Algorithm Analysis

Abraham J. Reines

April 28, 2024

Results

Please find all results in the .txt files of the assignment submisstion.

Task 7.6.1

Results

Log File Name	Total Events	Error Events	Critical Events	Process IDs Reporting Events
XUNHUA2023OFFIC_Setup_log.evtx	94	0	0	18412, 12536, ...15492 (Truncated)
DESKTOP-KUHNMLA_Setup_log.evtx	366	0	0	3884, 1824, ...2892 (Truncated)
XUNHUA2023OFFIC_System_log.evtx	29692	321	8	5412, 2780, ...15484 (Truncated)
XUNHUA2023OFFIC_Security_log.evtx	22782	0	0	3596, 3632, ...1416
DESKTOP-KUHNMLA_System_log.evtx	44357	249	8	5412, 5000, ...9936 (Truncated)

Table 1: Statistical Analysis of Windows Log Files

Please note the results are captured by the `'log_statistics.txt'` file. This file will be uploaded alongn tside this PDF.

Script

```
import Evtx.Evtx as evtx
import xml.etree.ElementTree as ET
import os
import concurrent.futures

class LogFileAnalyzer:
    """
    class for analyzing Windows log files.

    Attributes:
        Wheres_our_file (str): path to log file.
        How_many_events (int): total number of events in log file.
        ProcessIDs (set): set of unique process IDs found in log file.
        Errors (int): number of error events in log file.
        Criticals (int): number of critical events in log file.
    """
```

```
    def __init__(self, Wheres_our_file):
        self.Wheres_our_file = Wheres_our_file
        self.How_many_events = 0
        self.ProcessIDs = set()
        self.Errors = 0
        self.Criticals = 0

    def analyze(self):
        """
        Analyzes log file and processes record.

        method reads log file using ‘evtx‘ library and processes record.
        """
        with evtx.Evtx(self.Wheres_our_file) as log:
            for record in log.records():
                try:
                    self.process_these_records(record.xml())
                except Exception as e:
                    # Log error and continue with next record
                    print(f"Error processing record in {os.path.basename(self.
    ↪ Wheres_our_file)}: {e}")

    def process_these_records(self, xml_content):
        """
        Processes XML of log records.

        Args:
            xml_content (str): XML content of log record.

        This method extracts information from XML and updates attributes.
        """
        XML_records = ET.fromstring(xml_content)
        self.How_many_events += 1
        for elem in XML_records.iter():
            if elem.tag.endswith('Level'):
                if elem.text == '2':  # ERROR
                    self.Errors += 1
                elif elem.text == '1':  # CRITICAL
                    self.Criticals += 1
            if elem.tag.endswith('Execution'):
                self.ProcessIDs.add(elem.attrib.get('ProcessID'))

    def get_s0me_stats(self):
        """
        Returns some statistics about log file.

        Returns:
            dict: dictionary containing following statistics:
                - 'file_name': name of log file.
                - 'Total Events': total number of events in log file.
                - 'Process IDs': set of unique process IDs found in log file.
                - 'Error Events': number of error events in log file.
                - 'Critical Events': number of critical events in log file.
        """
        return {
            'file_name': os.path.basename(self.Wheres_our_file),
            'Total Events': self.How_many_events,
            'Process IDs': self.ProcessIDs,
            'Error Events': self.Errors,
            'Critical Events': self.Criticals
        }

def analyze_this(evtx_file):
    log_Wheres_our_file = os.path.join(Absolute_path, evtx_file)
    analyzer = LogFileAnalyzer(log_Wheres_our_file)
```

```
    analyzer.analyze()
    return analyzer.get_s0me_stats()

Absolute_path = '/home/reinesaj99/Desktop/'
S0me_stats = '/home/reinesaj99/Desktop/log_statistics.txt'
EVTXFiles = [f for f in os.listdir(Absolute_path) if f.endswith('.evtx')]

with concurrent.futures.ThreadPoolExecutor() as executor:
    futures = {executor.submit(analyze_this, evtx_file): evtx_file for evtx_file in
    ↪ EVTXFiles}
    results = []
    for future in concurrent.futures.as_completed(futures):
        evtx_file = futures[future]
        try:
            result = future.result()
            results.append(result)
        except Exception as exc:
            print(f'{evtx_file} generated an exception: {exc}')

with open(S0me_stats, 'w') as This_came_out:
    for result in results:
        This_came_out.write(f"Stats for {result['file_name']}:\n")
        This_came_out.write(f"Total Events: {result['Total Events']}\n")
        This_came_out.write(f"Process IDs: {', '.join(map(str, result['Process IDs'])
    ↪ )}\n")
        This_came_out.write(f"Error Events: {result['Error Events']}\n")
        This_came_out.write(f"Critical Events: {result['Critical Events']}\n\n")

print(f"Statistics for .evtx filesm written to {S0me_stats}")
```

Listing 1: Code to analyze statistics of Windows log files.

Task 7.6.2

Results

Log File	List of Applications	# of Events for Each Application
auth.log	CRON, dbus-daemon, sshd, ...(Truncated)	3138, 15, 371, ...(Truncated)
kern.log	kernel	45663
syslog	CRON, ModemManager, NetworkManager, ...(Truncated)	1569, 30, 322, ... (Truncated)

Table 2: Analysis of Ubuntu Log Files

Please note the results are captured by the `'UbuntuLoggingResults.txt'` file. This file will be uploaded alongn tside this PDF.

Script

```
import re
import os
from collections import defaultdict
from tabulate import tabulate

# Determine the directory of the current script
Wheres_the_script = os.path.dirname(__file__)

# Log files to analyze
log_files = ['auth.log', 'kern.log', 'syslog']

```

```
# Regex pattern to capture the third "column" which may contain application names
Reg_for_apps = re.compile(r'\w{3}\s+\d{1,2}\s+\d{2}:\d{2}:\d{2}\s+\S+\s+([\w-]+)(?:\[\d+\])?:')
# Function to analyze a single log file
def analyze_these_logs(file_loc):
    how_many_events = defaultdict(int)
    with open(file_loc, 'r') as file:
        for line in file:
            match = Reg_for_apps.match(line)
            if match:
                app_name = match.group(1)
                app_name = app_name.rstrip(':')
                how_many_events[app_name] += 1
    return how_many_events

# Analyze each log file and collect results
results = []
for log_file in log_files:
    file_loc = os.path.join(Wheres_the_script, log_file)
    how_many_events = analyze_these_logs(file_loc)
    applications = ', '.join(sorted(how_many_events.keys()))
    events = ', '.join(str(how_many_events[app]) for app in sorted(how_many_events.keys()))
    results.append([log_file, applications, events])

# Save the results to a .txt file
output_file_loc = os.path.join(Wheres_the_script, 'UbuntuLoggingResults.txt')
with open(output_file_loc, 'w') as output_file:
  for result in results:
    log_file = result[0]
    applications = result[1]
    events = result[2]
    output_file.write(f"{log_file}:\n")
    output_file.write(f"-"*20+'\n')
    output_file.write(f"Number of events per app:\n")
    for app, event_count in zip(applications.split(', '), events.split(', ')):
      output_file.write(f"{app}: {event_count}\n")
    output_file.write('\n')

print(f"Results saved to {output_file_loc}")
```

Listing 2: Code to analyze statistics of Ubuntu log files.

Task 7.6.3

Results

List of IP Addresses	# of Events in Each Log File Respectively	Combined # of Events
52.167.144.55	1, -, -	5114
40.77.167.15	1, -, -	5114
134.126.120.54	5076, 7411, 7933	5114, 7504, 7942
...	...	...
146.190.129.170	24, -, 7	5114, -, 7942
...	...	...

Table 3: Statistical Analysis of Apache Log Files

Please note the results are captured by the result files for the Apache analysis. This file will be uploaded alogn tside this PDF.

Script

```
import os
import re
from collections import defaultdict

# Author: Abraham Reines
# Date: April 22, 2024

# Determine directory of current script
Wheres_that_script = os.path.dirname(__file__)
# Compute directory path for logs
Dir_loc = os.path.join(Wheres_that_script, '')

# Define a function to parse log file
def analyze_s0me_logs(Wheres_those_logs):
  """
  Parses Apache log file; compiles statistics for IP addresses and number of events.

  Args:
    Wheres_those_logs (str): path to log file.

  Returns:
    dict: dictionary with IP addresses as keys and a list of event counts as values.
  """
  How_many_IPs = defaultdict(lambda: [0, 0])  # Dictionary to store IP
  total_events = 0

  # Regex match IP addresses
  RegexIPs = re.compile(r'^(\d+\.\d+\.\d+\.\d+)')

  try:
    with open(Wheres_those_logs, 'r', encoding='ISO-8859-1') as file:
      for line in file:
        Found_match = RegexIPs.match(line)
        if Found_match:
          LocalizeIPs = Found_match.group(1)
          How_many_IPs[LocalizeIPs][0] += 1
          total_events += 1

    # Update total event counts
    for ip in How_many_IPs:
      How_many_IPs[ip][1] = total_events

    return How_many_IPs
  except FileNotFoundError:
    print(f"log file {Wheres_those_logs} was not found.")
    return {}

def Show_results(How_many_IPs, Wheres_those_logs):
  """
  Displays results of log analysis

  Args:
    How_many_IPs (dict): containing IP event counts.
    Wheres_those_logs (str): path to log file.
  """
  result_file_path = Wheres_those_logs.replace('.7', '_results.txt')
  with open(result_file_path, 'w') as file:
    file.write("{:<20} {:<40} {:<15}\n".format('List of IP Addresses', '# of events
    ↪ in each log file respectively', 'Combined # of events'))
    for ip, counts in How_many_IPs.items():
      file.write("{:<20} {:<40} {:<15}\n".format(ip, counts[0], counts[1]))
  print(f"Results saved to {result_file_path}")

# Path to log file
```

```
Files_with_logs = [file for file in os.listdir(Dir_loc) if file.endswith('.7')]
for log_file in Files_with_logs:
  Wheres_those_logs = os.path.join(Dir_loc, log_file)
  # Analyze log file and display results
  How_many_IPs = analyze_s0me_logs(Wheres_those_logs)
  Show_results(How_many_IPs, Wheres_those_logs)
  print("Done!")
```

Listing 3: Code to analyze statistics of Apache log files.

References

1. Real Python. (n.d.). *Speed Up Your Python Program With Concurrency.* Retrieved from https://realpython.com
2. Pascariu, C. (2022, September 13). *Log File Analysis with Python.* Pluralsight. Retrieved from https://www.pluralsight.
3. Toptal®. (n.d.). *Python Multithreading Tutorial: Concurrency and Parallelism.* Retrieved from https://www.toptal.com
4. HuangYiwei. (2019, December 30). *concurrent-log.* PyPI. Retrieved from https://pypi.org/project/concurrent-log/

Academic Integrity Pledge

"This work complies with the JMU honor code. I did not give or receive unauthorized help on this assignment."