# Ganghuan Liu

**Email:** admin@arloor.com
**Phone:** 17625955421
**Github:** https://github.com/arloor
**Website:** www.arloor.com

I’m a software engineer with 6 years of experience, mainly in infrastructure and middleware. For the past 4 years, I’ve focused on observability, especially distributed tracing. I built a tracing system from scratch for my company and have experience in its architecture, design, and performance optimization. I’ve also worked with Redis proxy and Elasticsearch middleware. My main programming language is Golang, but I also have strong experience in Rust, with projects in HTTP proxies and eBPF. In my free time, I write blogs on observability, Rust, eBPF, Redis, and Kubernetes.

## Experience

> **ByteDance —— Real-Time Communication —— Scheduling (2024.06 - Present)**
>
> - Nothing notable

> **Xiaohongshu —— Observability (2023.03 - 2024.06)**
>
> - **Distributed Tracing System**
>   - Instrumentation: Implemented client-side instrumentation based on CAT client (open-sourced by Dianping), drawing inspiration from Skywalking’s protocol.

    - Storage: Utilized Clickhouse for storing trace details, trace indexes, service topology, and performance metrics.
    - Performance Optimization: Optimized TraceID format to improve Clickhouse's point query performance; used SummingMergeTree to store performance metrics, reducing the cost of metric aggregation.
    - Product Feature: trace searching, trace details, service topology, traffic topology, slow trace profiling, and QPS estimation.
- **Profiling/Continuous Profiling**
    - Used async-profiler to collect performance data from Java applications and generate flame graphs at the function, file, and package levels.
    - Compared daily flame graphs to identify performance degradation causes and pinpoint the code changes responsible.
    - Performed company-wide flame graph aggregation and analysis to identify CPU waste caused by misuse/abuse.

**MeiTuan —— Observability (2020.06 — 2023.03)**

- **Tracing：** Optimized serialization and Message Queue to increase system throughput from 400 million QPS to 1.1 billion QPS. Improved the query system to resolve Java

OOM issues caused by the creation of a large number of new objects during service topology queries.

- **Dashboard：** Through modular design, support models for different metrics, providing a one-stop display of all business-relevant metrics, with support for formula calculations and custom alerts.
- **Metrics:** Host/container metrics monitoring, covering CPU, memory, disk, network, kernel network stack, and other indicators. Supports storage, querying, and alerting.

**China Merchants Bank —— Infra Team (2019.07 - 2020.05)**

- **Redis Cluster Proxy:** Communicates with the Redis cluster using Redis's RESP protocol to obtain the mapping between slots and nodes, and routes client query requests to the correct node. Additionally, it supports multi-key requests such as mget and mset which may cross multi hashtags.

## Education

**Bachelor of Engineering in Software, Nanjing University, 2015-2019**

Graduated as an outstanding graduate.

## Rust Experience

**rust_http_proxy:**

- HTTP forward proxy, reverse proxy, and static resource server based on Tokio, Hyper, and RustIs.

- **Async Rust:** Extended TcpStream and TlsStream to implement AsyncRead and AsyncWrite traits, enabling bandwidth statistics at the TCP stream level and automatic connection closure on idle timeout.
- **Async Rust:** Extended TlsAcceptor and TlsStream to periodically refresh TLS certificates.
- **Observability:** Used eBPF (based on libbpf-rs) to monitor bandwidth usage on network interfaces and cgroups, and utilized Prometheus exporter to track user traffic and external link clicks.
- Implemented an HTTP1 proxy connection pool using LruCache.
- Visualized traffic monitoring page using React and ECharts.js.
- Supported GZIP compression and resumed downloads.

**axum-bootstrap:**

- Rust web service scaffolding based on Axum, integrating common components like sqlx and reqwest.

**shadowsocks-rust (fork):**

- Extended sslocal to support HTTP tunnel proxy as a server.

**bpf_rs_hub:**

- A collection of BPF programs based on libbpf-rs, including network interface bandwidth statistics, cgroup bandwidth monitoring, and tracing TCP connection events.

# Technical Blog

- Obs • ARLOOR
- Redis • ARLOOR
- Ebpf • ARLOOR
- K8s • ARLOOR
- Rust • ARLOOR
- Golang • ARLOOR