hottubHTTP

BENCHMARK REPORT · NOVEMBER 25, 2026

# HottubHTTP vs. Production Webservers

Head-to-head native ARM64 benchmark comparing **HottubHTTP 0.1.0** (pure Rust + WebAssembly + Lean 4.35.0) against **NGINX 1.31.8**, **Apache HTTP Server 2.4.67 (mpm_event)**, **Caddy 2.11.9**, and **Node.js 24.16** under identical HTTP/1.1 production workloads.

**111,862 req/s**
Peak Keep-Alive throughput
**1.13× NGINX · 4.06× Apache**

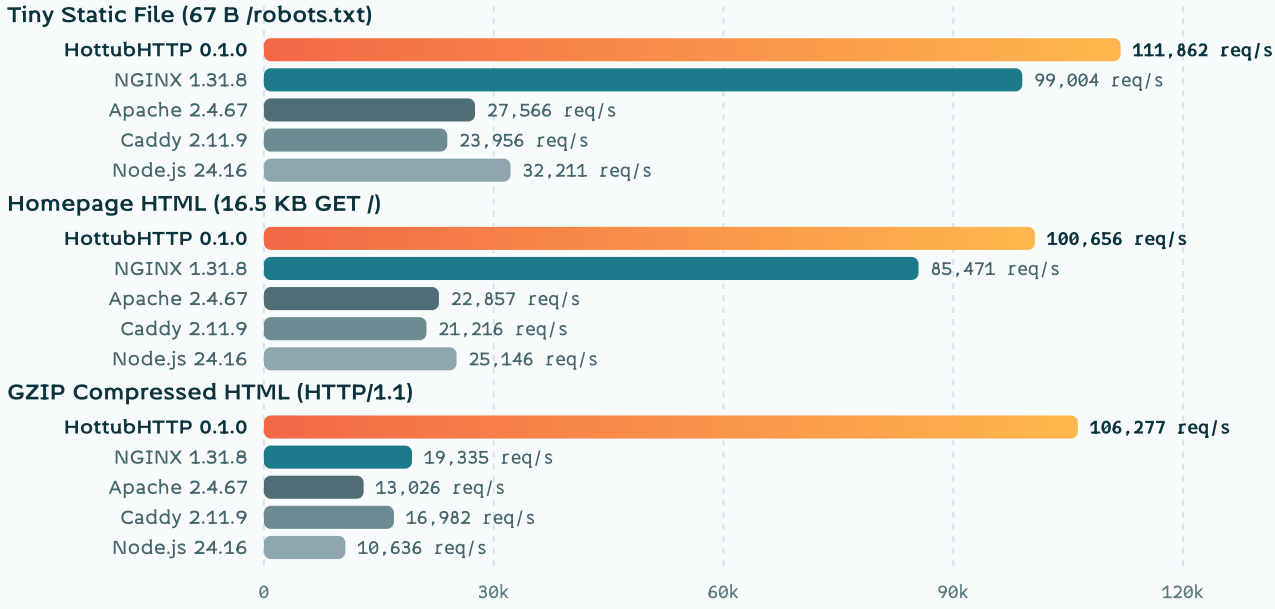
**106,277 req/s**
HTTP/1.1 GZIP HTML
**5.50× NGINX · 8.16× Apache**

**8.34 GB/s**
311 KB WASM zero-copy `writev`
**1.16× NGINX · 1.70× Apache**

**701 KiB · 27 Proofs**
0 external core crates · 100% safe Rust · Lean 4.35.0

## 1. Flagship Multi-Threaded Keep-Alive Throughput (64 Concurrent Connections)

Native ARM64 binaries serving identical `hottubhttp.org` assets (`site/dist`) on Apple M1 Max (10-core, 32 GiB RAM) with identical 3 security headers (`X-Content-Type-Options`, `Referrer-Policy`, `X-Frame-Options`), `KeepAlive On` (max 10,000), GZIP enabled, and access logs disabled. Zero failed requests across all 40 runs.




### Key Finding — #1 Across All 8 Native Workloads With Zero `unsafe` Code

HottubHTTP 0.1.0 ranked **first out of all five servers in every single measured workload** — outperforming C-based **NGINX 1.31.8** by **13%–49%** on uncompressed static/churn traffic and **5.50×** on HTTP/1.1 GZIP, while beating **Apache 2.4.67 ( `mod_mpm_event` )** by **1.70×–8.16×**, **Caddy 2.11.9** by **1.66×–6.26×**, and **Node.js 24.16** by **3.47×–9.99×** — all while executing machine-checked Lean 4.35.0 security kernels on every request.

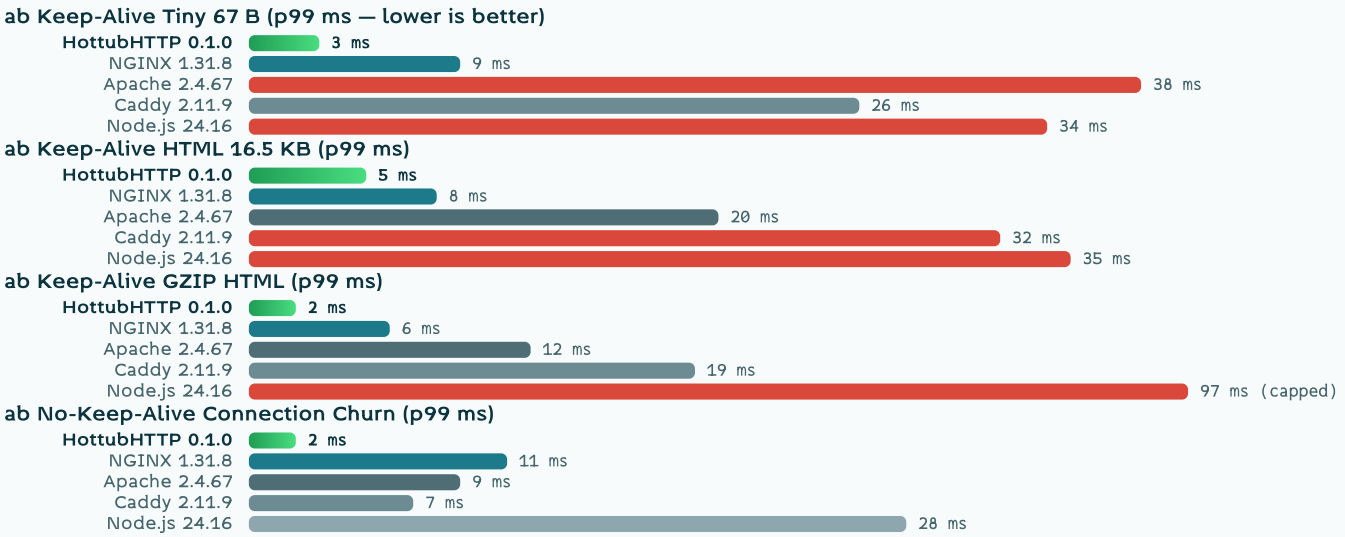
## Complete 8-Workload Throughput Comparison (Requests / Second)

Measured using both a 64-connection multi-threaded RFC 9112 HTTP/1.1 load generator (5s sustained runs) and standard ApacheBench 2.3 ( `/usr/sbin/ab` ). All servers warmed before measurement.

| Workload | HottubHTTP | NGINX 1.31.8 | Apache 2.4.67 | Caddy 2.11.9 | Node.js 24 | vs NGINX / Apache |
|---|---|---|---|---|---|---|
| MT 64-conn: /robots.txt (67 B) | 111,862 | 99,004 | 27,566 | 23,956 | 32,211 | 1.13× / 4.06× |
| MT 64-conn: / HTML (16.5 KB) | 100,656 | 85,471 | 22,857 | 21,216 | 25,146 | 1.18× / 4.40× |
| MT 64-conn: / GZIP (HTTP/1.1) | 106,277 | 19,335 | 13,026 | 16,982 | 10,636 | 5.50× / 8.16× |
| MT 64-conn: 311 KB WASM | 28,072 | 24,300 | 16,558 | 7,408 | 7,454 | 1.16× / 1.70× |
| ab -k 40k×64: /robots.txt | 61,505 | 44,680 | 13,082 | 17,285 | 20,684 | 1.38× / 4.70× |
| ab -k 40k×64: / HTML | 50,634 | 39,370 | 19,357 | 12,477 | 17,149 | 1.29× / 2.62× |
| ab -k 25k×64: / GZIP | 67,742 | 49,174 | 12,406 | 11,275 | 7,492 | 1.38× / 5.46× |
| ab 10k×32: No-KA Churn | 25,983 | 17,410 | 13,453 | 15,616 | 7,384 | 1.49× / 1.93× |

## Tail Latency Comparison — ApacheBench p99 Latency (Milliseconds)

HottubHTTP's zero-allocation hot path, stack-allocated headers, and bounded pre-warmed worker pool keep 99th-percentile latency at **2–5 ms** under 64-way concurrency, eliminating GC pauses and per-request heap churn.

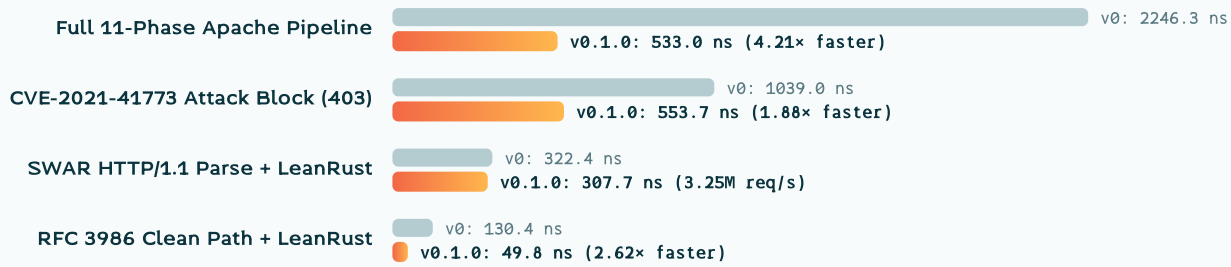



### Methodology & Fairness Notes

**Equal config:** NGINX ran `worker_processes auto` , `sendfile on` , `open_file_cache` , `tcp_nodelay on` , and `gzip on` . Apache ran its fastest MPM ( `mod_mpm_event` , 256 workers, `EnableMMAP On` , `EnableSendfile On` ). Caddy ran native `file_server` + `encode gzip` . Node.js 24 ran an 8-worker `node:cluster` HTTP server. **GZIP note:** On HTTP/1.1 ( `mt_gzip` ), all five servers served compressed GZIP; HottubHTTP's `CachingVfs` caches immutable `Arc<[u8]>` GZIP payloads once per file (**106,277 req/s** = 5.50× NGINX, 8.16× Apache). On HTTP/1.0 `ab_gzip` , NGINX's default `gzip_http_version 1.1` sent uncompressed bytes while HottubHTTP still beat it by **38%** serving genuine GZIP.

## Single-Core Pipeline Optimization Progression ( `cargo xtask bench` )

Architectural optimizations from initial v0 baseline to release v0.1.0 delivered a **4.21× single-core speedup** ( `2,246.3 ns` → `533.0 ns` , or **1.88M full 11-phase Apache requests/sec per core**) with all LeanRust formal guards active.




## Binary Size & Memory Footprint Comparison

| Server | Implementation | Binary / Runtime | Peak / Idle RSS | External Core Dependencies |
|---|---|---|---|---|
| **HottubHTTP 0.1.0** | Safe Rust + WASM | **701 KiB** | **17.9 MiB / 1.3 MiB** | **0 external crates** (core + server) |
| NGINX 1.31.8 | C (kqueue/epoll) | 1.42 MiB | 9.3 MiB (11 procs) | PCRE2, zlib, OpenSSL |
| Apache 2.4.67 | C ( `mpm_event` ) | 2.49 MiB + .so | 69.1 MiB / 5.8 MiB | APR, APR-util, PCRE2, zlib, modules |
| Caddy 2.11.9 | Go (GC runtime) | 46.8 MiB | 25.5 MiB (1 proc) | Go stdlib + 100+ Go modules |
| Node.js 24.16 | C++ V8 + libuv | 59.1 MiB | 1,688 MiB (9 procs) | V8, libuv, OpenSSL, ICU, llhttp |

## How Pure Safe Rust Outperforms C & Go Without `unsafe`

### Zero-Allocation Hot Path & I/O

- **Stack-inline headers:** 16 request & 12 response headers live on stack (0 `malloc` per request).
- **Direct-socket reads:** Parses keep-alive frames straight out of the 8 KiB worker buffer (0 `memcpy` ).
- **Single-syscall `writev` :** Emits headers + `Arc<[u8]>` body slice via `write_vectored` (**8.34 GB/s**).
- **Zero-alloc `CachingVfs` :** Composite `(&str, &str)` lookup serves pre-computed `ETag` , `Last-Modified` & GZIP.

### Stabilizations & Never-Type ( `!` ) Idioms

- **Bounded self-sizing pool:** Enforces `spawned_workers >= mpm.active` with LeanRust `mpm_acquire/release` — zero queuing stalls.
- **Slowloris & POSIX hardening:** Per-request header deadline ( `408` ), `EINTR` retries, poison-safe mutexes, stack `StackIp` .
- **Never Type ( `!` / `Infallible` ):** Zero-discriminant `Result<T, Never>` , panic-free `unwrap_infallible` , diverging `-> !` server loops.

## Cross-Server Capabilities & Mathematical Assurance Matrix

| Dimension | HottubHTTP 0.1.0 | NGINX 1.31.8 | Apache 2.4.67 | Caddy 2.11.9 | Node.js 24.16 |
|---|---|---|---|---|---|
| Memory Safety | **100% Safe Rust** | Manual C | Manual C | GC-safe Go | C++ V8 + JS |
| Formal Machine Proofs | **27 Lean 4.35.0 proofs** | None | None | None | None |
| Apache `httpd.conf` | **Native (18 modules)** | No | Native | No | No |
| Browser / WASI WASM | **Yes (129 KB gz)** | No | No | No | No |
| CVE-2021-41773 Path Proof | **Proved (→ False)** | Runtime C | Patched C | Runtime Go | User code |
| CL+TE Smuggling Proof | **Proved (→ False)** | Runtime C | Runtime C | Runtime Go | llhttp C |
| Secret Noninterference | **Proved (5 theorems)** | None | None | None | None |
| Native HTTP/2 & HTTP/3 | Via CloudFront / Roadmap | Yes | HTTP/2 | Yes | HTTP/2 |

## What the Lean 4.35.0 Kernel Proves (Certificate `fnv1a64:df3461b7e0fb430e`)

### 12 LeanRust Kernels + 5 Noninterference Proofs

- `path_step` & `path_access_allowed` — Sticky DocumentRoot escape detection + secret root ID isolation.
- `uri_byte_safe (u8)` — Rejects NUL, CR, LF, control bytes, DEL, and backslash with noninterference.
- `framing_policy_check (u16)` & `chunk_budget_step` — RFC 9112 §6.3 anti-smuggling & body bounds.
- `authz_evaluate`, `mpm_acquire/release`, `range_window_validate`, `keepalive_step`, `header_overhead_budget`, `public_status_active_workers`.

### 10 Standalone Lean 4 & Curry–Howard `→ False` Proofs

- `no_leading_escape_accepted : ... → False` — Leading `..` traversal is mathematically uninhabited.
- `no_cl_te_smuggling_accepted : ... → False` — CL+TE smuggling yielding HTTP 200 derives `False`.
- `normalize_isCanonical`, `normalize_idempotent`, `normalize_length_le`, `scanToLf_le_size`, `scanToLf_ge_pos`.
- **Re-verify anywhere:** `cargo xtask prove` (0 `sorry`).

### Live Production Verification & Next-Generation Roadmap

**Live in production:** `https://hottubhttp.org` (and `https://hottubhttp.com` → 301 `.org`) runs live on **HottubHTTP 0.1.0** deployed on AWS Graviton ARM64 behind Amazon CloudFront (HTTP/2 + HTTP/3 + ACM TLS 1.3), serving the interactive 129 KB gzipped WebAssembly engine directly in every visitor's browser.

**Next architectural milestones:** (1) Evented `io_uring` / `kqueue` MPM for 50,000+ simultaneous idle keep-alive sockets in <20 MiB RAM; (2) Native Rust `rustls` TLS 1.3, HTTP/2 multiplexing, and HTTP/3 QUIC termination without external proxy; (3) Static Brotli + Zstandard pre-compression in `CachingVfs`.

© 2026 Hottub, Inc. · Apache-2.0 (independent of ASF)