Mingyong Ma

Email：m7ma@ucsd.edu **Phone**: 858-539-6919 **Website**: https://incandescent-licorice-a37843.netlify.app/

EDUCATION BACKGROUND

2022.9-2023.12(expect) **University of California, San Diego** **Master of Science in Computer Science**

SKILLS SETS

Programming Language：Go, C++, Python, HTML, CSS, JavaScript, Java

Skills: Network Socket, Distributed System (Raft), Database(B+ Tree), Spark Analysis, image processing, time series forecast

WORKING EXPERIENCE

2023.6-2023.09 **Adobe Software Engineer Intern** **Bay Area**

- Building Adobe primary AI platform infrastructure, improve API call from **sync** to **async**, and **reduce the network I/O** from **1GB** to **9MB** per inference call. Using **Jmeter** for load-testing, able to generate **1600 TPS (token per second)**.
- Implementing using **REST API** that able to **CRUD** a specific task, and save it in **postgres DB** with **almebic** version control.
- Generating a **docker** container for each **fine-tuning** task, able to **scheduling** according to the load on **Azure cloud**.
- Facilitate the fine-tuning of **FlanT5** and **llama2** models, with the capability to independently store the base model and fine-tuning layer utilizing **PEFT**.

2022.7-2022.08 **Amazon Software Engineer Intern.** **Shenzhen**

- Developed an image processing system that combines **deep learning** techniques with the **Unsharp** algorithm, achieving superior results compared to the camera algorithm used in tablets. And evaluated the performance of the system using **MTF-50**

2021.11-2022.02 **Lenovo Digital Transformation Department Data Analytic Intern** **Beijing**

- Conducted time series forecasting to predict future sales of Lenovo's notebook products and tablets, utilizing Lenovo's historical sales data as well as data from other companies such as IDC and GFK.
- Increased the forecasting accuracy of the model by 4.2% by implementing ML algorithms **Prophet** and DL models like **LSTM**.

PROJECT EXPERIENCE

2023.03-2023.06. **B+ Tree with Buffer Management (detail in my website)**

- Build a **Buffer Pool** on top of I/O layer, and realize **Buffer Replacement Policy** and **LRU clock algorithm**.
- Build a **B+ Tree** on top of **Buffer Pool**, supporting INSERT/DELETE operation. Besides, it allows inserting int, double, char* variable and support more than three layer of Leaf/nonLeaf nodes.
- Initializing with **Bulk Loading**, and split a page when sizes exceed "**fanout**".

2023.01-2023.03. **Distribute Cloud File System(Go)**

- Implementing the **http protocol** (simple version):
  - Clients send request messages to the server, and servers reply with response messages layered on top of the **TCP** protocol.
  - Implemented **HTTP persistent connection**: a client can reuse a TCP connection to a given server
  - Provide **safe control** not allowing clients to access memory other than document root.
- creating a **fault tolerant cloud-based file storage service** called SurfStore (client and server communicating using **gRPC**):
  - The SurfStore service is composed of the following two services:
    - BlockStore: Stores these **blocks**, and when given an identifier, retrieves and returns the appropriate block.
    - MetaStore: Manages the metadata of files and mapping of filenames to blocks (hash marshalled by **SHA-256**).
  - The clients' file data is stored in local database with **version**. When invoking into client, the sync operation will occur, and new files added to base directory will be uploaded to the cloud, files that were sync'd to the cloud from other clients will be downloaded to base directory, and any files which have "edit conflicts" will be resolved.
  - Store and manage the block in different BlockStore using **Consistent Hashing Ring**.
  - Ensure that the MetaStore is fault tolerant and stays consistent regardless of minority of server failures by **RAFT** protocol.

2022.9-2022.12. **Operating System Implementation**

- Implementing internal structures of the operating system: Alarm() function to call timer interrupt; Join() function to sleep the parent while waiting for the child thread to finish; Implement **semaphores** to provide atomicity;
- create the pageTable data structure for each user process, which maps the process's virtual addresses to physical addresses.
- Implementing the file system calls create, open, read, write, close, unlink, join, exit and exec.
- Implement demand paging, page replacement to free up a physic page to handle page faults.