Transform Mainframe Testing with Open Source Tools

Adam Munawar Rahman
Staff Software Developer @ IBM
Medium: @msradam

Marist Computing Conference 2025

What is Load Testing?

Simulating real-world traffic at scale to verify system resilience.

When inadequate testing causes catastrophic failures: [1,2]

- **Ticketmaster (Nov 2022)** - Taylor Swift presale: 3.5B system requests (4x previous peak). Site crashed. Testing hadn't accounted for unprecedented traffic spikes.
- **TSB Bank (Apr 2018)** - £330M cost, 6M customers locked out for weeks. IT migration without full-volume testing in production-like environment. IBM identified insufficient performance testing.

The impact of proper load testing: [3]
Teams using modern load testing practices report **45% latency reduction, 2x throughput gains, and 70% faster response times**.

Load testing reveals capacity limits and bottlenecks **before** they impact millions of customers.

[1] "Why Load Testing is Important: The Taylor Swift Debacle," TESTINGMIND, Dec. 2022
[2] "TSB Bank Data Migration Failure," iceDQ, Jan. 2025
[3] N. Yadav, "Real-World Examples of Performance Testing Failures," Testriq, Sep. 2025

What Are Mainframes?

The backbone of global commerce and critical systems.

30+ billion transactions daily. 90% of all credit card transactions. [1]
92 of the top 100 banks run on IBM mainframes. [1]

The problem? Testing tools frozen in the 1980s.

[1] IBM, "IBM Mainframe Ushers in New Era of Data Protection," Jul. 2017

The Gap

Legacy Tools TPNS (1976) • JMeter (1998) • WSim (2002)	**Modern Tools** Locust (2012) • k6 (2016)
Thread-per-user (1MB+ per thread) Few thousand users max [1,2]	Event-driven: per machine [1,2]
Developer experience: GUI config, proprietary scripting (REXX/STL), XML management, complex setup [3,4]	**Developer experience:** **Tests-as-code** in Python/JavaScript `pip install` or single binary [3,4]
Steep learning curve, slow GUI,	, pipeline-ready

The breakthrough: Modern tools + REST APIs = mainframe accessibility.

[1] N. van der Hoeven, "Comparing k6 and JMeter," Grafana Labs, Feb. 2024
[2] T. Koot, "k6 vs. JMeter - Head2Head," LinkedIn, Oct. 2021
[3] B. Roy, "JMeter vs k6," TestVagrant Medium, Dec. 2022
[4] "JMeter vs. Locust," PFLB, Mar. 2025
[5] S. Mamoru, "Advanced Performance Testing with JMeter," Geek Culture Medium, Jun. 2023
[6] "LoadRunner vs JMeter," BrowserStack, May 2025
[7] "Performance Tools Benchmarking," QAInsights, May 2022

The Bridge: Two Enablers

1. z/OS REST APIs - The modern interface:

z/OSMF • z/OS Connect • Zowe API ML • CICS REST
HTTP requests = mainframe testing

2. Open-source tooling - The connectivity layer: [1,2,3,4]

x3270/c3270/py3270 - 3270 terminal emulation & automation
ZOAU - z/OS automation utilities (Python, shell, Node.js)
Galasa - Integration testing framework (3270, REST, batch, web)
tnz - Python 3270 library for terminal automation

The breakthrough:
Modern tools + modern APIs + open-source bridges = mainframe accessibility

[1] "py3270: Python interface to x3270," IBM GitHub, 2025
[2] "IBM Z Open Automation Utilities," IBM, 2024
[3] "Galasa - Open Source Testing Framework," Open Mainframe Project, 2024
[4] "tnz: Tn3270 to Z Python library," IBM GitHub, 2025

The Solution: Adapt, Don't Reinvent

Two industry-proven tools. Two strategic adaptations.

🐍 **Locust** (Python) - Used by EA/DICE, AWS, Learnosity [1]
→ Extended with py3270 for 3270 terminal testing

⚡ **k6** (Go) - Used by GitLab, Carvana, Olo [1]
→ Ported to run natively on z/OS

The breakthrough: Modern tools already exist.
We just made them work on mainframes.

[1] "k6 Testimonials," k6.io, 2025

Why These Tools?

Technical advantages: [1]
- Scale: Millions of concurrent users
- Modern architecture: 10-30x more efficient than JMeter
- Flexibility: Dynamic load patterns, distributed testing
- Observable: Real-time metrics, rich plugin ecosystems

Industry adoption & results: [2,3,4]

EA/DICE (Locust): "Mandatory part of development of any large scale HTTP service at DICE"

Grafana (k6): "Managed to identify bottlenecks and errors in code before shipping" • "Ensures no performance regression reaches production"

AWS endorsement: Published official guides for using Locust on their infrastructure

[1] N. van der Hoeven, "Comparing k6 and JMeter," Grafana Labs, Feb. 2024
[2] "Locust testimonials," Locust.io, 2025
[3] M. Bergquist, "How we use k6 for developing Grafana," Grafana Labs, Aug. 2021
[4] "Using Locust on AWS Elastic Beanstalk," AWS DevOps Blog, Jun. 2022

How This Works: Two Patterns

Pattern 1: External control (Locust + py3270)

```
Laptop/CI → HTTP   → z/OSMF, CICS, Zowe
          → Telnet → 3270 terminals (via py3270)
          → SSH    → Unix System Services
```

Pattern 2: Native execution (k6)

```
z/OS USS → localhost → z/OSMF, CICS, Zowe
         (same LPAR/sysplex)
```

External: Run from anywhere. **Native:** The mainframe tests itself.

Zero agents. Zero mainframe installation footprint.

Locust: Python Ecosystem

```python
class MainframeUser(HttpUser):
    @task
    def submit_job(self):
        response = self.client.put("/zosmf/restjobs/jobs",
                                   data=jcl)
        assert response.json()["jobid"]
```

Open source contribution: Built py3270 plugin for 3270 terminals [1]
Merged: locust-plugins PR #206

[1] A. Rahman, "Plugin for Telnet 3270 User," locust-plugins PR #206, Mar. 2024

k6: Native on z/OS ⚡

The challenge: Run performance tests 24/7 on the mainframe itself.

The solution: Ported k6 to z/OS—added build flags, fixed dependencies [1]

```
export default function() {
  http.post('https://localhost/zosmf/restjobs/jobs', jobData);
}
```

```
$ k6 run test.js --vus 1000 --duration 24h
```

Compiled Go binary. Native execution. Zero external dependencies.

[1] A. Rahman, "Update ui_unix.go with z/OS build flags," k6 PR #2892, Feb. 2023

Why This Matters

Your team already has these skills.

Before:

- $\$/yearvendorlicenses$
- Isolated tooling

After:

- Open source: $0
- Any Python/JS/Go developer
- Unified CI/CD pipeline

The win: Transferable skills. Lower costs. Zero vendor lock-in.

Real-World Impact at IBM Z Test 🎯

Production deployments:

- **Wazi as a Service** - Locust + z/OSMF APIs
- **System testing** - py3270 terminal workflows
- **Customer simulation** - k6 native on z/OS

Full testing stack:
REST APIs • 3270 Terminals • Batch Jobs • Mixed Workloads • CI/CD Gates

The proof: Three production environments. Zero legacy tools.

Try It Yourself

Install:

```
pip install locust py3270
# or
brew install k6  # macOS
# or
go install go.k6.io/k6@latest  # Any platform
```

Read the journey:
Medium: @msradam

- "Swarming Stressed Servers" (Locust + z/OSMF)
- "Ticks by Telnet" (py3270 plugin)
- "Go-ing Native" (k6 porting)

Start: Pick one legacy test. Rewrite it. See the difference.

Questions?

Adam Munawar Rahman
Staff Software Developer @ IBM
M.S. Computer Engineering Student @ NYU Tandon

🌐 adamr.io
✉️ Medium: @msradam
💻 github.com/msradam

"Radical efficiency."