Enming Guo

858-319-5558 | enguo@ucsd.edu | linkedin/EnmingGuo | github.com/EnmingGuo | enmingguo.github.io

Education

University of California, San Diego — La Jolla, CA
Master of Computer Science — *09/2023 - 03/2025*

Northeastern University — Shenyang, China
Bachelor of Computer Science & Technology (GPA:4.435/5.0, Ranking:No.1/221) — *09/2019 - 07/2023*
Awards: *Baosteel Outstanding Student Award (Top 0.02%), National Scholarship (Top 1%), 10 contest awards.*

Experience

Amazon Web Service — Jun. 2024 – Sep. 2024
Software Dev Engineer Intern | **C, Aurora PostgreSQL, EC2** — *Seattle, WA*

- Implemented Aurora PostgreSQL Optimized Reads with Tiered Cache (TC) as an intermediate cache between the buffer cache and storage, improving query latency by **8x** and reducing costs by **30%** on NVMe storage instances.
- Developed an internal Tiered Cache mechanism based on PostgreSQL C source code, enabling asynchronous batch writes of pages eligible for eviction from the Buffer Cache and evicting cold pages from TC.
- Designed **Read-Only (RO)** and **Read-Write (RW)** nodes with Survival-RO Reload algorithm, ensuring consistent data transfer from RW to RO nodes, achieving efficient load balancing and multi-node consistency.
- Implemented **user-driven** TC management interfaces, supporting relation-level TC search, caching, and eviction. Caching utilized three concurrent processes to batch process relation pages, reducing execution time.
- Utilized **locking mechanisms** to manage internal asynchronous TC writes and user-driven relation-level TC writes, ensuring efficient multi-process execution and improved performance.
- Led comprehensive performance testing on **EC2** instances (r6gd.xlarge, r6gd.4xlarge, and r6gd.16xlarge), utilizing TB-scale datasets to validate the effectiveness of the concurrent relation caching algorithm. Achieved up to **51.8%** improvement in TC caching efficiency compared to the previous asynchronous background TC writes mechanism.

Projects

Parallel Computing Acceleration | *CUDA, MPI, OpenMP* — Sep. 2024 – Dec. 2024

- Optimized CPU-based matrix multiplication in C with cache hierarchy, data packing, and the SVE instruction set, improving performance from 2.7 to **24.6 GFLOPS** on c7g.medium.
- Optimized GPU-based matrix multiplication in CUDA with memory coalescing, shared memory, and OpenMP, improving performance from 403 to **4160 GFLOPS** on g4dn.xlarge.
- Parallel Aliev-Panfilov model using MPI and AVX2, with balanced data distribution and ghost cell sync. Achieved sublinear scalability up to **3199 GFLOPS** across **384 cores** on the Expanse supercomputer.

Fault Tolerant Distributed File System | *Go, gRPC* — Apr. 2024 – Jun. 2024

- Developed a scalable, Dropbox-like file storage system in **Go** with **gRPC**, employing **channels**, **goroutines**, and **mutex locks** to manage cluster behaviors and synchronize file updates.
- Utilized **consistent hashing** to distribute data across BlockStores, providing load balancing across 1,000 servers.
- Implemented the **RAFT** protocol to ensure data consistency across multiple clusters and guarantee the correctness of the MetaStore during server failures and network partitions.

Database Management System | *C++* — Jan. 2023 – Mar. 2023

- Implemented a Relational Database System in **C++** supporting records operations; database catalog and schema update; indexing and querying. Tested with over **100000** records/operation that finished in **60** seconds.
- Developed a paged-file manager using a **Heap file** structure with slot directory for efficient variable length records management. Achieved **O(1)** record access through optimized slot indexing.
- Developed an index manager using B+ Tree for indexing and conditional scanning on various attributes, optimizing queries, insertions, and deletions. Achieved logarithmic time complexity, resulting in a **60%** reduction in query response time and a **30%** improvement in data insertion speed compared to linear search methods.
- Devised a query engine that supports filter, aggregation, block/index nested-loop join and grace hash join.

Technical Skills

Languages: C/C++, Java, Python, Go, Scala, SQL, Matlab, JavaScript, HTML/CSS, R, MQL
Frameworks: Vue.js, Springboot, Hadoop, Django, ElasticSearch, Spark, LayUI
Developer Tools: Git, Docker, AWS, Bash Shell, Kubernetes, Vim