Ao Shen

shen634@purdue.edu | https://github.com/ShenAo1111

Education

Purdue University(transfer) West Lafayette, IN

Bachelor of Computer Information Technology; Minor Maths (GPA 3.69) *Aug 2022 – May. 2025*

Research Interests

Machine Learning System, Machine Learning for System, GPGPU Architecture.

Skills

Programming Languages & Software Tools: Python, C++, Cuda, Pytorch, Nsight System, Hugging Face related libraries.

Languages: Mandarin (Native), Cantonese (Very Fluent), English (Fluent). Served as an interpreter at the world's largest trading fair.

Honors and Awards

Dean's List 2022, 2023

Major Contributor and Admin of CUDA Group Chat, AI Inference and Deployment Group Chat, AI Architecture Group Chat

Research Experience

Full Time Research Assistant Shanghai AI Lab/Qizhi Institute

Advisor: Mingyu Gao, Assistant Professor at IIIS, Tsinghua University *May 2023 – Aug. 2024*

Project: LLM Inference System Research and Development: FastSwitch

- Finished as a first-author paper. Under guidance, developed an inference system based on vLLM, designed for high-frequency preemption and multi-round conversation.
- Asynchronous Operator Dispatch with Multi-threading, Multi-stream, and Graph Integration: Overcame Python's GIL limitations by enabling asynchronous custom operator dispatch through multi-threading in C, allowing full overlap of IO and COMPUTE operations.
- Task Handling and Conflict Resolution: Improved system efficiency by separating context switch tasks from normal storage tasks. Designed a clean cache reuse strategy, effectively reducing context switch load.
- Advanced System Level Memory Management: Integrated the existing paged attention-based strategy with a prediction-driven approach, and explored a BCF-based memory fragmentation management method. This improved task memory continuity and combined the benefits of dynamic memory pools and buddy allocators, enhancing I/O speed without additional overhead.
- Dynamic Priority Services: Explored the potential of integrating our technology in service scenarios with rapidly changing priorities to ensure fairness.
- Impact: FastSwitch achieves a speedup of 1.4×–11.2× in TTFT and TBT tail latencies across percentiles, with approximately no more than 1%additional call stack overhead.

Project: Neural Architecture Search (NAS) Integration Research and Development

- Objective: Integrate Neural Architecture Search (NAS) with program transformations, such as loop reconstruction, from a systems perspective. Detailed information in the arXiv paper [2304.07741] titled "Canvas: End-to-End Kernel Architecture Search in Neural Networks" can be referred to.
- Kernel Selection: Focused on selecting optimal kernels based on numerous results. Reproduced multiple DARTS and related works, such as EoiNAS, and applied them to the work for rapid kernel selection. Aimed to identify over 100 high-precision kernels from a pool of over 100,000 fast kernels.
- Results: Small-scale experiments demonstrated a speedup of more than 2x.

Paid Research Assistant Purdue University

Advisor: Baijian Yang, Ph.D, Professor, Purdue University *Dec 2022 – Mar. 2023*

- Implemented and deployed the buffer overflow attack lab and ROP attack lab with containerization technology such as Docker & K8s, allowing users to complete the learning of common network attacks and defenses without the need for very large resources.

Member Taiyuan University of Technology

ASC competition with First Prize Paper *Nov 2021 – Mar. 2022*

- Utilized SSH Communication between two nodes, and created NFS Shared Storage in two nodes.
- Fully responsible for building a software development environment for HPL & HPCG.
- Tested our equipment's computing ability as close to the official data as possible through HPL and HPCG by controlling variables and reached the official performance by 95%.
- Built software development environment for the DeepMD kit which is software realized in C++ and used in Molecular Dynamics.
- Leveraged parallel optimization based on the underlying source code principles and relevant disciplinary background.
- Utilized some CPU parallel programming optimization such as Avx vectorization and OpenMP programming. By using the thoughts and functions of OpenMP, the speed increased by 25%. And also optimized the code structure, removed some redundant code, and raised the performance by 50%.