Yingtong Yu

JARVISHHH.github.io yingtongyujobs@gmail.com https://github.com/JARVISHHH

Education

Brown University **Sep. 2022 – May 2024 (Expected)**
Master of Science in Computer Science *Providence, RI*

Courses taking: **Computer Graphics**, **2D Game Engines**

Nankai University **Sep. 2018 – May 2022**
Bachelor of Engineering in Computer Science and Technology - GPA: 3.78/4.00 *Tianjin, China*

Technical Skills

Languages: C/C++, Golang, Java, Python, Shell, SQL
Developer Tools: VS Code, Visual Studio, Goland, Anaconda, Virtual Box, Vim, Grafana, Qt Creator
Technologies/Frameworks: Linux, Git, Elasticsearch, Thrift, RPC, JavaFX, Flask, SQLite, CUDA, OpenGL

Work Experience

ByteDance(TikTok) **Apr. 2022 – Jul. 2022**
Back-end Software Development Engineer Intern | Golang *Beijing, China*

- Reduced the latency of the packing part of the Suggestion Middle Page **from 160ms to 10ms** by reducing the number of RPC calls and parallelizing different processes, and increased the speed by about **1500%**.
- Refactored an entire API service, making it more readable and extensible.
- Added metrics and AB test, built **Grafana** dashboards to visualize performance.
- Implemented Pinyin fuzzy search and supported the proximity-based filtering with **Elasticsearch**.
- Integrated the new version of the recommendation engine and provided more informative search bar options, such as property type of real estate, tips to switch cities, etc.

Projects

Parallelization of Triangular Raster Anti-aliasing Algorithms | *C++* **May 2021 - Jun. 2021**

- Studied the parallelization of a dominant anti-aliasing algorithm and achieved a **7-time** speed-up on the algorithm.
- Added the code of the low-pass filter based on the stencil code of the GAMES101(Assignment 2).
- Parallelized Fourier Transform and the multiplication operation in the frequency domain, using **loop expansion**, **multithreading**, **multi-node** technologies.

Design of a Reliable Transport Protocol based on UDP | *C++* **Nov. 2020 - Dec. 2020**

- Implemented a connection-oriented reliable data transmission protocol in user space based on UDP, including connection establishment, error detection, retransmission confirmation, etc.
- Integrated the **Go-Back-N (GBN)** algorithm based on the sliding window for the flow control mechanism. Used fixed window size and achieved cumulative confirmation in the algorithm. Improved the average throughput rate by **908.9%**, comparing with Stop-and-Wait(SW).
- Integrated the improved **TCP Reno** algorithm for congestion control. Enhanced the average throughput rate by **252.8%**, comparing with the application without a congestion control algorithm.

Search Engine based on Vector Space Model and PageRank | *Python* **Nov. 2020**

- Designed and implemented a search engine that is able to perform phrase queries, site queries, wildcard queries, document queries, etc., which can return the 10 most matched pages.
- Used **BeautifulSoup** and **Urllib** to crawl the related web pages of Nankai University to obtain related web URLs, text content, anchor text, etc.; established the index and query using the **Whoosh** library.
- Calculated the similarity between web pages and the query based on the **Vector Space Model** using **Scikit-Learn** and **NumPy**, integrated the result (the score of the Vector Space Model) with the **PageRank** score to obtain the final score of web pages and determined their ranking.

Recommendation System Based on Collaborative Filtering | *Python* **Apr. 2020 - May 2020**

- Implemented a product recommendation system by predicting users' ratings for the products based on **Singular Value Decomposition(SVD)** as the leader of a team of 3.
- Used SVD to fit the training and the regularization items to prevent overfitting.
- Improved the recommendation quality by comparing the deviation of the global average score, user average scores and commodity average scores.
- Employed the grid parameter adjustment method to obtain the ideal parameters.
- Improved the algorithm and reduced the overall Root Mean Squared Error(RMSE) **from 35 to 26.3** by making full use of the given data set.