SIYANG ZHANG

Vancouver, BC | (236) 989-8162 | siyangzhang1@gmail.com

Interested in backend & cloud development, have experience in dev ops / testing, database, cloud and distributed system.

EDUCATION

Northeastern University — Vancouver, Canada

Master in Computer Science; GPA: 3.9 / 4.0 — *Jan. 2021 - Dec. 2022*

Harbin Institute of Technology — Harbin, China

Bachelor in Financial Management; GPA: 3.7 / 4.0 — *Sept. 2014 - Jul. 2018*

SKILLS

- **Languages**: Java (JVM, GC, JProfiler, JMS), Go, Python, C/C++, SQL (MySQL Server, Workbench), NoSQL (MongoDB, Redis, DynamoDB), HTML, CSS, JavaScript, TypeScript, HCL, YAML, JSON, GraphQL
- **Development**: Cloud, Database, Distributed System, Microservices, CI-CD Pipeline & Automation & Testing, Dev Ops Configuration & Containerization Orchestration & Monitor, Object-oriented, IAM, RESTful Routing, gRPC, Web API, Message Queue, Cache, Streaming, Scaling & Load balance, IaC
- **Tools and Technologies**: AWS (EC2, S3, RDS, Lambda, SNS, DynamoDB, IAM, API Gateway), Spring/SpringBoot (JPA/ Hibernate, Maven, AOP, MVC, Security, Cloud), Junit, Mockito, React, Redux, Node, Express, Linux, VM, NPM, Git, Docker, Kubernetes, Helm, Ansible, ELK, Filebeat, Prometheus, Hashicorp (Terraform, Vault, Consul, Vagrant), Nginx, Kafka, Redis, Jenkins, Istio, Jira, Confluence, Kanban, Swagger

PROFESSIONAL EXPERIENCE

- **Index Exchange** — *Toronto. Canada*

  Software Engineer Intern, Cloud Platform — *Jan. 2022 - April. 2022*

  - Open-Source Project - Hashicorp Vault (User Secrets Management)
  - Developed Ansible playbook run by Python scripts to set up Gitlab CI/CD pipelines, automating Vault approle and policy creation
  - Spearheaded the implementation of a Vault audit monitor pipeline, configuring Vault audit and server logs to pass through Kafka into the ELK (Elasticsearch, Logstash, Kibana) stack with Filebeat processing for data format validation and error checking, allowing visualize and monitor system performance, providing insights for further analysis.
  - Integrated ArgoCD to enable automatic secret rotation within the CI/CD pipeline instead of manual reconfigurations.

PROJECTS

- **Distributed System**: Implement fault-tolerant sharded key/value-service system with log compaction using Go — *Jan. 2023*
  - System shards the keys over replica groups for performance(throughput) and load balance, can handle linearizable concurrent calls and replicas reconfiguration (join and leave) while maintaining strong consistency.
  - Servers use Raft for replication in spite of failures or network partitions. Enable Raft leader election.
  - Implement Raft as a Go object, which supports appending index-numbered client request as log entries and storing the sequence of reconfigurations as replicated logs, both log files will eventually be committed.
  - Each replica group handles requests for shards subsets in parallel and maintain consensus when reconfiguration.
  - Implement log compaction / snapshots to save persistent state (for server reboot & resume) where server will periodically persistently store a snapshot of its current state, and Raft will discard old, preceding log entries to save space.
- **Distributed System**: Implement Multi-Threaded Client-Server Communication with Java — *Jun. 2022*
  - Designed and implemented single-threaded client-server key-value (KV) store using TCP and UDP protocols respectively.
  - Extended the project to enable concurrent, multi-threaded communication by leveraging remote procedure call (RPC) through Java RMI.
  - Added replicated servers ensuring consistent responses from concurrent client requests through a Two-Phase Commit protocol.
  - Implemented fault tolerance and consensus mechanisms using the Paxos protocol, enhancing system reliability and data integrity.
- **Database System**: Built a fully functional, optimized database supporting simple and nested correlated queries with Java — *May. 2020*
  - Implemented a parser for SQL statements and designed an index system using B+ trees.
  - Engineered logic for various types of joins, including partial, inner, outer, and equi-joins.
  - Optimized query execution by transforming incoming queries into relational algebra.
  - Applying Selinger optimization techniques based on dataset sizes and clusterings.
- **Ticketing App**: Full-Stack E-Commerce Application using Microservices with Node, React, Docker and Kubernetes — *Sep. 2019*
  - Utilized React, Hooks, and Next.js to deliver an exceptional user experience, enabling client registration, seamless order booking, and secure payment processing with JWT-based authorization.
  - Architectured a robust multi-service application, facilitating asynchronous, event-based communication between services.
  - Developed individual services using Node and Express, with data storage in Redis, optimizing performance and ensuring scalability.
  - App is written with Typescript, orchestrated deployment within Docker containers executed on a Kubernetes cluster, enhancing scalability and reusability, utilized AWS for seamless deployment and hosting of the application.
- **YelpCamp**: Full stack of web development (a Yelp.com style website for campgrounds) — *Feb. 2019*
  - Implemented full CRUD functionality, enabling users to register, log in, post reviews and comments, and edit their submissions.
  - Utilized Bootstrap for responsive and visually appealing user interfaces.
  - Leveraged the Express framework and Node.js for the server-side logic and routing.
  - Designed and implemented a MongoDB database to efficiently store reviews, comments, and relevant users information