Xiangfeng(Allen) Zhu

zxfeng@umich.edu · xzhu27.me · 650-660-0918 · github.com/Romero027 · xzhu

Education

University of Michigan, Ann Arbor — Expected: May 2020
Bachelor of Science, Computer Science — GPA:4.0/4.0

Experience

Dropbox San Francisco, CA — May. 2019
Incoming Software Engineer Intern

- Will be working in Dropbox as a Software Engineer intern starting May 2019

Software Systems Lab University of Michigan — Dec. 2018 - Present
Research Assistant Work Under: Prof. Mosharaf Chowdhury

- Co-developing a general-purpose execution engine tailored for latency-sensitive wide-area computation on top of Apache Spark
- Improved the job completion time by 6.8x on average compared to the state-of-the-art Spark-based wide-area computation frameworks

Hainan Airline Beijing, China — July. 2018 - Sep. 2018
Software Engineer Intern

- Worked in Airline Map team to create new navigation app for pilots
- Implemented newly-designed pages and built an interactive navigation with HTML, CSS, XML and OpenLayers3

Disorderly Lab UC Santa Cruz — Mar. 2018 - Sep. 2018
Undergraduate Researcher Advisor: Prof. Peter Alvaro

- Developed a debugging approach based on analysis of provenance data obtained during system executions equipped with correctness specifications
- Helped Design a standalone prototype Debugger Nemo and validated Nemo on protocols from real-world distributed bugs

Skills

- Language: Java, C, C++, Python, Scala, MATLAB, Bash, SQL, HiveQL, HTML, CSS, LaTeX, Go(Limited), JavaScript(Limited)
- Tool: Perf, Valgrind, Git, Vim, Neo4j, Docker, Xcode, Flask
- Data: Oracle, MySQL, SQLite3, Hadoop, Hive, Spark, Flink

Projects

Distributed Debugger Using Provenance Graph (Go) — Mar. 2018 - Aug. 2018

- Designed a lineage-driven distributed debugger(Nemo) with graduate students that can analyze the program and give suggestions to the programmer how and where to correct the program

Fault-tolerant Scalable Key-Value Store (Python) — Jan. 2019 - Mar. 2018

- Developed a distributed, fault-tolerant key-value store that can store the amount of data that cannot fit into one single machine, using consistent hashing