arquivos de Configuração

nginx.conf

```
In [ ]: events { worker_connections 1024; }

http {
    upstream wordpress { # configura um pool de endereço de servidores
        server wordpress1;
        server wordpress2;
        server wordpress3;
    }

    server { # configura esse servidor
        listen 80 default_server; # escutando por conexões na porta 80
        listen [::]:80 default_server;
        root /usr/share/nginx/html;
        index index.php;
        location / {
            add_header X-Upstream $upstream_addr;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header x-forwarded-for $proxy_add_x_forwarded_for;
            proxy_pass http://wordpress;
        }
    }
}
```

docker-compose.yml

```
In [ ]: version: '3'

services:
  nginx:
    image: nginx:1.19.0
    ports:
      - 80:80
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - wordpress1
      - wordpress2
      - wordpress3

  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: r00t
      MYSQL_DATABASE: wordpress
    volumes:
      - ./mysql-data:/var/lib/mysql

  wordpress1:
    image: wordpress:5.4.2-php7.2-apache
    depends_on:
      - mysql
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: r00t
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - ./wordpress:/var/www/html

  wordpress2:
    image: wordpress:5.4.2-php7.2-apache
    depends_on:
      - mysql
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: r00t
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - ./wordpress:/var/www/html

  wordpress3:
    image: wordpress:5.4.2-php7.2-apache
    depends_on:
```

```

- mysql
environment:
  WORDPRESS_DB_HOST: mysql
  WORDPRESS_DB_USER: root
  WORDPRESS_DB_PASSWORD: r00t
  WORDPRESS_DB_NAME: wordpress
volumes:
  - ./wordpress:/var/www/html

prometheus:
  image: prom/prometheus
  volumes:
  - ./prometheus.yml:/etc/prometheus/prometheus.yml
  command:
  - '--config.file=/etc/prometheus/prometheus.yml'
  ports:
  - 9090:9090

node-exporter:
  image: prom/node-exporter
  volumes:
  - /proc:/host/proc:ro
  - /sys:/host/sys:ro
  - /:/rootfs:ro
  command:
  - '--path.procfs=/host/proc'
  - '--path.sysfs=/host/sys'
  - '--path.rootfs=/rootfs'
  - '--collector.filesystem.ignored-mount-points=^/(sys|proc|dev|host|etc)($$|/)'

locust:
  image: locustio/locust
  volumes:
  - ./locustfile.py:/mnt/locust/locustfile.py
  command: -f /mnt/locust/locustfile.py --host=http://nginx
  ports:
  - 8089:8089

locust-exporter:
  image: containersol/locust_exporter
  environment:
  - LOCUST_MASTER_HOST=http://locust:8089
  ports:
  - 9646:9646



























grafana:
  image: grafana/grafana
  ports:
  - 3000:3000
  volumes:
  - grafana-storage:/var/lib/grafana
  depends_on:
  - prometheus
volumes:
  grafana-storage:

```

Rodar containers do Docker

```
docker-compose up -d
```

Containers funcionando

<input type="checkbox"/>	 sistdist		Running (10/10)
<input type="checkbox"/>	 locust-exporter-1 ab79628f0ded 	containersol/locust_exporter	Running 9646:9646 
<input type="checkbox"/>	 prometheus-1 02914272670a 	prom/prometheus	Running 9090:9090 
<input type="checkbox"/>	 locust-1 eda12a34613e 	locustio/locust	Running 8089:8089 
<input type="checkbox"/>	 wordpress1-1 4323c70571a6 	wordpress:5.4.2-php7.2-apache	Running
<input type="checkbox"/>	 wordpress3-1 309e79130775 	wordpress:5.4.2-php7.2-apache	Running
<input type="checkbox"/>	 wordpress2-1 33b917ca3c2d 	wordpress:5.4.2-php7.2-apache	Running
<input type="checkbox"/>	 node-exporter-1 f163f473dce2 	prom/node-exporter	Running
<input type="checkbox"/>	 nginx-1 d819c6c04fa1 	nginx:1.19.0	Running 80:80 
<input type="checkbox"/>	 grafana-1 ad0e6c8bb3c3 	grafana/grafana	Running 3000:3000 
<input type="checkbox"/>	 mysql-1 78112143c487 	mysql:5.7	Running

Criar os posts no Wordpress:

`http://localhost/wp-login.php`

Criar o arquivo de configuração dos testes de carga:

locustfile.py

```
In [1]: !locust --version
```

```
locust 2.15.1 from C:\Users\marco\AppData\Roaming\Python\Python39\site-packages\locust (python 3.9.16)
```

```
In [ ]: %%writefile locustfile.py
from locust import HttpUser, TaskSet, task, between

class UserBehavior(TaskSet):
    host = "http://nginx" # O endereço do servidor Nginx a ser testado
    wait_time = between(5, 15) # O tempo de espera entre as tarefas para cada usuário virtual é um valor aleatório entre

    @task
    def index(self):
        self.client.get("/")

    @task
    def blog_post_with_large_image(self):
        self.client.get("/2023/06/17/post1/")

    @task
    def blog_post_with_text(self):
        self.client.get("/2023/06/15/post2/")

    @task
    def blog_post_with_small_image(self):
        self.client.get("/2023/06/17/post3/")

class WebsiteUser(HttpUser):
    tasks = [UserBehavior]
    wait_time = between(5, 15)
```

```
!locust -f locustfile.py --host=http://172.28.224.1:80
```

Plotar os resultados colhidos pelo Locust

Função para plotar agrupado por quantidade de usuários de teste

```
In [2]: def plot_loadtest(subfolder):
# !pip install seaborn
from matplotlib.ticker import FuncFormatter
from matplotlib.lines import Line2D
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import glob

# Obtém uma lista de todos os arquivos CSV na pasta atual
# csv_files = glob.glob('teste_carga\output_u_*_stats.csv')
csv_files = glob.glob('teste_carga\\'+subfolder+'\\output_u_*_i_*stats.csv')
print(f'Lendo arquivos da {subfolder}')

# Cria um DataFrame vazio para armazenar todos os dados
all_data = pd.DataFrame()

# Itera por todos os arquivos CSV
for file in csv_files:
    # Lê o arquivo CSV em um DataFrame
    df = pd.read_csv(file)

    # Adiciona colunas para o número de usuários e instâncias do WordPress
    df['users'] = int(file.split('_')[-4])
    df['instances'] = int(file.split('_')[-2][0])

    # Adiciona os dados ao DataFrame principal
    all_data = pd.concat([all_data, df])

# Ordena por quantidade de usuários e de instâncias
all_data.sort_values(['users', 'instances'], inplace=True)

# Agrupa os dados pelo número de usuários e instâncias do WordPress e calcula a mediana
grouped = all_data.groupby(['users', 'instances'])["Median Response Time"].median().reset_index()
sizes = all_data.groupby(['users', 'instances'])["Average Content Size"].median().reset_index()
requests = all_data.groupby(['users', 'instances'])["Requests/s"].median().reset_index()
failures = all_data.groupby(['users', 'instances'])["Failures/s"].median().reset_index()
```

```

# Suponha que 'all_data' é o seu DataFrame
all_data['Requests/s'] = pd.to_numeric(all_data['Requests/s'], errors='coerce')
all_data['Failures/s'] = pd.to_numeric(all_data['Failures/s'], errors='coerce')
all_data['users'] = pd.to_numeric(all_data['users'], errors='coerce')

fig, ax = plt.subplots(figsize=(12, 5)) # Definindo o tamanho da figura

# Plotagem das barras agrupadas
sns.barplot(data=all_data, x='users', y='Median Response Time', hue='instances', ax=ax)

# Adicionar rótulos de dados
for container in ax.containers:
    for bar in container:
        bar_height = bar.get_height()
        if np.isfinite(bar_height):
            ax.text(
                bar.get_x() + bar.get_width() / 2,
                bar_height,
                round(bar_height, 2),
                ha='center',
                va='bottom'
            )

# Definir posições dos ticks e rótulos no eixo x
x_positions = np.arange(len(all_data['users'].unique()))
ax.set_xticks(x_positions)
ax.set_xticklabels(all_data['users'].unique())
# ax.set_ylabel('Median Response Time (ms)')
ax.set_ylabel('Mediana do Tempo de resposta (ms)')

# Agrupar os dados por 'users' e calcular as médias de 'Requests/s' e 'Failures/s'
grouped_lines = all_data.groupby('users')[['Requests/s', 'Failures/s']].mean()

# Plotagem das linhas no eixo secundário
ax2 = ax.twinx()
requests_line = ax2.plot(x_positions, grouped_lines['Requests/s'], color='blue', marker='o', label='Requests/s')
failures_line = ax2.plot(x_positions, grouped_lines['Failures/s'], color='red', marker='o', label='Failures/s')

# Configurar rótulos e Legendas do eixo secundário
# ax2.set_ylabel('Requests/s and Failures/s')
ax2.set_ylabel('Taxas de Requisições/s e Falhas/s')
lines = requests_line + failures_line
labels = [line.get_label() for line in lines]
ax2.legend(lines, labels, loc='center left')

plt.title("Taxas de falhas por medianas de tempo de resposta em função do número de usuários e instâncias")
plt.show()

```

Função para plotar agrupado por quantidade de instâncias

```

In [5]: def plot_loadtest_instancias(subfolder):
# !pip install seaborn
from matplotlib.ticker import FuncFormatter
from matplotlib.lines import Line2D
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import glob

# Obtém uma lista de todos os arquivos CSV na pasta atual
# csv_files = glob.glob('teste_carga\output_u_*_stats.csv')
csv_files = glob.glob('teste_carga\\'+subfolder+'\\output_u_*_i_*stats.csv')
print(f'Lendo arquivos da {subfolder}')

# Cria um DataFrame vazio para armazenar todos os dados
all_data = pd.DataFrame()

# Itera por todos os arquivos CSV
for file in csv_files:
    # Lê o arquivo CSV em um DataFrame
    df = pd.read_csv(file)

    # Adiciona colunas para o número de usuários e instâncias do WordPress
    df['users'] = int(file.split('_')[4])
    df['instances'] = int(file.split('_')[-2][0])

    # Adiciona os dados ao DataFrame principal
    all_data = pd.concat([all_data, df])

# Ordena por quantidade de usuários e de instâncias
all_data.sort_values(['users', 'instances'], inplace=True)

# Agrupa os dados pelo número de usuários e instâncias do WordPress e calcula a mediana
grouped = all_data.groupby(['users', 'instances'])["Median Response Time"].median().reset_index()
sizes = all_data.groupby(['users', 'instances'])["Average Content Size"].median().reset_index()
requests = all_data.groupby(['users', 'instances'])["Requests/s"].median().reset_index()
failures = all_data.groupby(['users', 'instances'])["Failures/s"].median().reset_index()
# Suponha que 'all_data' é o seu DataFrame
all_data['Requests/s'] = pd.to_numeric(all_data['Requests/s'], errors='coerce')

```

```

all_data['Falures/s'] = pd.to_numeric(all_data['Falures/s'], errors='coerce')
all_data['users'] = pd.to_numeric(all_data['users'], errors='coerce')

fig, ax = plt.subplots(figsize=(12, 5)) # Definindo o tamanho da figura

# Plotagem das barras agrupadas
sns.barplot(data=all_data, x='instances', y='Median Response Time', hue='users', ax=ax)

# Adicionar rótulos de dados
for container in ax.containers:
    for bar in container:
        bar_height = bar.get_height()
        if np.isfinite(bar_height):
            ax.text(
                bar.get_x() + bar.get_width() / 2,
                bar_height,
                round(bar_height, 2),
                ha='center',
                va='bottom'
            )

# Definir posições dos ticks e rótulos no eixo x
x_positions = np.arange(len(all_data['users'].unique()))
ax.set_xticks(x_positions)
ax.set_xticklabels(all_data['users'].unique())
ax.set_ylabel('Mediana do Tempo de resposta (ms)')

# Agrupar os dados por 'users' e calcular as médias de 'Requests/s' e 'Falures/s'
grouped_lines = all_data.groupby('users')[['Requests/s', 'Falures/s']].mean()

# Plotagem das linhas no eixo secundário
ax2 = ax.twinx()
requests_line = ax2.plot(x_positions, grouped_lines['Requests/s'], color='blue', marker='o', label='Requests/s')
failures_line = ax2.plot(x_positions, grouped_lines['Falures/s'], color='red', marker='o', label='Falures/s')

# Configurar rótulos e Legendas do eixo secundário
ax2.set_ylabel('Taxas de Requisições/s e Falhas/s')
lines = requests_line + failures_line
labels = [line.get_label() for line in lines]
ax2.legend(lines, labels, loc='center left')

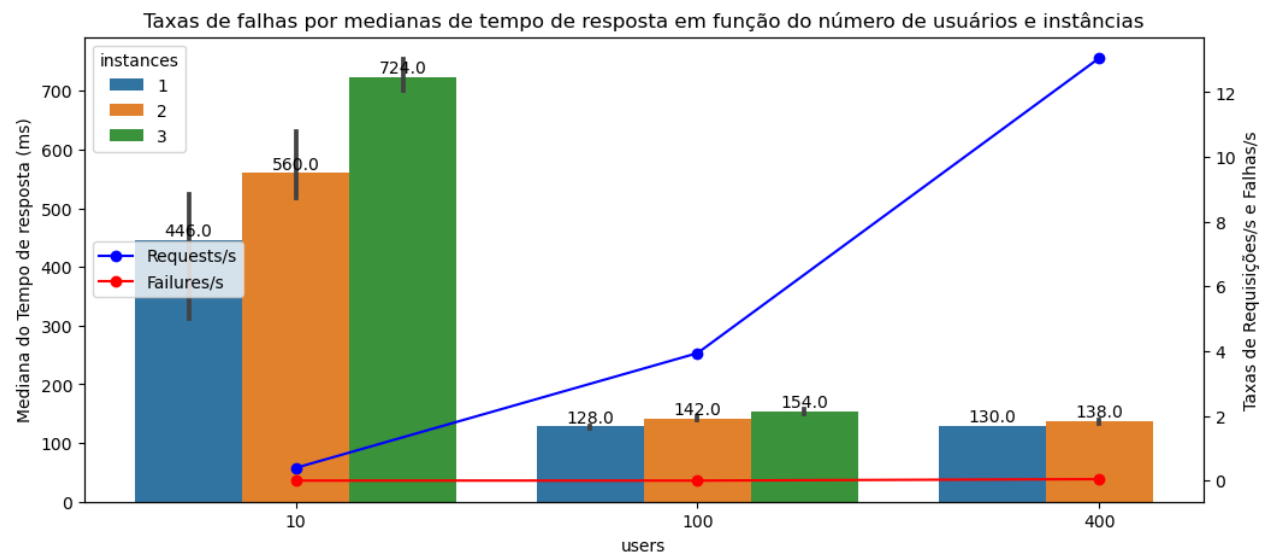
plt.title("Taxas de falhas por medianas de tempo de resposta em função do número de usuários e instâncias")
plt.xlabel("Número de usuários")
plt.show()

```

03 instâncias - Bateria de testes 01: 10, 100, 400 usuários

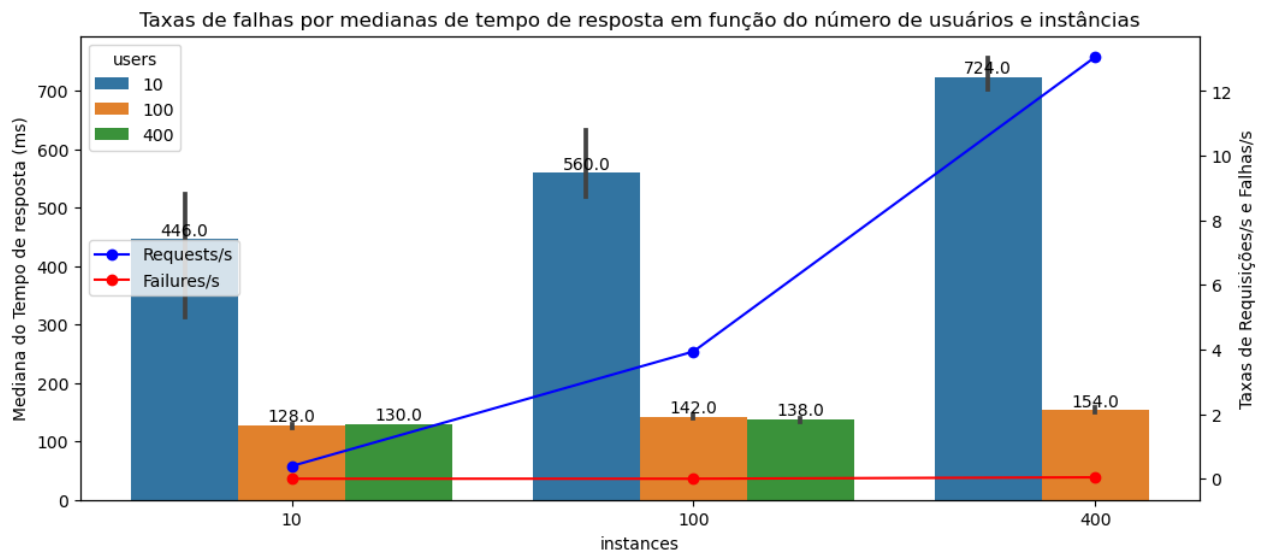
In [3]: `plot_loadtest('bateria01')`

Lendo arquivos da bateria01



In [6]: `plot_loadtest_instancias('bateria01')`

Lendo arquivos da bateria01

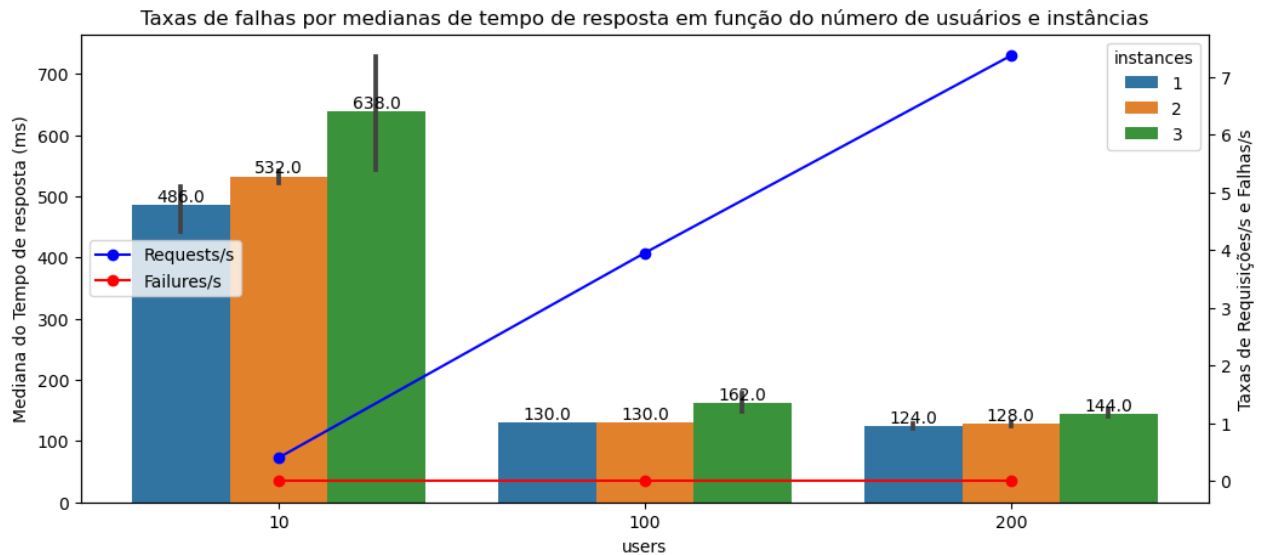


Aqui o teste foi interrompido pela máquina reiniciando, devido a falha de disco local

03 instâncias - Bateria de testes 02: 10, 100, 200 usuários

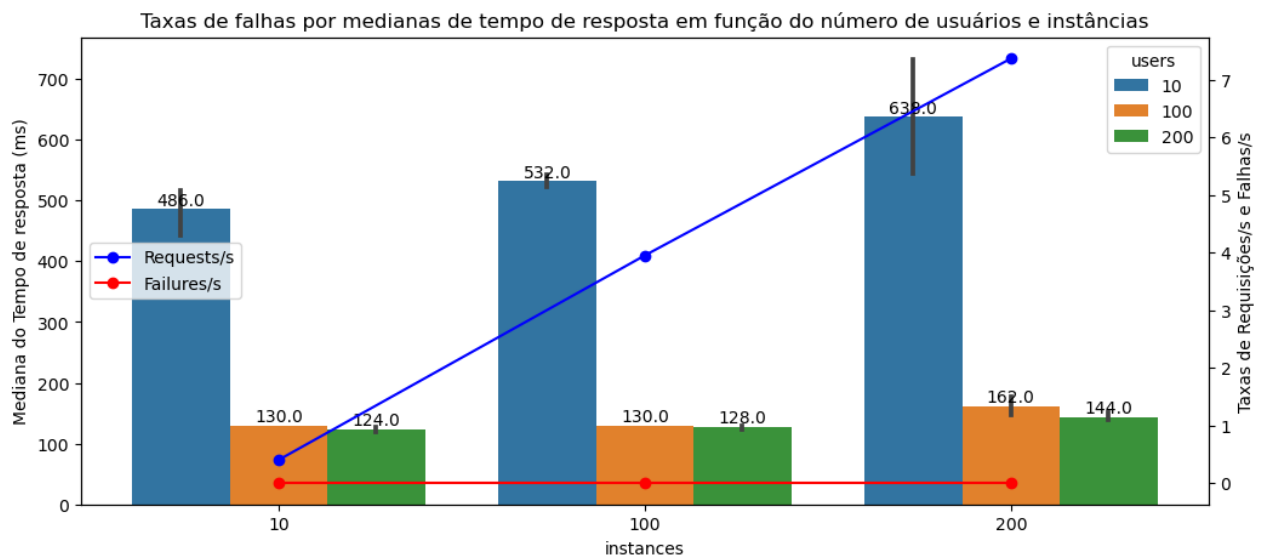
```
In [4]: plot_loadtest('bateria02')
```

Lendo arquivos da bateria02



```
In [7]: plot_loadtest_instancias('bateria02')
```

Lendo arquivos da bateria02



FaseExploratória

```
In [ ]: # !pip install seaborn
        from matplotlib.ticker import FuncFormatter
        from matplotlib.lines import Line2D
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd
        import numpy as np
        import glob

        # Obtém uma lista de todos os arquivos CSV na pasta atual
        # csv_files = glob.glob('teste_carga\output_u_*_stats.csv')
        csv_files = glob.glob('teste_carga\bateria01\output_u_*_i_*stats.csv')

In [ ]: len(csv_files)

In [ ]: list(csv_files)

In [ ]: csv_files[0].split('_')

In [ ]: # file.split('_')[-4]

In [ ]: # file.split('_')[-2][0]

In [ ]: # !pip install seaborn
        from matplotlib.ticker import FuncFormatter
        from matplotlib.lines import Line2D
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd
        import numpy as np
        import glob

        # Obtém uma lista de todos os arquivos CSV na pasta atual
        # csv_files = glob.glob('teste_carga\output_u_*_stats.csv')
        csv_files = glob.glob('teste_carga\output_u_*_i_*stats.csv')

        # Cria um DataFrame vazio para armazenar todos os dados
        all_data = pd.DataFrame()

        # Itera por todos os arquivos CSV
        for file in csv_files:
            # Lê o arquivo CSV em um DataFrame
            df = pd.read_csv(file)

            # Adiciona colunas para o número de usuários e instâncias do WordPress
            df['users'] = int(file.split('_')[-4])
            df['instances'] = int(file.split('_')[-2][0])

            # Adiciona os dados ao DataFrame principal
            all_data = pd.concat([all_data, df])

        # Ordena por quantidade de usuários e de instâncias
        all_data.sort_values(['users', 'instances'], inplace=True)

        # Agrupa os dados pelo número de usuários e instâncias do WordPress e calcula a mediana
        grouped = all_data.groupby(['users', 'instances'])["Median Response Time"].median().reset_index()
        sizes = all_data.groupby(['users', 'instances'])["Average Content Size"].median().reset_index()
        requests = all_data.groupby(['users', 'instances'])["Requests/s"].median().reset_index()
        failures = all_data.groupby(['users', 'instances'])["Failures/s"].median().reset_index()

In [ ]: all_data.iloc[:,0:11]

In [ ]: all_data.iloc[:,12:]

In [ ]: grouped

In [ ]: sizes

In [ ]: requests

In [ ]: failures

In [ ]: all_data

In [ ]: all_data.keys()
```