SCHOOL OF SCIENCE & ENGINEERING

SPRING 2024

CSC 5356 01 Data Engineering and Visualization

Project#0 Unix Pipeline

January 21st, 2024

Realized by:

Khadija Salih Alj

Noura Ogbi

Supervised by:

Prof. Tajjedine Rachidi

Table of Contents

I. Introduction

The analysis of access logs from web servers is pivotal for gaining insights into user behavior and popular content on a specific website. This report delves into the Nginx access log file to learn how to create a data pipeline.

Diving into the project, we have developed a ***Unix shell*** pipeline, which is a series of commands where the output of each command serves as the input to the next one. Through this approach, we have crafted a systematic process to extract and analyze the frequency of page requests. This enables us to extract the top ten visited pages, along with their respective visit counts. The pipeline incorporates commands like ***cat, awk, sort, uniq, and head***, creating a smooth flow of data processing that transforms raw log data into actionable insights.

The utilization of a pipeline presents clear advantages compared to a non-pipeline method, particularly in terms of efficiency and flexibility. In the absence of a pipeline, processing tasks would require the generation of intermediate files or temporary storage, leading to potential data redundancy and increased complexity in managing interim results. Conversely, the pipeline simplifies the processing, diminishing the need for intermediate storage and enhancing the overall efficiency of the data analysis.

II. Methodology

In this section, we outline the Unix shell commands employed in the pipeline for extracting and analyzing the Nginx access log data. Each command plays a critical role in processing the log file and deriving valuable insights regarding the most popular pages accessed by users.

```
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <logfile> <pages>"
    exit 1
fi

LOG_FILE=$1
PAGES=$2

cat "$LOG_FILE" | awk -F' ' '{print $10}' | sort | uniq -c | sort -nr | head -n "$PAGES"
```

In the if statement, we check if the number of command-line arguments passed to the script is not equal to 2. If it is not, the script prints a usage message indicating the correct usage with log file, and the desired number of pages, then exits.

1. **cat "$LOG_FILE"**

Contribution: The *cat* command is used to read the content of the "$LOG_FILE" (nginx_json_logs.txt file in our case) and send its lines to the next command in the pipeline.

Rationale: This command serves as the starting point, providing the raw log data as the input for subsequent processing.

2. **awk -F' ' '{print $10}'**

Contribution: The *awk* command with the field delimiter -F' ' (space) is used to extract the 10th field from each log entry, which represents the requested page.

Rationale: By using awk, we isolate the page requests from the log entries, streamlining the data for further analysis.

3. **sort**

Contribution: The *sort* command is used to arrange the extracted page requests in alphabetical order.

Rationale: Sorting the page requests facilitates the identification of unique pages and prepares the data for subsequent frequency analysis.

4. **uniq -c**

Contribution: The *uniq* command, in combination with the *-c* option, is used to count the occurrences of each unique page request.

Rationale: By applying *uniq -c*, we obtain the frequency of each page access, laying the groundwork for identifying the most popular pages.

5. **sort -nr**

Contribution: Another *sort* command is employed with the *-nr* option to arrange the unique page requests based on their access counts in descending order, placing the paths with the highest counts at the top.

Rationale: Sorting the unique pages in descending order of access counts allows for the identification of the most popular pages at the top of the list.

6. **head -n "$PAGES"**

Contribution: The head command is used to select the top entries based on the specified number of top pages provided as a command line argument.

Rationale: Using head, we filter the results to extract the top pages along with their respective access counts as per the user's specified criteria.

III. Results

Top 10 Most Popular Pages:

The processing of Nginx access logs revealed the following top 3 most popular pages based on user access counts (even if we wanted to get 10 pages, we got 3 because the file contains only 3 pages):

```
dizalj@DESKTOP-P863LL5:/mnt/f/Semester 8/CSC 5356/Project$ ./pipeline.sh nginx_json_logs.txt 10
  30285 /downloads/product_1
  21104 /downloads/product_2
     73 /downloads/product_3
```

If we want to get the most popular page (1 only), we do:

```
dizalj@DESKTOP-P863LL5:/mnt/f/Semester 8/CSC 5356/Project$ ./pipeline.sh nginx_json_logs.txt 1
  30285 /downloads/product_1
dizalj@DESKTOP-P863LL5:/mnt/f/Semester 8/CSC 5356/Project$
```

Since the Nginx access logs indicated only three distinct pages **/downloads/product_1**, **/downloads/product_2**, and **/downloads/product_3**, we further extended our analysis to examine the top 10 most frequent visitors to the website based on their IP addresses.

Top 10 Most Frequent Visitors (by "remote_ip"):

In addition to page popularity, we investigated the most frequent visitors to the website based on their IP addresses.

```
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <logfile> <ip>"
    exit 1
fi

LOG_FILE=$1
IP=$2

cat "$LOG_FILE" | awk -F'"' '{print $8}' | sort | uniq -c | sort -nr | head -n "$IP"
```

The top 10 visitors, along with their access counts, are as follows:

```
dizalj@DESKTOP-P863LL5:/mnt/f/Semester 8/CSC 5356/Project$ ./pipelineip.sh nginx_json_logs.txt 10
   2350 216.46.173.126
   1720 180.179.174.219
   1439 204.77.168.241
   1365 65.39.197.164
   1202 80.91.33.133
   1120 84.208.15.12
   1084 74.125.60.158
   1064 119.252.76.162
    628 79.136.114.202
    532 54.207.57.55
```

This information provides insights into the user base, highlighting the IP addresses associated with the most frequent visits to the website.

IV. Conclusion

In conclusion, the successful implementation of a data pipeline has been pivotal in unlocking the potential hidden within raw website logs. This achievement not only demonstrates the power of structured data processing but also lays the groundwork for advanced personalization and recommendations, enlightening the future of data-driven website optimization.

However, our journey was not without its challenges, notably navigating unforeseen errors such as overlooking the command ‘cd /mnt.’😅 This emphasized the critical importance of mastering fundamental Unix commands in the effective execution of data pipelines. Overcoming these challenges has not only enhanced our technical proficiency but has also reinforced the resilience needed in the dynamic landscape of data processing.

Looking ahead, this project sets the stage for a future where data plays a pivotal role in predicting user needs and elevating overall online experiences. The solid foundation laid during this exploration marks the beginning of a new era, where websites can harness the power of integrated code and data to offer personalized recommendations and insights.