Locust

==========

INTRODUCTION

➢ **Locust:**

Locust is an open-source load testing tool written in Python. Its purpose is to load test web applications, APIs and any server. Locust is used to simulate an application being used by multiple users on a website to analyse the performance and scalability of the application.

➢ **Overview**:

This document provides a comprehensive analysis of the performance testing conducted on the application using Locust. The test aimed to evaluate the performance metrics including response times, throughput, and error rates to assess the scalability and stability of the application.

INSTALLATION

➢ **Installation of Python and Pip:**

Locust is written in Python, so first you'll need to install Python and Pip.

```
$ sudo apt update
$ sudo apt install python3 python3-pip
```

➢ **Locust Installation:**

Install Locust via pip. Run these commands in terminal:

```
$ pip3 install locust
```

➢ **Verify the version of Locust:**

After the installation is complete, verify whether Locust is installed correctly. Run these commands in terminal:

```
$ locust --version
```

(Output)

```
locust 2.24.1 from /home/user/.local/lib/python3.10/site-packages/locust
(python 3.10.12)
```

➢ **Update PATH:**

You can include the /home/user/.local/bin directory in your PATH. Run these commands in terminal:

```
$ export PATH="$PATH:/home/user/.local/bin"
```

To test APIs using Locust, follow these steps:

1. **Create Locust File:**

   Begin by creating a Locust file. This is a Python script where you define your test cases. Define how your users will behave, what requests they will send, and how they will handle responses.

2. **Define Test Cases:**

   In your Locust file, define the test cases you want to execute. You can send requests, parse responses, and collect performance metrics.

3. **Run Locust:**

   Next, run Locust with your defined test cases. Locust provides a web interface where you can see how your users behave and view performance metrics.

4. **Analyse Results:**

   Use Locust's web interface to analyse performance metrics such as requests per second, response times, and errors. Analysing these metrics helps evaluate the performance of your APIs.

Locust Script for API Testing

➢ locust_Beneficiary.py

```
from locust import HttpUser, between, task
import time

class MyUser(HttpUser):
    host = "http://mtainternal.finobank.keenable.in"
    wait_time = between(0.5, 1.5)


    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.start_time = time.time()

    def run_time_elapsed(self):
        return time.time() - self.start_time

    def run(self):
        while not self.should_stop():
            self.test_beneficiary_add_api()
            self.test_beneficiary_list_api()
            self.test_beneficiary_search_api()
            self.test_beneficiary_delete_api()
            self.test_beneficiary_limit_tmp_api()
            time.sleep(1)

    def should_stop(self):
        return self.run_time_elapsed() >= 300

    @task
    def test_beneficiary_add_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "CustomerAuth": {
                "Auth_id": "0",
                "MobileNo": "7722846604",
                "Otp": {},
                "Fp_Auth": {},
                "Aadhaar": {}
```

```
            },
            "AddBeneficiary": {
                "appId": "FINOMER",
                "customerNumber": "153155813",
                "beneficiaryType": "2",
                "nickName": "",
                "beneficiaryAccount": "20323831263",
                "beneficiaryAccountType": "10",
                "beneficiaryName": "Upendra Singh Dhakar",
                "beneficiaryBankIfsc": "",
                "beneficiaryAddress1": "",
                "beneficiaryZip": "",
                "beneficiaryEmailId": "",
                "beneficiaryBankMicr": None,
                "beneficiaryMobileNumber": "9691877503",
                "beneficiaryMaxLimit": "",
                "beneficiaryBankCity": "",
                "beneficiaryBankBranch": "",
                "beneficiaryState": "",
                "beneficiaryCity": ""
            }
        }
        response = self.client.post("/mta/beneficiary/add",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_beneficiary_list_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "customerId": "153128895"
        }
        response = self.client.post("/mta/beneficiary/list",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_beneficiary_search_api(self):
```

```
        headers = {'Content-Type': 'application/json'}
        payload = {
            "beneficiaryId": "153128895"
        }
        response = self.client.post("/mta/beneficiary/search",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_beneficiary_delete_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "appId": "MB",
            "customerNumber": "135992641",
            "beneficiaryType": "3",
            "beneficiarySequence": "2",
            "beneName": "Jameel"
        }
        response = self.client.post("/mta/beneficiary/delete",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_beneficiary_limit_tmp_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {"beneficiaryId": "123456789"}

        response = self.client.post("/mta/beneficiary/limitTmp",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)
```

➢ locustfile_Walkin.py

```
from locust import HttpUser, between, task
import time

class MyUser(HttpUser):
    host = "http://mtainternal.finobank.keenable.in"
    wait_time = between(0.5, 1.5)


    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.start_time = time.time()

    def run_time_elapsed(self):
        return time.time() - self.start_time

    def run(self):
        while not self.should_stop():
            self.test_walkin_add_api()
            self.test_walkin_update_api()
            self.test_walkin_search_api()
            self.test_walkin_ministatement_api()
            self.test_walkin_limit_api()
            self.test_walkin_reversal_api()
            time.sleep(1)

    def should_stop(self):
        return self.run_time_elapsed() >= 300


    @task
    def test_walkin_add_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "mobileNumber": "9999888877",
            "name": "Jay",
            "address": "Jaipur",
            "dateofBirth": "15-01-1999",
            "gender": "M",
            "idProofType": "AadhaarCard",
            "idProofNumber": "123412341234",
            "addressProofType": "UtilityBill",
```

```
            "addressProofNumber": "UB987654321",
            "customerNumber": "123456789012",
            "walkinStatus": "Active",
            "lastUpdateDate": "2024-01-11",
            "walkinRefNum": "WALKIN7890123"
        }
        response = self.client.post("/mta/walkin/add", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_walkin_update_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "customerNumber": "123456789012",
            "updateFields": {
                "name": "Akash",
                "address": "Ballabgarh",
                "dateOfBirth": "15-01-1999",
                "gender": "Male",
                "idProofType": "DriverLicense",
                "idProofNumber": "DL123456789",
                "addressProofType": "UtilityBill",
                "addressProofNumber": "UB987654321"
            }
        }

        response = self.client.post("/mta/walkin/update", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_walkin_search_api(self):
        headers = {'Content-Type': 'application/json'}
        params = {'mobileNo': '20186929191', 'name': ''}
        response = self.client.get("/mta/walkin/search", params=params,
headers=headers)
```

```
        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_walkin_ministatement_api(self):
        headers = {'Content-Type': 'application/json'}
        params = {
            "mobileNumber": "8250978340"
        }

        response = self.client.get("/mta/walkin/ministatement",
params=params, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_walkin_limit_api(self):
        headers = {'Content-Type': 'application/json'}
        params = {
            "mobileNumber": "9999888877",
            "trangroup": "DMTREMIT"
        }

        response = self.client.get("/mta/walkin/limit", params=params,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_walkin_reversal_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "appId": "FINOMER",
            "valueDate": None,
            "isInclusive": 0,
            "isClubbed": 0,
            "analysisFlag": None,
            "reversalFlag": None,
```

```
            "referenceNo": "402218075148",
            "reversalFlag": "F"
        }

        response = self.client.post("/mta/walkin/reversal",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)
```

➢ locustfile_Merchant-Management.py

```
from locust import HttpUser, between, task
import time

class MyUser(HttpUser):
    host = "http://mtainternal.finobank.keenable.in"
    wait_time = between(0.5, 1.5)


    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.start_time = time.time()

    def run_time_elapsed(self):
        return time.time() - self.start_time

    def run(self):
        while not self.should_stop():
            self.test_check_duplicate_api()
            self.test_get_null_gl_api()
            self.test_get_effective_balance_api()
            self.test_create_account_api()
            self.test_update_details_api()
            self.test_account_link_api()
            self.test_holdlien_add_api()
```

```
            self.test_holdlien_remove_api()
            self.test_get_charges_api()
            self.test_get_merchant_details_api()
            time.sleep(1)

    def should_stop(self):
        return self.run_time_elapsed() >= 300


    @task
    def test_check_duplicate_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "referenceNo": "403208103911",
            "reversalFlag": 0
        }

        response = self.client.post("/mta/checkduplicate", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_get_null_gl_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "singleCodeRequest": {
                "flagDebitCredit": "C",
                "tranType": "DMTIMPS2A",
                "appId": "FINOTLR",
                "userClass": "SYSTEMUSR4"
            }
        }

        response = self.client.post("/mta/getnullgl", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
```

```
    def test_get_effective_balance_api(self):
        headers = {'Content-Type': 'application/json'}
        params = {"accountnumber": "20025784129"}

        response = self.client.get("/mta/merchant/getEffectivebalance",
params=params, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_create_account_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "12345678901",
            "accountType": "Savings",
            "customerName": "SANJAY",
            "panNumber": "EBWTT2009A",
            "address": "Main Street",
            "phoneNumber": "1234567890",
            "gstin": "22AAAAA0000A1Z5",
            "cifNumber": "12345678901",
            "customerNumber": "1234567890",
            "branchCode": "11223378987"
        }

        response = self.client.post("/mta/merchant/createAccount",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_update_details_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "12345678901",
            "updateDetails": {
                "panNumber": "EBWTT2009A",
                "address": "Main Street",
                "phoneNumber": "9876543210",
                "gstin": "22AAAAA0000A1Z5",
```

```
                "branchCode": "11223378987",
                "cifNumber": "12345678901"
            }
        }

        response = self.client.post("/mta/merchant/updateDetails",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)


    @task
    def test_account_link_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "12345678901",
            "linkUserIdDetails": {
                "userID": "Mer11"
            }
        }

        response = self.client.post("/mta/merchant/accountLink",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_holdlien_add_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "12345678901",
            "lienDetails": {
                "amount": 5000,
                "holdCodeId": "HOLD456",
                "expiryDate": "21/08/2016",
                "comments": "test3"
            }
        }

        response = self.client.post("/mta/merchant/holdlienAdd",
```

```
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_holdlien_remove_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "12345678901",
            "removeDetails": {
            "holdCodeId": "HOLD456"
        }
        }
        response = self.client.post("/mta/merchant/holdlienRemove",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_get_charges_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "transactionType": "XYZ",
            "amount": 100
        }

        response = self.client.post("/mta/getcharges", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_get_merchant_details_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "20025784129"
        }
```

```
        response = self.client.post("/mta/merchant/getDetails",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)
```

➢ locustfile_external.py

```
from locust import HttpUser, between, task
import time

class MyUser(HttpUser):
    host = "http://mtainternal.finobank.keenable.in"
    wait_time = between(0.5, 1.5)


    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.start_time = time.time()

    def run_time_elapsed(self):
        return time.time() - self.start_time

    def run(self):
        while not self.should_stop():
            self.test_post_commission_cbs_api()
            self.test_transaction_status_api()
            self.test_get_pending_transaction_api()
            self.test_transaction_check_allowed_transaction_api()
            self.test_transaction_posting_api()
            self.test_get_null_gl_api()
            self.test_caas_check_auth_api()
            self.test_neft_status_api()
            self.test_transaction_api()
            self.test_update_rfu_api()
            self.test_check_balance_api()
```

```
            time.sleep(1)

    def should_stop(self):
        return self.run_time_elapsed() >= 300

    @task
    def test_post_commission_cbs_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "12345678901",
            "commissionAmount": 100,
            "transactionType": "commission",
            "postedDate": "2024-02-26"
        }

        response = self.client.post("/mta/merchant/postCommissionCBS",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_transaction_status_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "txnReferenceNo": "334108018698",
            "status": "success"
        }

        response =
self.client.post("/mta/transaction/transactionStatus", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_get_pending_transaction_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "accountNumber": "",
            "mobileNumber": "9999999988"
```

```
        }

        response =
self.client.post("/mta/merchant/getPendingTransaction", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_transaction_check_allowed_transaction_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "userClass": "MER1",
            "applicationId": "FINOMER",
            "transactionType": "DMTIMPS2A",
            "exists": "TRUE"
        }

        response =
self.client.post("/mta/transaction/checkAllowedTransaction",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_transaction_posting_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "uniqueTransactionId": "1234567890",
            "transCategory": "IMPS",
            "appId": "FINOMER",
            "iftFileId": "IFT123",
            "referenceNo": "REF123",
            "reversalFlag": "false",
            "chargeOverride": "false",
            "analysisFlag": "true",
            "isClubbed": "false",
            "isInclusive": "true",
            "valueDate": "2024-03-14",
            "initiatingBranchCode": "12345",
```

```
            "acctFundTransferLegs": [
                {
                    "postingAmount": "1000.00",
                    "accountNumber": "1234567890",
                    "amount": "1000.00",
                    "currency": "USD",
                    "creditDebitFlag": "C"
                },
                {
                    "postingAmount": "100.00",
                    "accountNumber": "9876543210",
                    "amount": "100.00",
                    "currency": "USD",
                    "creditDebitFlag": "D"
                }
            ],
            "chargeLegs": [
                {
                    "accountNumber": "1234567890",
                    "postingAmount": "10.00",
                    "transactionType": "Charge",
                    "creditDebitFlag": "D"
                }
            ],
            "taxLegs": [
                {
                    "postingPair": 1,
                    "taxLegId": "TAX001",
                    "taxType": "VAT",
                    "taxAmount": "20.00",
                    "taxGLNumber": "GL001",
                    "trantype": "TRAN1"
                }
            ]
        }

        response = self.client.post("/mta/transaction/posting",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_get_null_gl_api(self):
```

```
        headers = {'Content-Type': 'application/json'}
        payload = {
            "singleCodeRequest": {
                "flagDebitCredit": "C",
                "tranType": "DMTIMPS2A",
                "appId": "FINOTLR",
                "userClass": "SYSTEMUSR4"
            }
        }

        response = self.client.post("/mta/getnullgl", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_caas_check_auth_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "token": "bXadcrxDyYfoRmoadi66CmJYX6x8eIp2Yxmy2VK3",
            "token_type_hint": "access_token",
            "client_id": "myclient",
            "client_secret": "dfnfgk3awwdcws"
        }

        response = self.client.post("/mta/caas/checkauth", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_neft_status_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "transactionDetails": {
                "transactionId": "TX123",
                "utrNumber": "UTR456"
            }
        }
```

```
        response = self.client.post("/mta/transaction/neftStatus",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_transaction_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "appId": "FINOMER",
            "valueDate": "01/01/2023",
            "isInclusive": 0,
            "isClubbed": 0,
            "analysisFlag": None,
            "reversalFlag": None,
            "referenceNo": "334108018700",
            "acctFundTransferLegs": [
                {
                    "accountNumber": "",
                    "amount": 5000,
                    "currency": "INR",
                    "creditDebitFlag": "D",
                    "transactionType": "DMTIMPSP2A",
                    "transactionComment": "FT IMPS",
                    "costCenter": 120201,
                    "supportData":
"SPROD#1~PVTXNID#334020026355~PVTXNDT#06/12/23~IPADDR#192.168.1.7~DEVICE
ID#c47f5cf77eb015dd~APPID#FINOMER~AUTHID#@authid~LOCATION#~CELLID#10.774
8083,77.4421183~MCC#55e911,99,405,869~RPTFLG#0~PRTXNID#334108018700~PRTX
NAMT#5000~SPLTSEQ#0~CHGAMT#50.0~ZRFUT1#1495108017124~ZRFUT2#CNRB0000404~
ZRFUT3#CANARA BANK~ZRFUT4#~ZRFUT5#RAMATHILAGAM N
V~ZRFUT6#~ZRFUN3#156791337~ZRFUN4#9626651957~NETID#900000~MSGTYP#210~ZRF
UT8#334108018700~ZRFUT9#155881118724_120723082140389",
                    "beneficiaryRefOrMmid": "",
                    "beneficiaryMobile": "",
                    "remitterMobile": "9626651957",
                    "remitterMmid": "",
                    "beneficiaryAccountNo": "1495108017124",
                    "beneficiaryIfsc": "CNRB0000404",
                    "remarks": None,
                    "transactionComment":
"FTIMPS_9626651957_XX7124_334108018700_RAMATHILAGAM "
                },
```

```
                {
                    "accountNumber": None,
                    "amount": 5000,
                    "currency": "INR",
                    "creditDebitFlag": "C",
                    "transactionType": "DMTIMPSP2A",
                    "transactionComment": "FT IMPS",
                    "costCenter": 120201,
                    "supportData": "",
                    "beneficiaryRefOrMmid": None,
                    "beneficiaryMobile": None,
                    "remitterMobile": None,
                    "remitterMmid": None,
                    "beneficiaryAccountNo": None,
                    "beneficiaryIfsc": None,
                    "remarks": None,
                    "transactionComment":
"FTIMPS_9626651957_XX7124_334108018700_RAMATHILAGAM "
                }
            ]
        }

        response = self.client.post("/mta/transaction", json=payload,
headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)

    @task
    def test_update_rfu_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "referenceNo": "402218075148",
            "rfuField": "ZRFUT8",
            "rfuValue": "402218075148"
        }

        response = self.client.post("/mta/transaction/updateRFU",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)
```

```
    @task
    def test_check_balance_api(self):
        headers = {'Content-Type': 'application/json'}
        payload = {
            "glNumber": "123456789",
            "postingAmount": "1000.00",
            "creditDebitFlag": "credit"
        }

        response = self.client.post("/mta/transaction/checkbalance",
json=payload, headers=headers)

        if response.status_code == 200:
            print("Success: ", response.json())
        else:
            print("Failed: ", response.status_code)
```

Running the Tests

- Open a terminal window and navigate to the project directory.
- Execute the following command to start the Locust test:

```
$ locust -f locust_Beneficiary.py
```

```
$ locust -f locustfile_Walkin.py
```

```
$ locust -f locustfile_Merchant-Management.py
```

```
$ locust -f locustfile_external.py
```

- Open a web browser and go to **http://localhost:8089** to access the Locust Web UI.
- Enter the required parameters (number of users, hatch rate, etc.) and start the test.

➢ **Test Results:** locust_Beneficiary.py

During: 2024-04-12 10:35:00 - 10:40:00

Target Host: http://mtainternal.finobank.keenable.in

Number of users (peak concurrency) = 1000

Ramp up (users started/second) = 10

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
POST	/mta/beneficiary/add	37893	0	1600	2200	5300	1454.13	159	11036	80	35.3	0
POST	/mta/beneficiary/delete	37893	0	1600	2100	2300	1358.11	158	2838	82	38.9	0
POST	/mta/beneficiary/limitTmp	37893	0	1600	2100	2300	1367.85	160	3271	113	39.5	0
POST	/mta/beneficiary/list	37893	0	1600	2100	2300	1384.04	158	3557	373	36.4	0
POST	/mta/beneficiary/search	37893	0	1600	2100	2300	1368.22	158	2870	307	37.4	0
Aggregated	189465	0	1600	2100	2300	1386.47	158	11036	191	187.5	0	

➢ **Test Results:** locustfile_Walkin.py

During: 2024-04-12 10:45:00 - 10:50:00

Target Host: http://mtainternal.finobank.keenable.in

Number of users (peak concurrency) = 1000

Ramp up (users started/second) = 10

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
POST	/mta/walkin/add	28940	0	1800	2600	7600	1684.51	159	11767	92	33.5	0
GET	/mta/walkin/limit?mobileNumber=9999888877&trangroup=DMTREMIT	28940	0	1800	2500	2800	1554.97	159	4255	142	39.5	0
GET	/mta/walkin/ministatement?mobileNumber=8250978340	28940	0	1800	2400	2900	1545.19	160	3828	346	37.6	0
POST	/mta/walkin/reversal	28940	0	1800	2500	2800	1542.33	160	4123	188	40.7	0
GET	/mta/walkin/search?mobileNo=20186929191&name=	28940	0	1800	2500	2800	1549.29	159	4577	150	35.8	0
POST	/mta/walkin/update	28940	0	1800	2500	2800	1553.22	159	3414	106	34.2	0
Aggregated	173640	0	1800	2500	2900	1571.59	159	11767	170.67	221.3	0	

➢ **Test Results:** locustfile_Merchant-Management.py

During: 2024-04-12 11:00:00 - 11:05:00

Target Host: http://mtainternal.finobank.keenable.in

Number of users (peak concurrency) = 1000

Ramp up (users started/second) = 10

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
POST	/mta/checkduplicate	17222	0	1900	2800	9700	1902.46	160	12658	71	22.4	0
POST	/mta/getcharges	17222	0	1900	2500	2800	1641	161	3182	91	27.5	0
POST	/mta/getnullgl	17222	0	1900	2500	2800	1655.64	161	3232	49	23.2	0
POST	/mta/merchant/accountLink	17222	0	1900	2500	2800	1643.29	160	3196	76	25.5	0
POST	/mta/merchant/createAccount	17222	0	1900	2500	2800	1640.18	160	3160	87	24.1	0
POST	/mta/merchant/getDetails	17222	0	1900	2500	2800	1640.41	160	3267	268	28.8	0
GET	/mta/merchant/getEffectivebalance?accountnumber=20025784129	17222	0	1900	2500	2800	1635.82	161	3615	996	23.6	0
POST	/mta/merchant/holdlienAdd	17222	0	1900	2500	2800	1640.78	160	3187	75	26	0
POST	/mta/merchant/holdlienRemove	17222	0	1900	2500	2800	1638.98	160	3236	75	26.5	0
POST	/mta/merchant/updateDetails	17222	0	1900	2500	2800	1643.36	160	3189	68	24.7	0
Aggregated	172220	0	1900	2500	2900	1668.19	160	12658	185.6	252.3	0	

➢ **Test Results:** locustfile_external.py

During: 2024-04-12 11:15:00 - 11:20:00

Target Host: http://mtainternal.finobank.keenable.in

Number of users (peak concurrency) = 1000

Ramp up (users started/second) = 10

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
POST	/mta/caas/checkauth	17043	0	1800	2100	2400	1512.92	160	2899	208	24.4	0
POST	/mta/getnullgl	17043	0	1800	2100	2300	1513.88	160	3100	49	24.1	0
POST	/mta/merchant/getPendingTransaction	17043	0	1800	2100	2400	1511.8	161	3337	1359	20.4	0
POST	/mta/merchant/postCommissionCBS	17043	0	1800	2400	8800	1746.49	160	11517	61	19.4	0
POST	/mta/transaction	17043	0	1800	2100	2500	1517.02	162	3506	169	24.8	0
POST	/mta/transaction/checkAllowedTransaction	17043	0	1800	2100	2400	1512.48	161	3161	53	21.3	0
POST	/mta/transaction/checkbalance	17043	0	1800	2100	2300	1506.95	159	3085	53	25.7	0
POST	/mta/transaction/neftStatus	17043	0	1800	2100	2400	1508.22	159	3830	119	24.9	0
POST	/mta/transaction/posting	17043	0	1800	2100	2400	1510.3	161	3048	378	22.7	0
POST	/mta/transaction/transactionStatus	17043	0	1800	2100	2300	1517.46	160	3034	18	20	0
POST	/mta/transaction/updateRFU	17043	0	1800	2100	2400	1511.43	159	3096	55	25.2	0
Aggregated	187473	0	1800	2100	2500	1533.54	159	11517	229.27	252.9	0	

Definitions:

- **Type:** Type of HTTP request (e.g., POST, GET).
- **Name:** Endpoint or URL of the request.
- **# Requests:** Total number of requests sent during the test.
- **# Fails:** Total number of failed requests during the test.
- **Median (ms):** The response time at which 50% of the requests had a faster response time, and 50% had a slower response time.
- **95%ile (ms):** The response time at which 95% of the requests had a faster response time, and 5% had a slower response time.
- **99%ile (ms):** The response time at which 99% of the requests had a faster response time, and 1% had a slower response time.
- **Average (ms):** The average response time across all requests.
- **Min (ms):** The minimum response time observed.
- **Max (ms):** The maximum response time observed.
- **Average size (bytes):** The average size of the response payload in bytes.
- **Current RPS:** Current Requests Per Second, i.e., the rate at which requests are being sent per second.
- **Current Failures/s:** Current Failures Per Second, i.e., the rate at which requests are failing per second.

References:

- Locust Documentation: https://docs.locust.io/en/stable/