loadgen-rs (Short): Results, Quick Start, and Tool Choice

2026-03-03

A short version of the loadgen-rs article: what it does, benchmark highlights vs h2load, when to choose which tool, and how to start quickly in single-node and distributed mode.

This is the short version.
If you want profiling details, FFI internals, and full cloud deployment walkthroughs, read the full article: loadgen-rs: An HTTP Benchmark Client in Rust.

TL;DR

`loadgen-rs` is a Rust benchmark client for HTTP/1.1, HTTP/2, and HTTP/3 with:

- one CLI for all protocols
- JSONL/CSV output for automation
- distributed controller/worker mode
- scripted Deno scenarios via FFI (k6-style checks/extractors)

In short: very strong for reproducible automation and multi-node testing.
If your only goal is absolute H3 peak throughput, `h2load` still has an edge.

Quick Start

Single-node cli

```
# H2 example, 10s run
loadgen-rs -n 0 -D 10s -c 8 -t 4 -m 10 \
  --alpn-list=h2 --insecure 'https://bench.local:8082/?s=256k'
```

Single-node container

```
# HTTP/3 (QUIC) 10 seconds
podman run -it --rm \
  --network host \
  --name gh-loadgen ghcr.io/git001/loadgen-rs:v0.3.0 \
  -D 10s -c 8 -t 4 -m 10 \
  --alpn-list=h3 \
  --insecure \
  'https://bench.local:8082/?s=256k'
```

Distributed

```
# 1) start workers
bash examples/worker-start.sh

# 2) run controller
deno run --allow-ffi --allow-net --allow-read --allow-env \
  examples/distributed.ts http://worker1:9091 http://worker2:9091
```

Why distributed matters: you can scale beyond one machine and still get statistically correct merged percentiles (histogram merge, not naive p99 averaging).

Benchmark Snapshot vs h2load

High-concurrency test (`-c 512 -t 8 -m 8`, 30s):

Protocol	h2load RPS	loadgen-rs RPS	Ratio
H1	18,851	19,148	101.6%
H2	17,837	18,025	101.1%
H3	9,006	6,702	74.4%

Takeaway: - H1/H2 throughput: effectively on par (or slightly better for loadgen-rs in this run). - H3 throughput: h2load remains faster. - Memory footprint: h2load is smaller; loadgen-rs typically used more memory in tests.

Which Tool Should You Use?

Need	Better fit
Highest H3 throughput only	h2load
One tool for H1/H2/H3 + machine-readable reports	loadgen-rs
Distributed multi-machine load generation	loadgen-rs
Scripted scenarios with checks/extractors and correlation	loadgen-rs
Lowest memory footprint	h2load

Next Step

For the complete deep dive (profiling, architecture, FFI API design, Terraform/Ansible workflow), continue with the full article: loadgen-rs: An HTTP Benchmark Client in Rust.