Zhang Cao

tomcaottt@gmail.com | tom-caozh.github.io | Tom-CaoZH | Zhang Cao

Education

Xidian University Sep 2021 – June 2025

B.Eng in Electronic Information Engineering Xi'an, China

- **GPA**: 3.7/4.0 (86.4/100), Major GPA: 3.8/4.0 (88.17/100), Rank: 17%.
- **Awards**: Second-Class Scholarship of Xidian University in 2022 and 2023 (Top 10%).
- **A+ Courses**: Data structures and Algorithms, Introduction to Computer Science, Advanced Mathematics, and other several compulsory courses.

Research Interests

I am widely interested in system. Former, my focus is on **optimizing the Secondary Cache for LSM-KV stores**. Specifically, I worked towards optimizing the existing flash-based secondary cache for LSM-KV stores and also tried to replace the regular SSDs using ZNS SSDs as the backend of the persistent cache. **Currently, I'm looking to focus on storage, memory system and system for machine learning.**

Publications

- **[MSST'24] Zhang Cao**, Chang Guo, Ziyuan Lv, Anand Ananthabhotla, Zhichao Cao "SAS-Cache: A Semantic-Aware Secondary Cache for LSM-based Key-Value Stores." *The 38th International Conference on Massive Storage Systems and Technology (MSST), Research Track Full Paper, 2024,* To Appear.
- **[HotStorage'24]** Chongzhuo Yang, **Zhang Cao**, Chang Guo, Ming Zhao, Zhichao Cao "Can ZNS SSDs be Better Storage Devices for Persistent Cache?" *The 16th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage), 2024,* To Appear.
- **[DSDE'24]** Chongzhuo Yang, Baolin Feng, **Zhang Cao**, Zhichao Cao. "HyzoneStore: Hybrid Storage with Flexible Logical Interface and Optimized Cache for Zoned Devices." *The 7th International Conference on Data Storage and Data Engineering, 2024,* To Appear.

Research Experience

SAS-Cache: A Semantic-Aware Secondary Cache for LSM-KV Stores Aug 2022 – Jan 2024

Accepted by **MSST'24** as the **Primary Author** Advisor: Zhichao Cao

- **Background:** When LSM-KV stores are deployed on distributed storage systems (like HDFS), it suffers high read latency due to I/Os through the network. Meta officially propose secondary cache for RocksDB.
- **Motivation:** For current secondary cache, it has three problems. First, its lookup and insert are not selectively. Also compaction-triggered cache invalidation problem is more severe in the secondary cache senario. Second, There exists invalid blocks evicted from block cache and inserted into secondary cache. Third, There exists large amount of invalid items retained in the secondary cache.
- **Challenges:** There are three corresponding challenges. First, How to design a novel cache filter which can sync with the cache content with low overhead. Second, How to prevent invalid blocks from being inserted into the secondary cache with low memory assumption. Third, How to identify and handle the invalid blocks retained in the secondary cache with low overhead, and compensate for the decreased hit ratio.
- **Designs:** We propose SAS-Cache which supports selectively insert and lookup and solve the cache invalidation problem. It employs **LSM-Managed Cache Filter** to filter unused lookup, **Valid SST-Aware Admission Policy** to reduce unused insertions and **Compaction-Aware Cache Replacement** to solve the cache invalidation problem. Cache Filter and Admission Policy take little memory consumption and Compaction Replacement is asynchronous.
- **Implementation and Engineering Difficulties:** Implement low-overhead Cache Filter and Admission Policy. Modify RocksDB's source code to collect runtime compaction information and implements multi-level prefetching methods to enable compaction invalidated block replacement.

ZNS-Cache: A better Persistent Cache for Cache Systems Jan 2024 – March 2024

Accepted by **HotStorage'24** as the Second Author Advisor: Zhichao Cao

- **Motivations:** Compared with regular SSDs, ZNS SSDs eliminate the uncontrollable garbage collection (reducing write amplification) and don't need over-provisioning space (offering larger space). These two advantages benefit cache systems. Thus, we explore utilizing ZNS SSDs as the persistent cache.
- **Three Schemes:** We present three schemes to utilize ZNS SSDs as the persistent cache and then analyze their benefits and limitations. The first scheme is to utilize a ZNS-compatible file system like F2FS to handle the operations that interact with ZNS SSDs. The second is to directly map the cache on-disk management unit to the fixed-size zone; this method avoids the overhead of management. The third is to implement a middle layer to manage the interaction between the cache system and ZNS SSDs.

- **Implemenation and Evaluation:** We utilize CacheLib as the cache system to conduct evaluation and also integrate it with RocksDB as its secondary cache to further evaluate the schemes. The evaluation results show the benefits of using ZNS SSDs as the persistent cache and also the limitations and benefits of the three schemes.

Explore CXL-based Memory System Oct 2023 – Now

All the exploration is recorded in the Github Repo: CXL-101 Advisor: Zhichao Cao

- Understand the CXL protocol specifications (CXL 1.1, 2.0, and 3.0) and their application scenarios.
- Use QEMU and NUMA nodes to simulate the CXL-based memory system and evaluate its performance using micro-benchmarks like STREAM and end-to-end benchmarks like YCSB.
- Compare CXL-based memory systems with traditional memory systems like PMem and RDMA-based disaggregated memory systems in terms of research directions and summary current research on CXL-based memory systems.
- Explore potential research directions (on-going work).

HyzoneStore: Hybrid Storage for Zoned Devices April 2023 – June 2023

Accepted by **DSDE'24** as the Third Author Advisor: Zhichao Cao

- **Motivations:** Hybrid storage systems are promising to utilize the benefits of both SMR HDDs and ZNS SSDs. However, existing hybrid zoned storage cann't support felxible zone size allocation. We aim to build a general high performance hybrid storage system for zoned devices.
- **Designs:** We propose HyzoneStore, a hybrid storage system that supports flexible zone size allocation. It employs a Logical Interface to decouple the physical zone size and the data size. For high performance, it employs FIFO Log Tier to accelerate the write performance and a Read Cache Tier to accelerate the read performance of SMR drivers.
- **Implementation and Evaluation:** We implement HyzoneStore using RUST and evaluate the effectiveness of FIFO Log Tier and Read Cache Tier designs.

Personal Projects

LLM-NER: Finetune LLM Model for NER Task Jan 2024 – Feb 2024

LLM Finetuen and NER (Named Entity Recognition) Task LLM-NER

- NER is a sequence labeling task that aims to identify named entities in text. We transfer the sequence labeling task to a text generation task, which is more suitable for LLM models.
- First, we preprocess the given NER dataset, which includes instruction designs. Then, we fine-tune a total of six LLM models (like Baichuan, Mistral, etc.) using the LLaMA-Factory framework. Besides, we utilize the LoRA and qLoRA methods to fine-tune the models on a single 4090 GPU. Lastly, we evaluate the models on the custom dataset by calculating their F1 scores.
- For a single type of named entity, the best model achieves an F1 score of 0.85 on our custom dataset. For a total of six types of named entities, the best model achieves an F1 score of 0.60.

ChFS: A Distributed File System Dec 2022 – Jan 2023

Filesystem and RAFT ChFS

- Design and implement a user space **inode-based file system** using **FUSE** with support for basic file operations such as **CREATE/MKDIR, READ, WRITE, UNLINK, SYMLINK, READLINK** and etc.
- Implement **Redo log and Undo log** to enable each file operation atomic and persistent when system crashes and **checkpoint** mechanism to reduce the log file size.
- Enable the file system to be accessed by multiple clients by decoupling client and server through **RPC**. Implement a **lock server and two-phase locking (2PL) protocol** to ensure concurrency across clients.
- Use **Raft** to ensure fault-tolerance and achieve high availability with strong consistency.

XV6: A re-implementation of Unix v6 OS April 2022 – May 2022

RISC-V and OS XV6

- In-depth understanding of operating system architectures and concepts such as: **Page tables, traps, system calls, interrupts, locking mechanisms, and scheduling algorithms** and etc.
- Optimized the XV6 kernel by implementing **copy-on-write** to improve fork performance.
- Expanded the XV6 kernel capabilities by enabling support for **larger files and symbolic links**.
- Improved memory allocation efficiency by implementing **lazy allocation** for the sbrk command.
- Reduced context switching overhead in XV6 kernel by implementing **user-level threads**.

Skills

Programming Skills: Familiar with C++, Rust and Python.
Tech Skills: LSM-KV stores(LevelDB, RocksDB), Compute Express Link(CXL), Zoned Storage(SMR HDD and ZNS SSD), Distributed File System and other basic skills like Linux, OS, Git and etc.
LLM Related Skills: LLM inference system (llama.cpp), LLM Finetune (based on LoRA).
Language: CET-4: 550, CET-6: 520. (Passing Score: 425)