Log File Analysis

Balaaditya Matcha

24B0971

CS104

April 2025

Contents

1 Introduction to Webpage

In modern software systems, log files—especially error logs—are essential for maintaining, debugging, and monitoring applications. These files record information such as timestamps, error messages, warning levels, and system states when something unexpected occurs. They are often generated automatically by web servers, operating systems, or applications like Apache, Nginx, or Flask.

However, raw log files are hard to read and analyze due to their unstructured nature and sheer volume. Manually digging through lines of logs to identify issues or patterns can be time-consuming and error-prone.

This webpage addresses these problems by:

- Parsing raw log files and converting them into a structured, readable tabular format.
- Providing predefined visualizations through plots, enabling users to quickly spot trends or critical time periods.
- Offering a code editor for users to test and visualize custom plots from the data.
- Supporting filtering and targeted analysis via an intuitive web interface to isolate specific types of errors or time ranges.

This significantly improves the speed and accuracy of log analysis, particularly for developers, system administrators, and analysts working to maintain reliable services.

2 User Interface Layout

2.1 Upload Page

This page allows users to upload their **Apache log files (only)** for analysis.

2.1.1 File Upload

A user-friendly form is provided to upload the file. The system parses the uploaded file and informs the user whether it is readable and properly formatted.

2.1.2 Navigation to Other Pages

If the file is correctly parsed, the page displays three hyperlinks for navigation to other analysis pages.

2.2 Log Display Page

2.2.1 Filtering Option

This page includes an **additional feature** allowing users to filter the table based on timestamps.

2.2.2 Tabular Display

The parsed log file is displayed in a structured tabular format (CSV). Users can also download the CSV file for further analysis.

2.3 Plots Page

2.3.1 Filtering Option

Users can filter the data based on timestamps before plotting.

2.3.2 Various Plots

Three predefined plots are provided based on key fields from the CSV file. Each plot is available for download.

2.4 Pyeditor Page

2.4.1 Navigation to Plots Page

A button is available to open the predefined plots page in a new tab for comparison.

2.4.2 Filtering Option

Users can apply filtering before plotting their own graphs.

2.4.3 Code Editor

Users may write custom code to plot graphs and download the resulting image.

2.4.4 Output Plot

If the code is correct, the corresponding plot is displayed. Otherwise, an error message is shown.

3 Flask - Backend Workflow

3.1 Directory Structure

The project follows a modular structure to ensure clarity and maintainability. The key directories and files are described below:

- **myproject** (Main Project Directory)
  - **.venv** — Local Python virtual environment used, to isolate dependencies from the global Python installation(not tracked in version control)
  - **__pycache__** — Automatically generated Python bytecode cache created at runtime for performance optimization (not tracked in version control)

- **static** — Contains CSS stylesheets and plot images generated by users.
- **templates** — Holds the HTML files used by Flask for rendering webpages.
- **parse.sh** — Parses the uploaded Apache log file and generates a structured CSV using other helper scripts.
- **filter.sh** — Filters the log file based on timestamps to extract relevant entries.
- **table.sh** — Converts CSV data into an HTML table format for display in the browser.
- **updated.sed** — Replaces raw event names in the log file with shorter codes (e.g., E1–E6) for simplification.
- **reupdated.sed** — Replaces the event codes (E1–E6) with their full template names to enhance CSV readability.
- **plots.py** — Parses the filtered CSV file and dynamically generates plots, saving them as images.
- **web.py** — The main Flask backend file that routes requests, handles uploads, connects HTML pages, and invokes the appropriate scripts for parsing, filtering, table generation, and plotting.

3.2 Function of Each File

Each file is designed to handle a specific part of the log processing workflow:

- **parse.sh** — Initiates parsing and triggers the 'updated.sed' and 'reupdated.sed' scripts.
- **filter.sh** — Extracts relevant time windows from the parsed data.
- **table.sh** — Prepares the table representation of CSV.
- **plots.py** — Generates visual analytics for easy interpretation.
- **web.py** — Acts as the control hub integrating all components.

Modules Used

- `flask` – Web framework for backend and routing
- `subprocess` – To run shell scripts like `parse.sh`, `filter.sh`, `table.sh`
- `os` – File checks and path operations
- `io`, `base64` – For encoding plots into image data
- `matplotlib` – For plotting graphs in `plots.py`

4 Setup and Usage Instructions

Prerequisites

These modules should be downloaded before starting the server:

- Flask
- Matplotlib
- gawk

Note: Python interpreter should also be downloaded if not present. Use `python3` while running commands accordingly.

4.1 Running the Webpage from Server Side

- Step 1: Navigate to the Github repository to download the files.
- Step 2: Run the command `python3 web.py` in the terminal.
- Step 3: A Local IP address will be shown, which, when clicked, redirects to the website. Other devices on the same network can access this IP address as well.

5 Project Journey

5.1 Learnings

1. Learnt how to use Flask module and a basic idea on how web server backend works
2. Learnt the usage of sys,os,subprocess,io,base64

5.2 Challenges

1. Decreasing runtime by avoiding loops
2. Handling wierd errors by understand the text in debug mode
3. Had a lot of difficulty in securing the user info using sessions
4. Handling errors due to åppended by windows
5. Handling redirecting issues

References

[1] LogHub. *A collection of log files.* https://github.com/logpai/loghub. Accessed: 2025-04-29.

[2] LogHub. *Apache Logs.* https://github.com/logpai/loghub/tree/master/Apache. Accessed: 2025-04-29.

[3] Flask. *Flask Documentation.* https://flask.palletsprojects.com/. Accessed: 2025-04-29.

[4] Real Python. *Flask Tutorial.* https://realpython.com/tutorials/flask/. Accessed: 2025-04-29.

[5] Miguel Grinberg. *Flask Mega-Tutorial Part I: Hello World.* https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world. Accessed: 2025-04-29.

[6] Flask Module Documentation. *Flask module documentation for understanding its functionality.* https://flask.palletsprojects.com/en/stable/. Accessed: 2025-04-29.