Yiqi Xue

Los Angeles, CA 90007 | +1 (626) 418-5735
xuey666@gmail.com | https://github.com/Onexyq | https://www.linkedin.com/in/yiqixue/

Education

University of Southern California (USC) — Los Angeles, CA
Master of Science in Computer Science — *Aug 2023 — May 2025*

University of California, Los Angeles (UCLA) — Westwood, CA
Bachelor of Science in Applied Mathematics, Minor in computing — *Sep 2018 — Jun 2020*

Skills

Programming Languages	C++, C, Java, Python, Go, HTML/CSS, JavaScript, Shell, SQL
Databases	MySQL, PostgreSQL, Oracle, Redis, MongoDB, Cassandra, Memcached
Big Data	Hadoop, HDFS, MapReduce, HBase, Spark, Zookeeper
Frameworks	Spring Boot, MyBatis, JUnit, React, REST, SOAP, GraphQL, Protobuf, gRPC, Thrift, Agile, UML, Design Patterns
Developer Tools	Git, SVN, Unix/Linux, Docker, Kubernetes (GKE, EKS), Kafka, RabbitMQ, ElasticSearch, Nginx, AWS (EC2, S3, Lambda, RDS), GCP, makefile, CMake, Maven, GitLab CI, Jenkins, Postman

Work Experience

Huawei Technologies Ltd — Beijing, China
Software Engineer - SQL Engine Team — *Mar 2023 — Jun 2023*

- Engaged in development of GaussDB Engine and contributed **2,000+** lines of **C** to **openGauss (500+ stars)** codebase.
- Revamped partial implementations of **SQL layers** (optimizer, executor) and evaluated performance with TPC-C benchmark.
- Collaborated with a team of 15 using **Git**, constructed **Jenkins** CI/CD pipelines to validate test cases, and created **Linux bash scripts** to complete automation tasks.
- Debugged 5 performance issues using **profiling tools** such as perf and flamegraphs, analyzed core dumps files with **GNU Debugger (GDB)**, and diagnosed **20+** memory leak problems using **Valgrind**.
- Devised and constructed a PL/SQL Cache System that cached **1M+ stored procedures** and realized concurrency using a thread pool, DB locks, and shared memory, optimizing system run-time memory occupation rate by **80%**.

WellinTech Ltd — Beijing, China
Software Engineer - Backend Development Team — *Sep 2020 — Jun 2022*

- Coordinated with 4 R&D teams to develop a **Cloud-Native Data Warehouse** based on distributed microservices architecture.
- Leveraged **SQLite** and **HBase** to store **Terabytes** of historical data. Implemented, iterated, and documented **APIs** for data access and persistence layer using **C++** and lifted test coverage to **90%** with **gTest**.
- Negotiated with tech leaders and designers to refactor the **RPC** layer using **Kafka** along with Google **Protobuf**, improving the data throughput by **130%**.
- Enhanced query performance and system robustness by modifying **8,000+** lines of problematic code from legacy codebase, culminating in a **30%** increase in TPS (Transactions per second) for SQLite instances and a **20%** increase for HBase clusters.
- Deployed the services via **Docker** and **Kubernetes**, and employed Apache **Zookeeper** clusters for coordination to accomplish **redundancy** and **high availability** in live production environment.

Projects

ViewCrypto.io | *Java, ReactJS, REST, MongoDB, AWS, Kafka, ElasticSearch*

- Led a team of 4 to frame and develop a scalable web app to visualize real-time NFT prices in marketplace.
- Deliberated and implemented functionalities based on **RESTful APIs** such as user sign-in, authentication, merchandise showcase, personal info page using **React.js**, and aggregated user data in **MongoDB**.
- Utilized **Spring Boot** and **Spring MVC** to construct the backend framework and managed dependencies by **Maven**.
- Implemented system notification for user follow, like, and comment functionalities with **Kafka**.
- Engineered a search module with **ElasticSearch** to allow search by keywords and deployed the web app to **AWS EC2**.

TinyWebServer | *C++, Socket, HTTP, MySQL, Multithreading, OOP*

- Fabricated a **multi-threaded** web server which allows **10,000+** concurrent connections and realized **non-blocking IO**.
- Built a voting web application that handles **HTTP** requests and accesses **MySQL** database.
- Utilized **RAII** to create a **connection pool** for database connections management.
- Implemented an **asynchronized** log file system using **singleton pattern** and **block queue** to facilitate monitoring server status.

TinyKV | *Golang, Raft, Distributed System, RPC, Protobuf*

- Programmed a distributed fault-tolerant KV storage system using **Go** and integrated **Raft** algorithm features, including membership change and leadership change, log garbage collection, and snapshot for log compaction.
- Constructed the multi-version concurrency control (**MVCC**) layer, optimizing transaction management.
- Developed a basic scheduler for centralized node management and timestamp generation, utilizing **Protocol Buffers** over RPC for efficient inter-node communication.