Mingyong Ma

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

EDUCATION BACKGROUND

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

WORKING EXPERIENCE

2023.6-2023.09 Adobe Software Engineer Intern Generative AI Infra team

- Integrated **LLM** model **fine-tuning** and **inference** features to Adobe primary AI platform **Firefall**.
- User can submit a **fine-tuning** task to **Firefall**, Firefall will forward that request to **MMS** (Model Management System), which will download our code from remote artifactory **JFrog** and build a **Docker** container for that task. After fine-tuning task is finished, it will save the fine-tuned model in **Azure blob storage**, and returns the results back to the user.
- Improved the fine-tuning API call from **blocking** to **asynchronous**. Users no longer need to wait for the fine-tuning results; they receive a task_id instantly when submitting a fine-tuning task, which can be used to query **Firefall** for the results. This modification has reduced **overhead** and boosted **lantency by 90%**.
- **Reduced the network I/O** from **13GB** to **32MB** per **inference** call. By utilizing **PEFT**, the based model is consistent for every fine-tuning job, thus is stored in **in-memory-buffer** of the **Docker** container, with only the **Lora layer** being stored in **Azure blob storage**. Therefore, only the Lora layer (32MB) instead of the entire model (13GB) is downloaded into Docker container.
- Used **Jmeter** for load-testing, able to generate **1600 TPS (token per second)** with **multi-threading**.
- Innovatively proposed how to fine-tune **LLaMa2-7b** on a **CPU**, which offers alternative choice to save cost. (No need to run GPU entirely a day).
- Implementing using **REST API** that able to **CRUD** a specific task, and save it in **postgres DB** with **almebic** version control.

2022.6-2022.08 Amazon Software Engineer Intern Device team

- Developed an image processing algorithm 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**
- Demonstrated ability to compare the performance of algorithms by controlling the imaging device with **adb** and generating identical images with different image sharpening settings in the Amazon lab.
- ·Conducted an evaluation of our proposed algorithm using Imatest software in the Amazon lab, observing an increase in MTF-50, which showcases an improvement in image sharpness.

2021.11-2022.02 Lenovo Data Analytic Intern Digital Transformation team

- 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 machine learning algorithms such as **Prophet** and deep learning models like **LSTM** or **GRU**.
- Optimized hyperparameters of the existing code using **Optuna**, resulting in significant time savings (4x) compared to traditional **grid search methods**.

PROJECT EXPERIENCE

2023.03-2023.06. B+ Tree with Buffer Management

- Realized a Database index method utilizing **B+ Tree**, which shows **10 times faster** performance compared with **Hash index** or **file scan** on **range search**.
- Built a **Buffer Pool** on top of I/O layer, and realize **Buffer Replacement Policy** and **LRU clock algorithm**.
- Built a **B+ Tree** on top of **Buffer Pool**, supporting **CRUD** operation. Besides, it can save more than **50GB** data.

2023.01-2023.03. Distribute Cloud File System(Go)

- Created a **fault tolerant cloud-based file storage service** called SurfStore (client and server communicating using **gRPC**):
- Stored and manage the block in different BlockStore using **Consistent Hashing Ring**.
- Ensured 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

- Implemented **life cycle** of the OS process, **virtual memory** and **file system**.
- Create the **pageTable** data structure for each user process, which maps the process's virtual addresses to physical addresses.
- Implement demand paging, page replacement to free up a physic page to handle page faults.