Linsong Guo

No.800 Dongchuan Road ⋄ Shanghai, China 200240
gls1196@sjtu.edu.cn

EDUCATION

Shanghai Jiao Tong University (SJTU) *2018 - 2022 (Expected)*
Bachelor of Engineering in Computer Science
Member of ACM Class, an elite CS program for the top 5% talented students.
GPA: 88/100

PUBLICATIONS

Zijun Li, Yushi Liu, **Linsong Guo**, Quan Chen, Jiagan Cheng, Wenli Zheng, Minyi Guo. FaaSFlow: Enable Efficient Workflow Execution for Function-as-a-Service. **ASPLOS 2022** .

Zijun Li, **Linsong Guo**, Jiagan Cheng, Quan Chen, Bingsheng He, Minyi Guo. The Serverless Computing Survey: A Technical Primer for Design Architecture. **ACM Computing Surveys**.

RESEARCH EXPERIENCE

Database System Group, Pennsylvania State University
Supervised by Prof. Xie Dong *Jun 2021 - Present*

Adaptive Functions Placement for Disaggregated Storage Datacenters

I concentrate on the two issues under disaggregated storage architecture:

- Functions running in the compute server need several data transmissions including *get()/put()* with the storage server, which hurts end-to-end **latency** of functions.
- The storage server wastes some CPU on processing these *get()/put()* network requests, which hurts the **throughput** of both the compute and storage server.

Moving functions from the compute server to the storage server benefit their end-to-end latency by minimizing their network interactions with the storage servers. Also, this could save the storage server's CPU, thereby improving throughput. However, the throughput is hurt if the saved CPU is not enough to handle the computation introduced by these moved functions. I am developing the placement model and extending it to the distributed environment.

Emerging Parallel Computing Center (EPCC Lab), SJTU
Supervised by Prof. Quan Chen *Jul 2020 - Jun 2021*

Eliminating Cold Startup in Serverless Computing by Sharing Containers between Functions

- Proposed an effective **inter-function container sharing policy based on startup frequency**, which helped our system to alleviate 87.9% of cold startup.
- Implemented the majority of the system, designed and ran experiments, including a large-scale one to evaluate elimination in real-world applications.

Optimizing Data Communication across Functions within Serverless Workflows

- Proposed **a QoS-aware workflow partitioning policy** that divides a workflow into several groups.
- Developed **group-level granularity scheduling**, reducing the data transmission overhead in real-world stateless workflows by 50.1%.
- Designed **an adaptive storage library** that selects the most appropriate storage service between local memory and cloud database for user functions.
- Built a parser capable of flattening hierarchical workflows into DAGs for better scheduling.

OTHER EXPERIENCE

Teaching Assistant of C++ Programming Course, SJTU
instructed by Prof. Huiyu Weng *Sep 2019 - Jan 2020*

Assigned a set of problems every two weeks, delivered a lecture about the introduction to C++ programming, guided a group of students in programming and algorithms, and contributed one-third of the problems to the final exam.

Member in ACM-ICPC Team, SJTU
guided by Prof. Yong Yu *Jun 2018 - Jul 2019*

I was a member of a team named *Quasar*. In this team, I practiced programming and algorithms with two other members at least twice a week. We earned three gold medals (one as a **1st runner-up**) in ACM-ICPC Asia regional contests and one gold medal in China Collegiate Programming Contest. Therefore, my programming and algorithmic abilities have been improved in the ACM-ICPC team.

PROJECTS

Java-and-C-like Language Compiler (~16K lines in Java) [github]

The compiler can convert a piece of code to an AST, then to LLVM IR, and eventually to RISC-V assembly. I enhanced the compiler's back-end with numerous optimizations, including mem2reg, inlining, CSE(Common SubExpression Elimination), LICM(Loop Invariant Code Motion), SCCP(Sparse Conditional Constant Propagation), and so on.

Replicated KV Store Based on Raft Consensus Protocol (~1.8K lines in C++) [github]

The replicated store could run on a cluster of servers communicated via gRPC and support basic operations such as get and put.

RISC-V CPU with 5-Stage Pipeline (~3.7K lines in Verilog) [github]

To gain a better understanding of computer architecture, I added components including d-cache, i-cache, and a branch predictor combining BTB and BHT. The CPU could run successfully on an FPGA board.

HONORS AND AWARDS

1st Runner-Up, ACM-ICPC Asia Regional Contest, Nakhon Pathom Site	*2018*
Gold Medal, ACM-ICPC Asia Regional Contest, Qingdao Site	*2018*
Gold Medal, China Collegiate Programming Contest, Guilin Site	*2018*
Silver Medal, China Collegiate Programming Contest, Final	*2018*
Gold Medal, ACM-ICPC invitational Contest, Xi'an Site	*2019*
Silver Medal, National Olympiad in Informatics	*2017*
Zhiyuan Honorary Scholarship, Award for top 5% students	*2018, 2019, 2020*
Excellence Scholarship for Undergraduates	*2019, 2020, 2021*

SKILLS

Programming Languages: C/C++ > Python > Java > Rust, x86 and RISC-V assembly

Hardware: Verilog, PC assembly