Shihang (Vic) Li

shli@cs.washington.edu | vic-li.me | github.com/vic-lsh

Research Summary

Ph.D. student in systems and networking, working on LLM agent systems, HW/SW co-design for memory management, and microservice systems. I build systems across the stack—from hardware abstractions to OS kernels and application runtimes.

Education

University of Washington, Seattle, WA Sept. 2023 – Present
Ph.D. in Computer Science. Advisors: Simon Peter, Tom Anderson. GPA: 3.8/4.0

Brown University, Providence, RI Sept. 2021 – May 2023
M.S. in Computer Science. Advisor: Malte Schwarzkopf. GPA: 4.0/4.0

New York University, New York, NY Sept. 2016 – May 2021
B.S. in Computer Science & Business (Double Major). GPA: 3.72/4.0

Research Experience

FlowGuard: securing LLM agents from prompt injection attacks Jun. 2025 – Sep. 2025
Microsoft Research Intern. Mentors: Landon Cox, Pedro Henrique Penna, Rodrigo Fonseca Redmond, WA

Use information flow control (IFC) to prevent prompt-injection data leaks for LLM agents.

- Built a reference monitor in Rust that transparently and deterministically prevents agent data leaks with MCP interposition.
- Formalized prompt-injection data leak freedom requirements and implemented with secrecy & integrity label tracking.
- Proposed sub-agent delegation with bounded channels (e.g., booleans) to limit exfiltration while preserving functionality.
- Demoed blocking GitHub secret-exfiltration attacks, leading to a patent application and production deployment exploration.

Self-Defining Systems Dec. 2025 – Present
with Tom Anderson, Simon Peter, Ratul Mahajan, Luke Zettlemoyer *Position paper*

Use LLM agents to continuously evolve systems software end-to-end — from initial design to deployment.

- Designing a multi-agent system for e2e system evolution, incl. deployment, monitoring, and feedback-driven optimization.
- Early results show that LLM agents can collaborate to deploy apps with 40+ microservices without human input.

NEMO: high-fidelity memory observability via programmable memory controllers Mar. 2024 – Present
with Matthew Giordano, Daniel S. Berger, Tom Anderson, Simon Peter *Under submission*

A programmable memory controller (MC) that provides high-fidelity memory telemetry, flexibly programmed by the OS.

- Designed a programmable, match-action–style telemetry pipeline that inspects every memory request at the MC w/o sampling.
- Integrated NEMO into Linux's `mm` subsystem and a userspace memory manager via `userfaultfd`.
- Lowered detection time by up to 10x and reduced overhead by up to 350x compared to Intel PEBS and MBM.
- Improved performance by up to 1.7× in throughput and 23% lower tail latency across tiering and noisy-neighbor scenarios.

Masa: rethinking microservice SLOs with end-to-end deadlines Mar. 2024 – Present
with Simon Peter, Danyang Zhuo, Arvind Krishnamurthy, Ratul Mahajan

A microservice runtime that enforces e2e RPC SLOs in microservice graphs, instead of per-service SLOs.

- Designed a distributed RPC scheduler that manages RPC slack in e2e SLO across service graphs, improving goodput.
- Extended gRPC to transparently propagate per-RPC priority across microservice graphs with zero application change.
- Modified the Tokio async runtime in Rust to support per-task priority and custom scheduling policies (e.g., least-slack-first).
- Improved goodput by 20% on DeathStarBench and diverse Alibaba call graphs via trace replay.

Published Research

Quicksand: Harnessing Stranded Datacenter Resources with Granular Computing NSDI '25 (paper, code)
Z. Ruan, **S. Li**, K. Fan, M. Aguilera, A. Belay, S. Park, M. Schwarzkopf

Minimize stranded datacenter resources with *resource proclets*—small, migratable units that primarily use one resource.

Loom: Efficient Capture and Querying of High-Frequency Telemetry SOSP '25 (paper, code)
F. Solleza, **S. Li**, W. Sun, R. Tang, M. Schwarzkopf, A. Crotty, D. Cohen, N. Tatbul, S. Zdonik

A telemetry collector of high-frequency metrics for observability workloads in production.

Skills

- Systems programming (Rust, C, C++), async runtimes, RPC, OS kernel, HW design (SystemVerilog), Agents (Python, Go).