Locust Python load testing

CONCEPTS

A locustfile is plain Python: each User class is one simulated visitor running weighted tasks.

Locustfile User + tasks
↓ locust
Master spawns, aggregates
↓ --worker
Workers run the users
↓ HTTP
Target your system

Only load test systems you own or have written permission to test.

Python is single-core per process: use workers (or --processes) to use more CPUs.

RUNNING

locust
Start the web UI (port 8089) using locustfile.py

locust -f *<file>*
Use a specific locustfile

locust -H *<url>*
Target host for all requests

locust --web-port *<port>*
Change the web UI port

locust *<UserClass>*
Run only the named user classes

locust --config *<file>*
Read options from a config file

locust -L DEBUG
Set the log level

Defaults can live in locust.conf, ~/.locust.conf or pyproject.toml ([tool.locust]). Env vars use LOCUST_<OPTION>, flags win over both.

HEADLESS

locust --headless -u *<n>* **-r** *<rate>* **-t** *<time>*
Fixed run without UI

-u / --users *<n>*
Peak number of simulated users

HEADLESS (cont.)

-r / --spawn-rate *<n>*
Users started per second

-t / --run-time *<time>*
Stop after this long

-s / --stop-timeout *<time>*
Let running tasks finish on shutdown

--exit-code-on-error *<code>*
Exit code when requests failed

--reset-stats
Clear stats after ramp-up ends

In headless mode press w / W to add 1 / 10 users and s / S to remove them.

REPORTS

--csv *<prefix>*
Write <prefix>_stats.csv and failures/exceptions CSVs

--csv-full-history
Add a row per interval to _stats_history.csv

--html *<file>*
Write an HTML report at the end

DISTRIBUTED

locust --master
Start the master (no users run on it)

locust --worker
Start a worker, master on localhost

locust --worker --master-host *<host>*
Point a worker at a remote master

--master-bind-port *<port>*
Port the master listens on (5557)

--expect-workers *<n>*
Headless master waits for n workers

locust --processes *<n>*
Fork master plus n workers locally

locust --worker --processes *<n>*
Fork only n workers on this machine

DISTRIBUTED (cont.)

Copy the same locustfile to every worker. Run the master headless with --expect-workers for CI.

LOCUSTFILE

Minimal HttpUser

```
from locust import HttpUser, task, between

class ShopUser(HttpUser):
    wait_time = between(1, 5)

    @task(3)
    def home(self):
        self.client.get("/")

    @task
    def item(self):
        self.client.get("/item/1")
```

@task(*<weight>***)**
Pick this task proportionally more often

wait_time = between(*<a>***, *<b>*)**
Random pause after each task

wait_time = constant(*<s>***)**
Fixed pause after each task

wait_time = constant_pacing(*<s>***)**
Run a task at most once per interval

def on_start(self):
Per-user setup such as login

self.client.get(*<path>***, name=***<label>***)**
Group dynamic URLs in the stats

self.client.post(*<path>***, json=***<obj>***)**
POST a JSON body

catch_response=True
Decide pass/fail from the response

@tag(*<name>***)**
Label a task for --tags filtering

-T *<tag>* **/ -E** *<tag>*
Include / exclude tagged tasks

class *<Name>***(FastHttpUser)**
Faster HTTP client for heavy load

class *<Name>***(LoadTestShape)**
Script the user curve over time