Zhanghan Wang

University of Science and Technology of China – Hefei – China

+86 150 5958 9255 • wzh895650485@mail.ustc.edu.cn • RabbitWhite1

Education

University of Science and Technology of China **September 2018 – June 2022**

Undergraduate

Major in Computer Science

- **Performance**: Major: **4.06/4.3** or **93.2/100** , Overall: **4.00/4.3** or **92.43/100**, Ranking: **2/178**
- **Computer Science Core Courses**:
  - Mathematical Logic(98%);Graph Theory(92%);Algebraic Structure(93%);
  - Fundamental of Artificial Intelligence(90%);
    Foundations of Algorithms(95%);Data Structure(90%);Programming Desgin I, II(both 100%);
  - Computer Networks(98%);An Introduction to Database System(96%);
    Operating Systems(97%);Design Methods of Embedded System(100%);Introduction to Computer Systems(H)(100%);
    Introduction to Software Engineering(92%)
  - Computer Organization(91%);Analog and Digital Circuits(91%)
- **Mathematical Core Courses**:
  - Stochastic Processes B(96%);Linear Algebra(100%);Probability and Statistics(95%)

Standard English Test

TOEFL: $Total$: 103/120, $Reading$: 30/30, $Listening$: 25/30, $Speaking$: 22/30, $Writing$: 26/30
TOEFL: $Best$: 104/120, $Reading$: 30/30, $Listening$: 25/30, $Speaking$: 23/30, $Writing$: 26/30

Academic Experience

Locating Contention in Database-based Application **July 2021 - Present**

- **Advisor**: Prof. Jinyang Li(NYU, US) and Prof. Zhaoguo Wang(SJTU, China)
- **Introduction**: There are a lot of contention in database-based application that incur deadlocks and terrible performance. However, we found that some of them might be avoided by slightly modify the source code. Here we proposed a solution combining fuzzing and concolic execution that can help developers locate the code resulting in deadlocks.
- **Contribution**:
  Part of the codes implementation
  Survey related techniques

Locating Contention in Database-based Application **May 2021 - Sept 2021**

- **Advisor**: Prof. Jialin Li(NUS)
- **Introduction**: Facing highly skewed requests, distributed storage system may suffer from a performance degradation. Surprisingly, The emerging programmable switch can help mitigate this problem. One of the most effective solutions is Pegasus(OSDI'20), which introduces an in-network coherence directory. However, the current implementation only supports static hot keys(specified in advance). I contributed to this work by completing implementation of several important parts, which are most complicated parts in Pegasus.
- **Contribution**:
  Build a virtual hosts environment to simulate
  Upgrade its code from P4-14 to P4-16
  Expand it to support dynamical hot keys

Reinforced Causal Explainer for Graph Neural Networks **December 2020 - February 2021**

- **Advisor**: Dr. Xiang Wang (NUS, Singapore)
- **Introduction**: Understanding why graph neural networks (GNNs) give such predictions is significant for putting them to practical use. In this work, we take advantages of reinforcement learning to explain GNN predictions. The corresponding paper is now under reviewing of TPAMI.
- **Contribution**:
  Implementation of the main codes
  Analysis and Visualization of results

Deep Left Join Query Optimization in OpenLooKeng **October 2020 - December 2020**

- **Advisor**: Prof. Cheng Li (USTC, China)
- **Introduction**: Deep-left-join-like queries are very common in database queries. However, this pattern can't be recognized by OpenLooKeng's optimizers. We proposed a greedy algorithm to figure out the best join order based on the sizes of input tables etc. We also try to reuse the intermediate result to accelerate queries. In the end, we achieved a 1.36 times optimization and won the first prize of the optimization contest.
- **Contribution**:
  Benchmark testing and results visualization
  Implementation of left-deep-join optimizer

Graph DataBase File System — March-July 2019

- **Advisor**: Kai Xing (USTC, China)
- **Introduction**: The most usual file systems are based on directory structures, which sometimes might be inconvenient. Based on Neo4j, a graph database, we proposed a novel **G**raph **D**ata**B**ase **F**ile **S**ystem (GDBFS) by representing files in the form of graph and connecting files according to their relevancy degree, in which we take the inner features of files, like keywords of texts, into account. We also provide a delicate web UI for better user experience.
- **Contribution**:
  The FUSE(Filesystem in Userspace) implementation of GDBFS
  Framework of the web UI based on Django

Scholarship and Awards

Scholarship

- 2019-2020 National Scholarship — October 2020
- 2018-2019 National Scholarship — October 2019

Awards

- The First Prize of OpenLooKeng(a distributed SQL query engine) Optimizatoin Contest, in CCF Computing Intelligence Contest (BDCI) — January 2021
- The Third Prize of International Parallel Computing Challenge (IPCC) — December 2020
- MCM/ICM Mathematical Modeling Contest, Meritorious Winner(6%) — Spring 2020

Technical Knowledge

Advanced Knowledge

- Language: C, C++, Python

Intermediate Knowledge

- Language: Java, Bash, Verilog, SQL
- Framework: Pytorch, Presto(Trino)

Basic Knowledge

- Language: HTML/CSS/Javascript, Rust
- Framework: Tensorflow, Django, LLVM, MLIR

Interests

Distributed System

- Accelerating distributed system
- Consensus Protocols

Database System

- Query optimizations