Kaining Zhong

kaining6@illinois.edu | (447) 902-2049 | github.com/PRESIDENT810

EDUCATION

University of Illinois, Urbana Champaign - *Computer Engineering, MEng.* Aug. 2022 – May. 2024
The Chinese University of Hong Kong - *Computer Science, BEng.* Aug. 2017 – May. 2021

SKILLS

Programming languages: C++, Python, Golang, Rust, TypeScript, Java, SQL
Tools: AWS, LLVM, Bazel, Git, PyTorch, Relational DBMS, CUDA, Linux, Xcode

Work Experience

Amazon - *Software Development Engineer Intern, AWS CloudFront* May. 2023 – August. 2023

Control Plane - Realtime Logging

- Developed, tested and deployed an integral component of AWS CloudFront's realtime metrics monitoring system, focusing on **leveraging Google FlatBuffers to serialize realtime metrics** collected from CloudFront's Nginx & Squid caching servers, **utilizing zstd to compress serialized binary byte stream**, reduced 48% size of serialized realtime metrics
- Employed AWS C++ SDK to feed data on realtime metrics processing cluster io AWS Kinesis stream processing system, and employed AWS Java SDK to consume realtime metrics and display it on AWS CloudWatch monitoring service.

TikTok - *Software Development Engineer, iOS Infrastructure* July. 2021 – July. 2022

LLVM Compiler Infrastructure & Xcode Toolchain

- Improved Xcode debugging experience by optimizing Apple's LLDB, and deployed Xcode toolchain bundled with customized LLDB to multiple iOS APP's development team, including TikTok, Lark, Cloud IDE for iOS, etc. **Reduced debugging time cost for displaying variables & expression evaluation from 200s to 30s** for iOS applications in massive scale such as Lark, which **links more than 600 static frameworks** to a main executable **over 1G size**.
- Fixed bugs for LLVM's LLD linker and migrated from Apple's LD64 linker to LLD linker, enabling faster linking for iOS applications including Lark & Toutiao; **reducing linking time from 120s to 60s in average**.
- Implemented a LLDB performance analyzing tool by dynamically rebinding exposed C++ Script Bridge API symbols in LLDB bundled in Xcode, using Facebook's FishHook tool. Uploaded the obtained statistics to a backend visualization service using nlohmann's C++ JSON library and Objective-C's CFNetwork library.
- Maintained a backend service implemented by **ByteFaaS serverless** that managed Xcode toolchains compiled from internal fork of Apple's Swift repository, using TOS as Object Storage Service to store toolchains generated from CI tasks, and deploying customized toolchains automatically via Podfile.

Bazel Build System

- Collaborated in migrating TikTok and TouTiao's build system from xcodebuild to Bazel and transformed the overall project structure to monorepo, **reducing compilation time cost from 11min to 6min on average**.

PROJECTS

Needle, a PyTorch-like deep learning framework Python, C++, CUDA

- Implemented a PyTorch style deep learning framework with components including **dynamic computation graph, automatic differentiation, dataloader, optimizer, some widely used neural network modules, and backend operators including a img2col optimized convolution operator supported by C++ and CUDA**.
- Trained ResNet9 model on CIFAR-10 dataset, and LSTM model on Penn Treebank dataset entirely relying on Needle framework.

Bustub, a Relational Disk Based Database C++, GTest

- Implemented a **concurrent B+ tree index** for faster data retrieval, which supports high concurrency access using latch crabbing to avoid unnecessary multi-thread synchronization, along with a **buffer pool manager** using **extendible hash table** and **LRU replacement policy**.
- Implemented **volcano model operators** that execute SQL query plans, supporting a subset of SQL syntax including sorting, joining and aggregation.
- Implemented a **transaction manager to support concurrent SQL queries**, which supports isolation levels up to Repeatable Read, along with a **deadlock detector** that aborts transaction when deadlocks are detected.

Distributed, Fault-tolerant KV NoSql Database Golang

- Built a fault-tolerant Key-Value NoSQL database **based on Raft consensus algorithm** using Golang's concurrency & RPC packages, supporting Put, Get, Append operations, and guaranteeing **Consistency & Partition tolerance**.
- Reduced network communication overhead by employing fast log synchronization, and enabled crash recovery by implementing a **Snapshot persistence mechanism**.