Aaron Tan

(475)–326–9147 - aaron1004780912@163.com - linkedin.com/in/yuhan-aaron-tan

EDUCATION

Cornell University — **Ithaca, USA**
Candidate for Master of Information Science; GPA: 3.8/4.0 — *08/2023 - Present*

University of Liverpool — **Liverpool, UK**
(First Class Hons)BEng in Computer Science and Electronic Engineering; GPA: 3.8/4.0 — *09/2021 - 07/2023*

Xi'an Jiaotong-Liverpool University — **Suzhou, China**
Major in Computer Science and Technology — *09/2019 - 07/2021*

SKILLS

Languages: Java, Python, C++, C#, C, SQL, Shell, HTML, R
Framework&Tools: Spring, Spring Boot, MySQL, Redis, MyBatis Plus, Kafka, Docker, Linux, CentOS, Knife4j, Hadoop, Hive, HDFS, VIM, Qt, GIT, PyTorch, TensorFlow, Numpy, AWS, Jira, Agile

EXPERIENCE

Software Development Intern | Spring Boot, MyBatis Plus, MySQL, Redis — **05/2024 - 08/2024**
NextTier

- Implemented **distributed sessions** to synchronize login states across distributed servers by using **Redis**. Enabled single field modification by using **Hash** instead of **JSON** to store user information, reducing memory **by 20%**.
- Reduced API response time **by 95%** by caching user information in **Redis** and ensuring data integrity with **custom Redis serializers**.
- Achieved scheduled **cache warming** with **Spring Scheduler**, improving initial access speed **by 97.7%.**
- Improved database write operations by using **custom thread pools** with **CompletableFuture** concurrency, reducing import time for **1 million** rows **from 300 seconds to 54 seconds**.
- Introduced friend similarity matching function using the **edit distance algorithm**. Optimized matching speed **from 34 seconds to 7 seconds** through memory optimizations, selective data retrieval, and caching strategies, inspired by **large-scale** recommendation systems
- Enhanced concurrency control by **Redisson distributed locks** to prevent duplicate team joining and exceeding team capacity, ensuring **mutual exclusion** and **API idempotency**.

Software Development Intern | Kafka, Spring Boot — **08/2023 - 12/2023**
Eth Technology

- Developed a **streaming microservice** capable of processing over **1000 events** concurrently, using **Kafka** and **Spring Boot** to ensure efficient and **scalable** event handling.
- Designed and built **REST APIs** for event consumption and publication to Kafka topics, implementing Kafka Consumer and Producer patterns.
- Implemented **unit tests** and **integration tests** using **JUnit** and **Embedded Kafka**, achieving **90% code coverage**; Conducted **end-to-end testing** for different scenarios of data-streaming APIs using **Postman**; Implemented **concurrency testing** & **automatic load testing** process using **Jmeter**.
- Integrated **Spring JPA** and utilized **H2 database** to store events metadata.

PROJECTS

Asynchronous Processing Framework: AaronFlow | Spring Boot, MySQL, Redis — **03/2024 - Present**
*AaronFlow is an asynchronous task processing framework developed in Java that supports **automatic scheduling, automatic retries,** and **flexible task configuration**.*

- Implemented a two-layer architecture with Flowsvr (Server) providing **HTTP services via APIs** for **task querying, scheduling, and management**, and Aaron (Worker) pulling and consuming tasks, achieving a separation between **distributed task scheduling** logic and **business** logic.
- Designed database tables by separating task information, configuration, and scheduling, achieving a **loosely coupled** structure that reduces dependencies between tables. Used **indexing** for quick **task retrieval**, allowing flexible **task registration** and **management**.
- Implemented **timeout task monitoring** and **recovery**, using a **polling** mechanism to regularly check task status. Monitored table size to trigger **table partitioning** when thresholds are reached.
- Optimized multi-worker coordination by initially using **MySQL row-level locking** to prevent multiple Workers from pulling the same batch of tasks, and later improved performance by introducing **Redis distributed locks** on the Aaron (Worker) side.
- Implemented performance optimization by conducting **stress testing** with **wrk and Lua scripts** to analyze bottlenecks. Utilized a **MySQL connection pool** and increased the maximum number of connections, improving throughput **from 100 QPS to 500 QPS**.