Alireza (Parsa) Ghadimi

Systems Engineer

San Francisco, CA

hey@parsa.wtf   +1 917 420 2781   parsa.wtf   qti3e

Systems engineer with 15 years of experience, passionate about understanding how complex systems work at a fundamental level. I focus on building correct, reliable software—from implementing consensus protocols to designing compilers with rigorous semantics. Strong background in Rust, performance optimization, and distributed systems.

Work Experience

System Engineer *02/2025 – 02/2026*
Deno Remote

Working on Deno Deploy.

- Implemented HTTP cache for Deno Deploy (RFC 9111).
- Created a filesystem cache by building a user-space VFS emulating Linux semantics.
- Optimized filesystem performance in gVisor fork before migration to Firecracker.
- Implemented anti-entropy based cache invalidation broadcast.

CTO → Lead → Senior Engineer *02/2021 – 12/2024*
Fleek New York, USA

Distributed infrastructure platform ($25M Series A). Grew from senior engineer to CTO, leading 30+ engineers.

- Designed and built large-scale distributed systems handling petabytes of data.
- Implemented Raft consensus; created high-performance data pipelines.
- Built tooling and SDKs to accelerate engineering workflows and experimentation.
- Profiled and optimized performance-critical systems.

Junior Software Engineer *03/2018 – 07/2018*
PropelML Remote

Machine learning tooling for JavaScript. Worked directly with Ryan Dahl (creator of Node.js/Deno).

- Built interactive notebook environment for ML experimentation (Preact).
- Ported numpy's tensor formatter—understanding tensor representations and data layouts.
- Created visual inspection tools for debugging ML outputs and intermediate values.
- Designed serialization protocol for transferring complex data structures across workers.

Compiler Engineer *12/2018 – 07/2019*
Truebase Toronto (Remote)

- Implemented compiler backend: code generation, optimization passes, IR design.
- Built developer tooling for rapid iteration and debugging.

Technical Skills

Languages: Rust, C, Go, TypeScript, Python, x86-64, Bash, etc.
Compilers & IR: SSA, Cranelift, LLVM, WebAssembly (text & binary), EVM, e-graphs, abstract interpretation, symbolic execution
Performance: SIMD (AVX2, AVX-512), profiling, memory optimization
Distributed Systems: Large-scale data pipelines, Raft consensus, anti-entropy protocols, TLA+

Linux & Virtualization: eBPF, io_uring, KVM, Firecracker, gVisor
Runtimes: V8, Wasmtime, Deno

Current Focus

Building JOE, a JavaScript AOT compiler focused on correctness and sound optimizations. Exploring compiler theory: SSA, e-graphs, dataflow analysis, and symbolic execution.

Selected Projects

Cached (Deno – Proprietary) *2025*

- Edge caching proxy powering Deno Deploy; Varnish-like HTTP cache in Rust.
- RFC 9111 compliant: Cache-Control parsing, Vary header support, range requests.
- Zero outages in production; handled full edge traffic without incident.

Cranelift PTX Backend for Rust *Ongoing*

- Building a PTX backend for Cranelift, integrating it into rustc for GPU compilation.
- Implementing correct instruction selection and register allocation for NVIDIA GPUs.
- https://github.com/qti3e/wasmtime/commit/dc3141b — https://github.com/qti3e/rust/commit/671a0e2

JavaScript Optimization Engine (JOE) *Ongoing*

- JavaScript AOT compiler in Rust: SSA-based IR, e-graph optimization, aggressive static analysis.
- Zero dependencies (no std, alloc, libc)—correctness and understanding from first principles.
- Applying compiler theory: dataflow analysis, sound optimizations with rigorous semantics.

Tango *Ongoing*

- Content-addressed incremental view maintenance engine in Rust.
- Staged transaction processing with cascading updates; VTable-based projections.
- Blake3 content addressing, storage abstraction (in-memory/RocksDB).

Research Collaborations *2022*

- **Dfinity tECDSA:** Collaborated with Victor Shoup on cryptographic optimizations.

Performance Engineering *2024*

- **Blake3 JS:** WebAssembly JIT achieving 2.2x native WASM speed—https://parsa.wtf/blake3
- Experience profiling, optimizing, and understanding performance at the instruction level.

Approach & Interests

I learn by reading source code—I have V8, LLVM, the Linux kernel, and four SAT solvers cloned locally. I'm drawn to understanding how systems work mechanistically and building software that is provably correct. Whether it's implementing consensus protocols, designing compiler IRs, or optimizing critical paths, I care about getting the details right.

Open Source I Learn From

Compilers: LLVM, Cranelift, rustc, GHC, egg **Systems:** Linux kernel, V8, gVisor, Firecracker **Solvers:** z3, kissat, varisat, splr

Awards

South Korea Science & Engineering Fair Medalist 2019
"WaterScript": JavaScript optimization engine using execution simulation to understand program behavior.

Best Lecture in Computer Science & Math 2019

Lecture on "Dead Code Elimination in Dynamic Languages using Execution Simulation"—analyzing how programs compute.

3x Gold Medal – Festival of Student Achievements 2017–2019

First-place award 3 consecutive years for research projects.