Chapter 4: Log Data Analysis for Supporting Cloud Forensics

Recently, the number of cybercrimes take against cloud systems and services are in a rapid growth. Although, there are numerous procedures that are developed to protect cloud infrastructures and services from severe attacks. These procedures such as management systems, firewalls, and intrusion detection and prevention system, and antivirus, but still the risk of criminals’ activities is existing. This lead to attracting the attention of a lot of researchers around the world to the digital forensic field which is a science to aid law enforcement officers and digital investigators to identify, collect and analyze digital footprints or evidence which are collected from a crime scene. One of the significant sources of a digital evidence in the cloud is log data because they frequently connect events to a certain time. The process of log forensics mitigates the investigation process by identifying the malicious behavior and reveal the hidden malicious activities. Cloud log data analysis can help to reconstruct of cybercrime events which occurred in the cloud environment.

Traditional log data analysis procedures and tools can be adapted to cloud with using new analytics platforms such as Apache Hadoop and Apache Spark. Apache Hadoop has become one of the most popular solutions for handling a large amount of data. Apache Hadoop is an open source implementation of MapReduce programming model which operates on terabytes of data using commodity hardware. Apache Spark is a general-purpose cluster-computing engine, which is very fast and reliable. This chapter presents analysis approaches for log data using Apache Hadoop and Spark. The results show that Hadoop and Spark can use as a fast platform for handling the diverse large size of log data and extract useful information that can assist the digital investigators in analysis immense amount of generated cloud log data in a given frame time. Furthermore, the results can provision to reconstruct and generate a timeline related to historical past sequence events occurred during a crime as well as identify the malicious user’s IP address, date and time, with a number of access.
The rest of this chapter is structured as follows: section 4.1 provides chapter introduction while a brief overview about digital forensics, Apache Hadoop framework, Apache Spark, and log data forensics are presented in section 4.2. Section 4.3 provides previous and related work in the area of log data analysis while the proposed approaches using Apache Hadoop and Apache Spark are presented in section 4.4. Finally, the chapter summary of this innovative research area is presented in section 4.5.

4.1 Introduction
In the recent times, cloud computing has become one of the most popular processing paradigms. Cloud computing is an emerging revolutionary technology that has started changing the ways where people live and work. Cloud computing has five main characteristics that are identified from NIST’s definition [6] for the cloud computing as follows; On-demand self-service, Broad network access, Resource pooling, Rapid elasticity and Measured service. Cloud computing has two types of models as follows; deployment and Service Models. The deployment models are a Private cloud, Public cloud, Community cloud and Hybrid cloud while the service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

There are several challenges such as security which make cloud cannot be used to store data for many sectors such as business, healthcare, and banks which require an audit and regulatory compliance. Today, the process of investigating cloud-based crimes, extracting a digital evidence and reconstructing crime events is become a difficult task for digital investigators and examiners due to the large size of generated data from the cloud is growing in a credible way.

Criminal actions need the investigation but as far as the digital forensic methods are concert it is capable of recognizing the criminal user. In the cloud is log files are an imperative digital evidence for the reason that they frequently connect events to a certain time. The cloud user may be normal user as well as an attacker, so using log data can assist to determine vital information of the user identity such as IP address and Domain Network System (DNS) name. Using this information can be as a primary step to detect the suspicious user. Log forensics analysis mitigates the investigation process by identifying the malicious behavior and reveal the hidden criminals actions. There are various complex challenges
related to logging data analysis such as unification of log formats, the large size of log files, timestamp synchronization, timeline analysis of logs, the absence of critical information in logs, accessibility of logs, volatility of logs, decentralization and multiple tiers and layers.

Current procedures used for forensic analysis of digital evidence involve tools such as AccessData FTK and Encase. They are both missing in capabilities necessary to perform effective log analysis. There is a serious need to implement a forensic tool that is able to provide digital investigators the capability to achieve analysis whose results are efficient and effective. With regard to cloud forensics, significant events recorded in the access log may be extracted that aid in the development of an incident timeline relevant to an investigation. Issues specific to the analysis of log data involve the large size of target log data can be difficult for applications to handle efficiently, generated log data may or may not provide a means to log significant events, and as with most other log data categories events that are recorded, those recorded by different cloud devices such as firewalls, routers and etc. and related services are enigmatic in nature.

One of the significant sources in the cloud as a digital evidence is log data that can help to reconstruct timeline to know how, where and what happened during the crime. Petabytes of log files are generating from cloud data centers every day. It is complex challenging to store and analyze these huge volumes of log files. The problem of analyzing log files not only the size but also because of the incongruent structure of them. Traditional database solutions become not appropriate for analyzing such log files due to they are not capable of handling such a massive volume of logs professionally. Log analysis can be done by various procedures but what matters is response time.

Here, the analysis process is done using Apache Hadoop and Apache Spark as platforms. Hadoop contains MapReduce and Hadoop Distributed File System (HDFS). The MapReduce is used for processing and analyzing data in efficient and effective manner while the HDFS is used to store data before and after processing. Hadoop MapReduce framework provides parallel distributed processing and reliable data storage for large volumes of log files. Hadoop has a characteristic of moving computation to the data rather moving data to computation helps to improve response time. Spark is a general-purpose cluster-computing engine, which is very fast and reliable. This chapter aims to provide an implementation of batch and real-time log forensics approach that forensically analyzes large volume of log
files which identifying the malicious behaviour of attackers which can reduce forensics investigation time and costs as well as extract useful information then visualize the output data to help digital investigators to present evidential data in an effective manner in a court of law.

4.2 Background
This section presents a brief description about Digital Forensics, Apache Hadoop, Apache Spark and logs data forensics as follows:

4.2.1 Digital Forensics
With the vast increase in the volume of log data that generating every day in cloud computing environment, there is a grave need for cloud log analysis and monitoring. In the cloud, the size of log data has risen in an admirable way due to enormous cloud resources such as storage and processing with the intention of there is a need to new strategies to handle and analysis terabytes of cloud log data through performing share computation across a cluster of machines. There are various solutions that are developed to protect cloud systems from intense attacks. This lead to attracting the attention of researchers around the world of digital forensics. Digital forensics is a science to aid law enforcement officers and digital investigator to identify, collect and analyze digital footprint or evidence which are collected from a crime scene.

There are four main steps for performing investigation process. These steps comprise identification, preservation, analysis, and presentation is shown in Figure 4.1 as follows [54]:

- **Identification**: It is the process of identification of an incident and digital evidence, which will be required to prove the committed crime.
- **Preservation**: In the preservation process, digital investigator preserves the collected digital evidence from crime scenes such as hard disks, laptops, mobile phones, and any related pieces of evidence.
- **Analysis**: In the analysis phase, digital investigator interprets and correlates the evidential data to come to a conclusion, which can prove or disprove civil, or criminal activities.
• **Presentation:** In this process, the digital investigator makes a forensic report to summarize his findings of the criminal case. This report should be suitable to present to the court of law.

![Diagram](Image)

**Figure 4.1:** Digital Forensic Investigation Process.

### 4.2.2 Hadoop Framework

Apache Hadoop is an open source implementation of MapReduce programming model which operates on terabytes of data using commodity hardware. Hadoop is a framework for managing and handling large-scale computation and processing on a cluster of commodity hardware connected through a network. It allows applications to work with thousands of computational independent machines and petabytes of data. The prime precept of Hadoop is moving computations on the data rather the moving data for computation. Apache Hadoop is used to break down the large volume of input data into reduced chunks and each can be processed separately on various machines inside a cluster. MapReduce programming model is used inside the Hadoop framework to achieve parallel execution.

The MapReduce programming model is a simple programming model for parallel processing of a large amount of data. The main notion of MapReduce is to transform lists of input data to lists of output data. It occurs several times that input data is not in a readable format; it could be the tough task to understand large size datasets. In this case, there is a serious need to a model that can form input data lists into readable, understandable output lists. MapReduce does this conversion twice for the two major tasks: Map and Reduce just by dividing the entire workload into a number of tasks and distributing them over different computers in the Hadoop cluster. Log file consists of thousands of records i.e. logs which are in the text format. Currently, cloud servers are generating a lot of log data of the size of terabytes per a day. From the business perspective, there is a need to process these log data
so that can have appropriate reports of how business is going. For this application, MapReduce implementation in Apache Hadoop is one of the best solutions.

The MapReduce consists of three main functions which are Map, shuffle and reduce as shown in Figure 4.2 as follows:

1. **Map Function**: Map function does map stage in the MapReduce job. The map function produces intermediate output in the formula of (key, value) pairs. For every map task, a fresh case of the mapper is represented in a separate process. This function receives key, value, outputCollector, and reporter. Collect scheme from the OutputCollector object onwards intermediate (Key, value) pairs to reducer as an input for Reduce stage.

2. **Shuffle Function**: Just first Map function has completed, workers begin swapping intermediate output from map tasks to reduce tasks. This method of transfer intermediate output from the map tasks to reduce function as an input is named as shuffling. This is the single communication stage in MapReduce. Before pushing these (key, value) pairs as an input to Reducers; Hadoop groups all the values for the similar key. Such divided data is then allocated to reduce function.

3. **Reduce Function**: Reducer instance calls Reduce function for each key in the partition assigned to a Reducer. It accepts a key and iterator over all the values related to that key. Reduce function also has parameters such as iterator for all values, key, outputCollector, and reporter which works in a like way as for map function.

![MapReduce Logical Data Flow](image)

**Figure 4.2**: MapReduce Logical Data Flow.

### 4.2.3 Apache Spark

Apache Spark is an in-memory computing platform that offers an Application Programming Interface (API) for distributed programming to provide to be fast for interactive queries and iterative algorithms in compare with MapReduce programming model but is designed to be fast for interactive queries and iterative algorithms. Spark provides application programming interfaces in several programming languages such as Java, Python, and Scala. Spark
components are designed to operate close to the core and can be used as libraries through program development. Spark ecosystem contains different elements as shown in Figure 4.3 as follows [62]:

- **Spark Core**: It is the general execution engine for the Spark platform and all other functionality is built on top of it.
- **Spark SQL**: It is a Spark module for structured data processing. It can work as a distributed SQL query engine.
- **Spark Streaming**: It provides analytical and interactive applications across both streaming and historical data.
  - **MLlib**: It is a scalable Machine Learning (ML) library.
  - **GraphX**: It is a graph computation engine that allows users to use and achieve parallel processing in graphs.

![Figure 4.3: Apache Spark Ecosystem.](image)

In Spark, there is two type of analysis which are batch and stream processing as follows [63]:

- **Batch Processing**: Batch processing using Spark is can perform through loading data from documents all at once into one RDD, handle that RDD, the job finishes, and the program terminate. In a production system, you could set up a cron job to kick off a batch job each night to process the last days’ worth of log files and then publish statistics for the last day.

- **Stream Processing**: Steaming processing using Spark Streaming is an extension of the core Spark API that allows scalable, high-throughput and fault-tolerant stream processing of real-time data streams. Data can be ingested from various sources such as Kinesis, Flume, Kafka, or TCP sockets, and can be managed using intricate algorithms represented by high-level functions such as map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and online dashboards. It
operates as follows, the Spark Streaming receives real-time input data streams and splits the data into batches, which are then handled by the Spark engine to produce the final stream of outcomes in batches. Spark Streaming offers a high-level abstraction named Discretized Stream (DStream), which represents a continuous stream of data. DStreams can be formed either from input data streams from several sources such as Kinesis, Flume, Kafka and or by applying high-level operations on other DStreams. Internally, a DStream is represented as a sequence of RDDs as shown in Figure 4.4.

![Figure 4.4: Streaming Data Using Apache Spark.](image)

### 4.2.4 Log Data Forensic as Case Study

Log files are considered one of the imperative sources to aid system administrators and security engineers to determine the status of system and network during an execution time along with a specified timeline for troubleshooting of problems. They also used to capture the events happened within a computer system and networks [64]. Log files are a collection of log entries and each entry contains information related to a specific event that occurred in a system, virtual machine or network. Several log files within an association contain records associated with system security which is generated by operating systems on servers, workstations, networking devices in addition to this, there are various sources generated by software installed on a system such as an antivirus software, firewalls, Intrusion Detection, and Prevention System (IDS/IPS).

The log files are valuable for performing digital forensics analysis where they represent an important source of information for evidence gathering that records the events occurred
within the system, network, and cloud infrastructures. They also support to find digital evidence related to criminal activities and discover a weakness in the digital systems and network for security purpose. So, they can be said that logging is closely related to digital forensics. The log files also contain confidential information, they must be protected from illegal activities. The log files essential be admissible to be accepted in a court of law as proof of a committed crime by generating hash values to guarantee the integrity of them using cryptographic hashing algorithms such as Message-Digest 5 (MD5) and Secure Hash Algorithm 1 (SHA1). The digital Forensics process of the log file can be as shown in Figure 4.5.

![Digital Forensics Process Flow of Log File](image)

**Figure 4.5**: Digital Forensics Process Flow of Log File.

Log data represent one of the significant sources to support system digital investigators to define the status of system and network during a given time during occurring of a crime as well as help system administrators to create a specified timeline for troubleshooting of problems. They also used to seize the events occurred within a system [65]. Log data is a group of log entries which contain vital information related to a specific event that occurred in a system.
4.2.4.1 Types of Log Data

There are various types of logs as follows [66]:

- **Webserver Log:** Webserver log records all events occur on the web server such as IP address, date & time of access and request method.
- **System Log:** System logs are produced by an operating system which is pre-defined and holds information about system events, operation, drivers, and device change.
- **Setup Log:** Setup logs represents events happen during execution the installation of an application.
- **Security Log:** Security logs comprise security-related information to fix malicious behavior found in the system or network such as malware detection, file quarantines and time of malicious detection.
- **Application Log:** Application logs that are logged by an application. Application developers are in charge to specify what, when, and how to log through an application executing on a system.
- **Virtual Machine Log:** A file that contains records of each event performed on a virtual machine.
- **Audit Log:** Audit log holds user unauthorized access to the system and network for inspecting its responsibilities. It embraces destination addresses, user login information, and timestamp.
- **Network Log:** Network log is a log file that covers network related events like a description of the event, priority and time occurrence.

The log data are a vital source for forensics analysis where they capture events occurred within the system such as a computer, network and cloud infrastructures. This information support digital investigators to identify criminal activities as well as discover vulnerabilities.

4.2.4.2 Cloud Log Forensics

Cloud Log Forensics (CLF) is the incorporation of digital forensics with cloud logs arise a new research arena. It is very significant to produce and generate cloud logs for each and every event occurring in cloud computing in order to record all malicious behavior. However, cloud logs are generated at different locations resulting in a large number of cloud log files
which require proper cloud log management. Cloud log management is crucial to guarantee that cloud logs are stored on secure resources with adequate information for specific periods of time. Cloud logs benefit forensic investigators in the identification of policy violations, deceitful events, security incidents, and operational problems. Cloud logs also assist in carrying out audit analysis, recognizing long-term problems, and others benefits.

4.2.4.3 Cloud Log Analysis

Log analysis is the process to perform analysis on cloud log files collected from cloud log storage. Cloud log analysis identifies attackers through analyzing the cloud log files. The attackers want to keep themselves hidden from being investigated, which forces them to attack the log analysis resource/application to remove evidence of their attack. However, in large cloud computing infrastructure, finding the exact location where cloud log analysis is performed is a difficult task, which forces attackers to put more efforts in finding an exact location to attack. Decentralized CLF helps the investigator to perform analysis in multiple locations and prevents attackers from exploiting cloud log files at the time of analysis. Confidentiality and integrity are not exploited by attackers during their attacks while the availability of the cloud log files is affected based on their deletion. To analyze different cloud logs collected from various sources in cloud computing is not an easy task. The distributed infrastructure, virtualized environment, multi-tenant resources, huge running applications, millions of cloud users, real-time response, and a lot of other factors make cloud data forensics very challenging.

4.2.4.4 Cloud Log Forensics Challenges

Forensic analysis of cloud log data faced various challenges such as cloud log data as a big data, decentralized cloud logs, Accessibility of cloud logs, cloud log security, standardized cloud log format and fairness of cloud log analysis. Table 4.1 illustrate these challenges and proposed solutions. Here, we focused on cloud log data as a big data.
Table 4.1: Cloud Log Forensics Challenges and Proposed Solutions [66].

<table>
<thead>
<tr>
<th>Cloud Log Forensics Challenges</th>
<th>Proposed solutions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cloud log data as a big data</strong></td>
<td>Data filtering mechanism</td>
<td>To record important data in the cloud log data file.</td>
</tr>
<tr>
<td><strong>Accessibility of cloud logs</strong></td>
<td>Dependence on cloud service providers</td>
<td>The CSP has to deliver cloud logs to different investigators due to their control on numerous cloud logs. However, the integrity of data has to be confirmed by digital investigators.</td>
</tr>
<tr>
<td><strong>Cloud log security</strong></td>
<td>Proper access methods, Encryption of cloud log files and cryptographic key, and Replication of cloud log files</td>
<td>Cloud logs must only be accessed by authorized individuals through various access approaches. Both the cloud logs data and encryption key is encrypted due to better and reliable cloud log security.</td>
</tr>
<tr>
<td><strong>Decentralized cloud logs</strong></td>
<td>Centralized log analysis</td>
<td>To control and handle whole distributed cloud log analysis servers</td>
</tr>
<tr>
<td><strong>Standardized cloud log format</strong></td>
<td>Single cloud log format</td>
<td>Every cloud log generated at multiple locations in the cloud computing must have a single cloud log format with filled entries according to the requirement.</td>
</tr>
<tr>
<td><strong>Fairness of cloud log analysis</strong></td>
<td>Automatic cloud log analysis tool</td>
<td>A tool used to analyze cloud logs automatically with minimum human interventions.</td>
</tr>
</tbody>
</table>

4.2.4.5 Cloud Log Data as Big Data

Generating enormous amounts of cloud log data from several sources causes a problem for digital investigators in analyzing this large data. The problem relates to the concept named big data, i.e. cloud log data volume, variety, and value. The volume specifies the enormous amount of cloud log data created at multiple locations in the cloud, which causes complications for digital investigators in real-time cloud environments. The analysis of immense amounts of cloud logs data to investigate malicious activities performed by criminals, which are more complex in cloud computing than in traditional log data computing, requires time. The value of cloud log data produces a significant impact on cloud log forensic in terms of providing valuable information regarding events. For instance, if cloud logs do not provide sufficient information concerning an event occurring previously to assist digital investigators in understanding the situation, they are useless. The value provided by cloud log data is that they have to ensure the amount of information captured during the logging process is appropriate to investigate or analyze the situation simply.
4.2.4.6 Logging as a Service

The rise of cloud computing and the increasing requirement for processing and storing capabilities for log analysis resulted in the integration of cloud computing and log analysis. It has arisen a new concept named as Logging as a Service (LaaS). In this service, the users can gather logs from several devices and software in their organizations or infrastructures and then send them to the cloud for storage and processing. The general Cloud Log Forensics is shown in Figure 4.6 where the log data collect and then send to cloud to storage and analyzing, finally a report will generate to summarize recorded events according to the objective of log analysis.

Figure 4.6: Generalized Cloud Log Forensics Diagram [66].

4.3 Related Work

The logging operation is well thought-out as a crucial tool for security purpose which aids digital investigators in identifying and preventing operational subjects, episodes, attacks, and criminal activities. Logging is mostly used in auditing and monitoring systems to collect data for investigating different malicious crimes. The logs assist digital investigators to recognize the messages sources generated from numerous devices at different time intervals. Recently, researchers worked on analyzing log files using Hadoop framework but few focus on using log analysis for forensic purpose.

Currently, several researchers studied log file analysis topic but few are focused on log file forensics to help digital investigators to analyze large log files from cloud systems.
Saravanan et al. [65], they introduced a method to analyze NASA web log file to calculate a number of hits received by a website in each hour using Apache Hadoop framework are calculated and it is shown that Hadoop framework takes less response time to produce precise results. Khan, Suleman, et al [66], introduced a survey and reviews about current work related to Cloud log forensics (CLF) and explores various challenges involved in the forensic investigation of cloud log data.

Meena et al. [67] introduced a review of digital forensic with fuzzy rules and operations in order to analyze the malicious activity and the attacker from log files. They mentioned that study it seems to be that the digital forensic is efficient works with the multi-relation classification using frequent pattern mining. Sayalee et al. [68], they proposed a log file analysis tool called HMR that offers the user graphical reports showing traffic sources, user’s activity, and hits for web pages, in which part of website users are interested. From these reports, business communities can assess which parts of the website need to be enhanced, which are the potential users, from which geographical region website is getting maximum hits, etc., which will assist in planning upcoming marketing strategies. Amor et al [69], provided a comparison of the main considered web application forensics tools as well as presented a methodology of a web application forensics investigation.

SayaJee et al. [70], they applied MapReduce programming model for analyzing web log files to get hit count of the specific web application. In [71], they proposed a log file forensic model which specifies the phases that forensic investigators can follow with regard to the extraction and examination of digital evidence from log files for use in lawful proceedings. In [72], they proposed a methodology for analysis of web log files. The experimental results can be used by forensic investigators for the investigative task. As well, the proposed timeline analysis can be utilized by the web designer in order to schedule the upgrade and enhancement of the website. In [62], They presented a performance evaluation study of cloud-based log files using the cloud computational frameworks Apache Hadoop and Apache Spark. They proposed accurate log file analysis applications in using both frameworks and executed SQL-type queries in real Apache Web Server log files.

### 4.4 Proposed Approaches

This section presents two approaches which are used for analyzing log data based on Hadoop and Spark respectively.
4.4.1 Hadoop-based Large Log Data Analysis for Cloud Forensics

This section introduces a Hadoop-based Large Log Data Analysis approach for Cloud Forensics.

4.4.1.1 Proposed Approach Block Diagram

The block diagram of proposed approach is shown in Figure 4.7.

![Digital Proposed Approach Block Diagram](image.png)

**Figure 4.7:** Digital Proposed Approach Block Diagram.

4.4.1.2 Description of Proposed Approach

The proposed approach steps as follows:

- Enter the Digital Evidence “Log Files”.
- Perform Pre-processing on Log Files.
- Perform processing using Hadoop framework.
- Use the output of Hadoop framework.
- Visualization /Mapping the output data to extract more knowledge from the log files.

4.4.1.3 Pseudo Code of MapReduce Programs for Log Data Analysis

General steps of MapReduce Programs for Log Data Analysis can be as shown in Figure 4.8.
Log analysis Program_1: Procedure of calculating total access of IP Address or URL per hour in log data

**Input:** Large number of Log Data in the form like “199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245”.

**Task:** Compute number access of IP Address or URL per hour across all log data.

**Output:** Total number of occurrence of IP Address or URL per hour.

**While** There are more log lines **do**

**For** all log lines in log file **do**

**Map (String key, String value):**

// **key:** log file name,
// **value:** file contents

Extract IP Address or URL Field from Log Data

Extract timestamp Field from Log Data

Extract hour from timestamp

For each IP Address or URL + hour y in value:

EmitIntermediate (Y, "1");

**Reduce (String key, Iterator values):**

// **key:** IP Address or URL + Hour
// **values:** a list of counts

Int ip+hour_count = 0;

For each it in values:

ip_hour_count += ParseInt (t);

Emit (key, AsString (ip_hour_count));
LogAnalysis Program_2: Procedure of counting number of access for each IP Address or URL in log data

**Input:** Large number of Log Data in the form like “199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245”.

**Task:** Compute IP Address or URL count across all log data.

**Output:** Total number of access.

**While** There are more log lines **do**

<table>
<thead>
<tr>
<th>For all log lines in log file <strong>do</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map (String key, String value):</strong></td>
</tr>
<tr>
<td>// key: log file name,</td>
</tr>
<tr>
<td>// value: file contents</td>
</tr>
<tr>
<td>Extract IP Address or URL Field from Log Data</td>
</tr>
<tr>
<td>For each IP Address or URL X in value:</td>
</tr>
<tr>
<td>EmitIntermediate (X, &quot;1&quot;);</td>
</tr>
</tbody>
</table>

| **Reduce (String key, Iterator values):** |
| // key: IP Address or URL          |
| // values: a list of counts        |
| int IP/URL_count = 0;              |
| For each v in values:              |
| IP/URL_count += ParseInt(v);      |
| Emit (key, AsString(IP/URL_count));|

LogAnalysis Program_3: Procedure of counting hits per hour in log data

**Input:** Large number of Log Data in the form like “199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245”.

**Task:** Compute the number of hits per hour across all log data.

**Output:** Total number of hits per hour.

**While** There are more log lines **do**

<table>
<thead>
<tr>
<th>For all log lines in log file <strong>do</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map (String key, String value):</strong></td>
</tr>
<tr>
<td>// key: log file name,</td>
</tr>
<tr>
<td>// value: file contents</td>
</tr>
<tr>
<td>Extract timestamp Field from Log Data</td>
</tr>
<tr>
<td>Extract hour from timestamp for each hour h in value:</td>
</tr>
<tr>
<td>EmitIntermediate (H, &quot;1&quot;);</td>
</tr>
</tbody>
</table>

| **Reduce (String key, Iterator values):** |
| // key: hour                        |
| // values: a list of counts         |
| int hour_count = 0;                 |
| For each t in values:              |
| hour_count += ParseInt (t);        |
| Emit (key, AsString (hour_count)); |
4.4.1.4 Experimental Study and Analysis

4.4.1.4.1 Experimental Environment
The proposed system is tested and evaluated on a system with a virtual machine with Ubuntu 14.04 operating system with 1 GB RAM, Processor Intel® Core™ i5-4460 CPU @ 3.20GHz and 30 GB Hard Disk. Hadoop framework, version 1.0.3 is installed in this system. Three different log files are used as input data. The size of log files, file_1, file_2, and file_3 are 194 MB, 391 MB, and 587 MB respectively. Figure 4.9 shows the weblog file format and its component which is used during the experiment. The proposed programs can simply be adapted to other log file formats. The description of log file format is as follows: Host field refers to the IP address or URL of the user who makes the request. Then, Date and time stamp of the HTTP request followed by HTTP request involving three elements (1) HTTP method=GET, (2) HTTP request resource= history/apollo, and (3) HTTP protocol version= 1.0. Then the request status which is a numeric code used to state the status of HTTP request which means success or failure, followed by the size or bytes which are a numeric value refers to a number of bytes of data transferred during the HTTP request.

![Log File Format](image)

**Figure 4.9:** Access Log File Format.

4.4.1.4.2 Results Analysis and Discussion
Currently, the number of attacks and incidents are increasing against cloud infrastructure which makes security in charge to produce diverse types of logs from every part in the cloud such as application, network devices, servers and virtual machines which communicate with users or systems need to record communication events. There is various type of log files involves application, system, network logs, webserver logs and others. This work focusing on web server logs as a case study but also simply can extend and modify the proposed programs to adapt to other kinds of logs. Three different programs called LogAnalysis_1, LogAnalysis_2, and LogAnalysis_3 are implemented and tested on the three input log files. The first program, LogAnalysis_1, is used to calculate a total number of hits in each hour for each IP address or URL as tabulated in Table 4.2, a sample of the program results. Table 4.3 shows execution time of the first program for the three log files.
Table 4.2: Sample Results of LogAnalysis_1 Program for File_1.

<table>
<thead>
<tr>
<th>IP Address or URL</th>
<th>Hour</th>
<th>No of Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.99.73.86 09</td>
<td>09</td>
<td>10</td>
</tr>
<tr>
<td>130.99.73.86 12</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>130.99.73.86 13</td>
<td>13</td>
<td>44</td>
</tr>
<tr>
<td>130.99.73.86 18</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>130.99.74.81 7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>130.99.77.58 09</td>
<td>09</td>
<td>15</td>
</tr>
<tr>
<td>130.99.79.117 11</td>
<td>11</td>
<td>69</td>
</tr>
</tbody>
</table>

Table 4.3: Execution Time of LogAnalysis_1 Program.

<table>
<thead>
<tr>
<th>Files</th>
<th>Map Time (ms)</th>
<th>Reduce Time (ms)</th>
<th>Total Time (ms)</th>
<th>No of Records</th>
<th>Launched Reduce Tasks</th>
<th>Launched Map Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>File_1</td>
<td>14,450</td>
<td>4,510</td>
<td>18,960</td>
<td>1,891,715</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>File_2</td>
<td>41,360</td>
<td>24,790</td>
<td>66,150</td>
<td>3,783,430</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>File_3</td>
<td>62,670</td>
<td>29,390</td>
<td>92,060</td>
<td>5,675,145</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

The second program, LogAnalysis_2, is used to calculate a total number of hits each IP address or URL as tabulated in Table 4.4, a sample of the program results. Table 4.5 shows execution time of the second program for the three log files.

Table 4.4: Sample results from LogAnalysis_2 Program for File_1.

<table>
<thead>
<tr>
<th>IP Address or URL</th>
<th>No of Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.125.35.69</td>
<td>1</td>
</tr>
<tr>
<td>128.126.104.25</td>
<td>5</td>
</tr>
<tr>
<td>128.126.114.110</td>
<td>12</td>
</tr>
<tr>
<td>128.126.122.5</td>
<td>3</td>
</tr>
<tr>
<td>128.126.152.205</td>
<td>4</td>
</tr>
<tr>
<td>128.126.153.106</td>
<td>5</td>
</tr>
<tr>
<td>128.126.153.108</td>
<td>2</td>
</tr>
</tbody>
</table>
The third program, LogAnalysis_3, is used to calculate a total number of hits in each hour during a day as shown in Figure 4.10. Table 4.5 shows execution time of the third program for the three log files.

Table 4.5: Execution Time of LogAnalysis_2 Program.

<table>
<thead>
<tr>
<th>Files</th>
<th>Map Time (ms)</th>
<th>Reduce Time (ms)</th>
<th>Total Time (ms)</th>
<th>No of Records</th>
<th>Launched Reduce Tasks</th>
<th>Launched Map Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>File_1</td>
<td>19,910</td>
<td>4,760</td>
<td>24,670</td>
<td>1,891,715</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>File_2</td>
<td>34,180</td>
<td>6,600</td>
<td>40,780</td>
<td>3,783,430</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>File_3</td>
<td>66,730</td>
<td>44,010</td>
<td>110,740</td>
<td>5,675,145</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.6: Execution Time of LogAnalysis_3 Program.

<table>
<thead>
<tr>
<th>Files</th>
<th>Map Time (ms)</th>
<th>Reduce Time (ms)</th>
<th>Total Time (ms)</th>
<th>No of Records</th>
<th>Launched Reduce Tasks</th>
<th>Launched Map Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>File_1</td>
<td>11,460</td>
<td>2,820</td>
<td>14,280</td>
<td>1,891,715</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>File_2</td>
<td>23,470</td>
<td>7,750</td>
<td>31,220</td>
<td>3,783,430</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>File_3</td>
<td>52,990</td>
<td>19,360</td>
<td>72,350</td>
<td>5,675,145</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.7 and Figure 4.11 shows a comparison of the execution time for the three programs with three log files. Analyze the dates and time based on the occurrence of crime events that reported by first responders and system administrators so that from results can
determine the IP Address or URL that accessed and visited the server in a certain time and then filter malicious IP address based on behavior during the crime occurrence. Using hypothesis that drawn by investigators to reconstruct timeline related to the incident. Finally, a report can be generated from the findings during analysis log files. The results show that the proposed approach can be useful in forensic analysis of the large size of logs files in a cloud environment using big data solutions such as Hadoop framework.

**Table 4.7:** Comparison of Execution Time of three programs.

<table>
<thead>
<tr>
<th>Program</th>
<th>File_1 (194 MB)</th>
<th>File_2 (Size 391 MB)</th>
<th>File_3(Size 587 MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogAnalysis_1</td>
<td>18,960 ms</td>
<td>66,150 ms</td>
<td>92,060 ms</td>
</tr>
<tr>
<td>LogAnalysis_2</td>
<td>24,670 ms</td>
<td>40,780 ms</td>
<td>1,10,740 ms</td>
</tr>
<tr>
<td>LogAnalysis_3</td>
<td>14,280 ms</td>
<td>31,220 ms</td>
<td>72,350 ms</td>
</tr>
</tbody>
</table>

![Figure 4.7: Comparison of Execution Time of three programs.](image)

**Figure 4.11:** Execution Time of Three Programs for Different Files.

From these results, the digital investigators can use timeline reconstruction to extract information about malicious users during the hacking period as shown in Figure 4.12. This information includes IP address and number of access during a specific hour per day.
4.4.2 Spark-based Large Log Data Analysis for Cloud Forensics

This section presents Spark-based Large Log Data Analysis approach for Cloud Forensics.

4.4.2.1 Proposed Approach Flowchart

The flowchart of proposed approach is shown in Figure 4.13.

---

**Figure 4.12**: Conceptual View of Timeline Reconstruction.

**Figure 4.13**: Proposed Spark-based Approach.
4.4.2.2 Pseudo Code of Spark Programs for Log Data Analysis

**LogAnalysis Batch Program 1:** Procedure of counting number of access for each IP Address or URL in log data

**Input:** Log data file  
**Task:** Count IP Address or URL count across all log data.  
**Output:** Total number of access.  
**While** There are more log lines do  
  **For** all log lines in log file do  
    Read log line  
    Parse log line  
    Extract *IP Address or URL* Field from each log line  
    Build a dataset of (String, Int) pairs called *IPAddress*  
    Print *IPAddress*  

**LogAnalysis Batch Program 2:** Procedure of counting hits per hour in log data

**Input:** Log data file.  
**Task:** Calculate the number of hits per hour across all log data.  
**Output:** Total number of hits per hour.  
**While** There are more log lines do  
  **For** all log lines in log file do  
    Read log line  
    Parse log line  
    Extract *hour* from *timestamp* Field from each log line  
    Build a dataset of (String, Int) pairs called *HrCount*  
    Print *HrCount*  

**LogAnalysis Batch Program 3:** Procedure of calculating total access of IP Address or URL per hour in log data

**Input:** Log data file  
**Task:** Compute number access of IP Address or URL per hour across all log data.  
**Output:** Total number of occurrence of IP Address or URL per hour.  
**While** There are more log lines do  
  **For** all log lines in log file do  
    Read log line  
    Parse log line  
    Extract *IP Address* Field and *hour* from *timestamp* Field from each Log line  
    Build a dataset of (String, Int) pairs called *IPPlusHr*  
    Print *IPPlusHr*
LogAnalysis Streaming Program: Procedure of counting number of access for each IP Address or URL in log data

Input: Log data file
Task: Compute IP Address or URL count, number of hits per hour, number occurrence of IP Address or URL per hour across all log data.
Output: Total number of access, the total number of hits per hour, the total number of occurrence of IP Address or URL per hour.

While There are more log lines do
  For all log lines in log file do
    Read log line
    Parse log line
    Extract IP Address or URL Field from each log line
    Build a dataset of (String, Int) pairs called IPAddress
    Print IPAddress
    Extract hour from timestamp Field from each log line
    Build a dataset of (String, Int) pairs called IPPlusHr
    Print IPPlusHr
    Extract IP Address Field and hour from timestamp Field from Log Data
    Build a dataset of (String, Int) pairs called HrCount
    Print HrCount

4.4.2.3 Experimental Study and Analysis

4.4.2.3.1 Experimental Environment
The proposed approach is tested and evaluated on a system with a virtual machine with Ubuntu 14.04 operating system with 1 GB RAM, Processor Intel® Core™ i5-4460 CPU @ 3.20GHz and 30 GB Hard Disk. Apache Spark framework, version 2.0.2 is installed in this system. Three log files, file_1, file_2, and file_3 are used as input batch data with different size, 194 MB, 391 MB, and 587 MB respectively. Figure 4.9 shows the webserver log file format and its elements which are used during the experiment. The proposed programs can easily be adapted to other log file formats. The description of log file format is as follows: host field refers to the IP address or URL of the user who makes the request. Followed by rfc931 and Username. Then, date and time stamp of the HTTP request followed by HTTP request involving three elements (1) HTTP method=GET, (2) HTTP request resource, and (3) HTTP protocol version. Then the request status which is a numeric code used to state the status of HTTP request which means success or failure, followed by the size or bytes which are a numeric value refers to a number of bytes of data transferred during the HTTP request.
4.4.2.3.2 Results Analysis and Discussion

In recent times, the severe crimes and incidents against cloud infrastructures and services are growing inconceivable manner which makes security and forensic investigation tasks more challenging. Various types of logs from cloud resources and services such as application, network devices, servers and virtual machines. This work focusing on web server logs as a case study but also easily can extend and modify the proposed programs to adapt to other kinds of logs. Two type of log analysis is performed using apache-spark which are batch and real-time processing.

- **Batch Processing**: For performing batch processing using apache-spark, three programs, program_1, program_1, and program_1, are applied and tested on the three input log files. Program_1 is used to calculate a total number of access in each hour for each IP address or URL while program_2 is used to calculate a total number of access for each hour per day and finally, program_3 is used to calculate a total number of access in each hour for each IP address or URL. Sample results from Programs_1, Programs_2 and Programs_3 are tabulated in Table 4.8, Table 4.9 and Table 4.10. Table 4.11 and Figure 4.14 shows execution time of the three programs with the three log files.

Table 4.8: Sample of Results from Program_1.

<table>
<thead>
<tr>
<th>IP Address/URL</th>
<th>Number of Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>calvert.psc.rochester.edu</td>
<td>12</td>
</tr>
<tr>
<td>u'204.28.21.52</td>
<td>2</td>
</tr>
<tr>
<td>u'sl137.active.ch</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 4.9: Sample of Results from Program_2.

<table>
<thead>
<tr>
<th>Hour Per Day</th>
<th>Number of Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>37398</td>
</tr>
<tr>
<td>10</td>
<td>105507</td>
</tr>
<tr>
<td>11</td>
<td>115720</td>
</tr>
</tbody>
</table>
Table 4.10: Sample of Results from Program_3.

<table>
<thead>
<tr>
<th>IP Address/URL</th>
<th>Hour of Access</th>
<th>Number of Access Per Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad07-008.compuserve.com</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>128.95.122.124</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>162.127.24.81</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.11: Apache Spark Batch Processing Programs.

<table>
<thead>
<tr>
<th>Files</th>
<th>Batch Processing Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Program_1</td>
</tr>
<tr>
<td>File_1</td>
<td>41.729718 s</td>
</tr>
<tr>
<td>File_2</td>
<td>94.644172 s</td>
</tr>
<tr>
<td>File_3</td>
<td>142.013627 s</td>
</tr>
</tbody>
</table>

Figure 4.14: Execution Time for Three Batch Processing Programs.

- **Stream Processing**: Real-Time or stream processing is important in comparing with batch operations with extremely high latencies, and expensive, especially in security fields such as fraud and attack detection. In this work, a continuous stream of log entries (e.g. a web server log) to Spark Streaming to process stream of events and real-time computation over it for counting the number of accesses from different IP addresses, number of access on specific hour per day over a 1 second, window of time and then generate for example an alert when some condition is met such as the number of accesses is greater than 50.
In our streaming program, simply any lines typed in the terminal running the netcat server will be calculated and published on the screen every second. Figure 4.15 shows log lines which will be processed by the stream spark program. Screenshot of output in the terminal is shown in Figure 4.16. The output of the program to count a number of access of each IP address, a number of access of each hour per day and finally the number of access of IP address access per each hour per day. This program easily to modify to perform more operations on log entries.

Figure 4.15: Log Lines as Input for Streaming Program.

Figure 4.16: Output of Streaming Program.

As log data can use to study vulnerabilities and weaknesses in systems such as networks to fix them. Similarly, it can help to accomplish timeline reconstruction for
investigation purpose that helps to draw a conclusion about what happened during a crime and assist to identify attackers who hacked the systems. Based on aforementioned log data analysis programs results. The investigators use these results to construct a traceback for past sequence of events and analyze criminal actions in detail. The proposed approach can use to obtain past sequence of events in cloud computing environment during a specific interval of time then visualized using existing tools.

4.4.2.3.3 Automated Timeline Approach for Cloud Forensic

Design and develop an automated timeline approach for analyzing of the cloud infrastructures and services is very important for cloud forensic analysis. The proposed approach can assist cloud administrator and security engineers as well as cloud investigators to improve the security of cloud infrastructures by determining occurred connections that accessed the cloud infrastructure. Firstly, we have obtained access log files, which are recorded in cloud servers, then the obtained log files were analyzed by the proposed approach.

Develop an automatic timeline application for analyzing and monitoring of cloud log data for a forensic analysis purpose is an essential task for researchers in cloud forensics area. In the proposed approach, can be next built on top of cloud infrastructures to perform automated log analysis for each part of the cloud system. Building such type of application within the cloud can reduce time and cost of performing cloud forensic process through determining occurred events in the cloud.

A Case Study: Steal sensitive data from website hosted in cloud servers

Cloud administrator reported some users of that one website (called www.yyy.com) trying to access sensitive data or confidential documents that are not authorized for them. Then, website administrator reported that some documents are already stolen.

In this case, security experts and forensic investigators will use generated log data from the web server to help them to identify the criminals who stole the documents. Primary steps to perform timeline reconstruction related incidents in the cloud environment from available log data can be as follows:

1. Process available access log data, which are recorded in cloud storage.
2. Perform pre-processing and cleaning operations on log data.
3. Detect and determine specific attributes such as IP address and, date and times for timeline analysis purpose that helps to collect the evidence of criminal actions.
4. Apply specific statistics operations on these attributes such as count number of access to cloud server by the specific user per specific date and time during the criminal actions.

5. Visualize output of the previous step to conclude what has been happened during the specific period of time.

6. Generate a forensic report as the evidence that summary timeline events that support the trial to persuade the court against the suspect user.

4.5 Chapter Summary

Using specific security techniques and methods may help to prevent or detect attacks on cloud systems, nevertheless cannot detection out the attacker who has performed this attack. Being unable to trace back an attack, encourages criminals to launch new severe attacks against cloud infrastructure and systems. Cloud forensics aims to trace back and attribute cloud attacks to its creator. This may knowingly decrease cloud attacks and hence enhance its security. One of the significant tasks can be done by cloud investigator and incident responders are the reconstruction of an event timeline happening the crime under investigation. The aim of this chapter to use fast computing platform to analysis large size of batch and real-time log data to reconstruct crime events with accomplishment an accurate timeline that assists the investigators to detect criminals in petite time.

In this chapter, two forensics analysis approaches for log data using Hadoop framework and Apache Spark are introduced. The first approach, Apache Hadoop for analysis of log data is used while in the second approach an Apache Spark is used to provide batch and real-time analysis of web server log data. In each approach, three different programs are implemented and tested on three different log files in size. Each program extracts the different type of information that can help digital investigators in reconstructing timeline related crimes that are occurred. The results show that Hadoop and Spark can be used as fast platforms for processing various large size of log files and extract beneficial information that can support digital investigators in analysis massive amount of cloud log data in a given frame time as well as reconstructed timeline related to incidents. Furthermore, the results can provision to reconstruct and generate a timeline related to historical past sequence events occurred during a crime as well as identify the malicious user’s IP address, date and time, with a number of access.