Addis Ababa University

College of Natural and Computational Science
School of Information Science

Streaming Big-data Analytic Platform for Unified Log Management and Monitoring

---

By

Muluken Sholaye

Advisor

Dereje Teferi(Phd)

Sep, 2021
Addis Ababa
Ethiopia

Addis Ababa University

College of Natural and Computational Science

School of Information Science

(Information Science Stream)

Streaming Big-data Analytic Platform for Unified Log Management and Monitoring

A Thesis Submitted to the College of Natural and Computational Sciences of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science and Systems

By
Muluken Sholaye

Sep, 2021

Addis Ababa University

College of Natural and Computational Science

School of Information Science

(Information Science Stream)

Streaming Big-data Analytic Platform for Unified Log Management and Monitoring

By
Muluken Sholaye

Advisor
Dereje Teferi (PhD)

Name and Signature of Members of the Examining Board

Name	Title	Signature	Date
Dereje Teferi(PhD)	Advisor	_____	_____
Wondwosen Mulugeta(Phd)	Examiner	_____	_____
Michael Melese(Phd)	Examiner	_____	_____

Sep, 2021

Declaration

This thesis has not previously been accepted for any degree and is not being concurrently submitted in candidature for any degree in any university.

I declare that this thesis entitled “Streaming Big-data Analytic platform for Unified Log Management and Monitoring” is a result of my own investigation, except where otherwise stated. I have undertaken the study independently with the guidance and support of my research advisor. Other sources are acknowledged by citations giving explicit references. A list of references is appended.

Signature: _____

Muluken Sholaye Tesfaye

The thesis has been submitted for examination with my approval as a university advisor.

Advisor’s Signature: _____

Dereje Teferi(Ph.D.)

Acknowledgements

Before all I praise Almighty GOD and his beloved mother for all the blessings I had. It would have been almost impossible for me to complete this thesis if many people next to GOD weren't willing to give me their helping hands.

I would like to express my sincere gratitude to my advisor Dereje Teferi(Phd) for the continuous support and guidance and for his patience under those difficult circumstances. Without his mentor-ship i wouldn't have gotten far doing this thesis work.

Next I would like to thank my dearest family My Dad, My Mam and My Three brothers for all the help and support you gave me in my entire life.

Then I would like to thank all my friends and classmates for all the beautiful memories that I will take to the end of my times.

Table of Contents

.

List of Figures

List of Tables

List of Abbreviations

DSR:	Design Science Research
ELK:	ElasticSearch Logstash Kibana
MLLiB:	Machine Learning Library
YARN:	Yet another Resource Negotiator
HDFS:	Hadoop Distributed FileSystem
LMS:	Log management Software
RDD:	Resilient Distributed Datasets
API:	Application Programming Interface
DPI:	Deep Packet Inspection
UDP:	User Datagram Protocol
FTP:	File Transfer Protocol
SIEM:	Security Information and Event Management
XML:	eXtended Markup Language
JSON:	Java Script Object Notation
SOC:	Security Operation Centers
HPC:	High Performance Computing

Abstract

Over the past 20 years data has increased in a large scale and in various fields. Managing the produced data and gaining insights from it is often a challenge and possibly a key to competitive advantage, consequently forcing industries to find ways to collect and integrate massive data from widely distributed data sources. Nowadays organizations are overwhelmed by external data sources on the top of their internal data source. The challenges that organizations are facing is multifaceted such as the availability of big storage space (volume), the speed at which data creations takes place (velocity) and diversification of data types (variety).

It is mandatory for an organization to always know the overall posture of their IS systems and overall IT infrastructure at any given time. One way to always know the state of an environment is through system logs. Although logs are abundant and very rich source of information, they are mostly overlooked due to complexity and volume of the contents of the log.

The main objective of this research work is to design and demonstrate a generic service oriented big data architectural solution framework for efficient and fault tolerant log management and near-real-time monitoring of selected standard log types of syslog, metric log, net flow and audit log files. An integrated and layered stack of Beats data shippers, Apache Kafka, Apache Spark, Elastic search indexer and Kibana Visualization Toolkit are the main technologies used in the solution framework and Design Science Research (DSR) Methodology is applied.

Openstack Mirantis based Cloud system of 4 interconnected nodes is prepared to deploy the proposed Log management system. The above mentioned log files contents will be redirected to the Streaming log analytics engine in real-time. Performance, Scalability and Fault Tolerance are the main evaluation metrics used along with assessment against results obtained by similar research works.

The result of the research work shows that the proposed layered architecture of streaming technology stack has proven to visualize in average of less than 1 seconds of time. Data replication and buffering functionality provided by Apache Kafka has added fault tolerance for the system and scalability is as fast as installing client services of the technologies used in the prototype system.

Keywords: Streaming Big data, Log management system, Big data Analytics, Real-time monitoring, Log Stream processing

CHAPTER ONE

INTRODUCTION

1.1 Background

Large amounts of data have been generated at an alarming rate since the invention of personal computers. This phenomenon serves as the driving force for current and future research horizons. Mobile devices, digital sensors, communications, computer, and storage advancements have made it possible to collect data. The entire amount of data in the world has expanded nine times in five years, according to the renowned IT firm Industrial Development Corporation. It is expected that this amount will quadruple every two years at the absolute least [1].

Data has increased on a vast scale in a variety of industries over the last 20 years. Managing and obtaining insights from created data is a problem, and it may be a key to competitive advantage, thus industries must create ways to collect and integrate huge data from widely dispersed sources. Cloud computing and the Internet of Things (IoT) are accelerating the accumulation of data even faster [2].Large amounts of data have been generated at an alarming rate since the invention of personal computers. This phenomenon serves as the driving force for current and future research horizons. Mobile devices, digital sensors, communications, computer, and storage advancements have made it possible to collect data. The entire amount of data in the world has expanded nine times in five years, according to the renowned IT firm Industrial Development Corporation. It is expected that this amount will quadruple every two years at the absolute least [1] [3].

The economic industry in the twenty-first century, dubbed the "information era," is heavily reliant on data. How much data must be processed in order for it to be useful? Researchers found that barely 0.5 percent of all data created globally is evaluated. In a world where “every two days we create as much information as we did from the beginning of time until 2003,” there is a need to connect data examined with current trends to improved business models, with systematic processing and analysis of big data serving as the underlying element [3].

Big data is a collection of big data sets that are difficult to analyze in a timely manner using traditional processing approaches like Relational Database Management Systems (RDBMS). The process of extracting information, cleansing data, data integration, and data representation, as well as the physical storage that holds Big Data and decreasing redundancy, are all problems in Big Data. Big data is a combination of data sets which size, the rate of growth and complexity make them

very difficult to be captured, managed, processed or analyzed by traditional technologies and tools such as relational databases and desktop statistics or visualization packages [4] [5].

The principle of big data has been endemic within information science and the world of digital communications since the beginning days of computing. Because data is created by everyone and for everything, such as mobile devices, call centers, web servers, and social networking sites, big data is rising by the day. However, the problem is that it is too large, too quick, and too difficult to handle for existing traditional database systems and technology. Many enterprises, such as contact centers, sensors, online logs, and digital photographs, collect huge amounts of data created by high-volume interactions. Meeting big data challenges while continuously enhancing operational efficiency is critical to their business's success. [6].

Big Data strives to uncover hidden patterns, and it has ushered in a shift from a model-driven to a data-driven research paradigm. It is based on the interaction of:(1) Technology: optimizing compute power and algorithmic precision to collect, analyze, link, and compare enormous data sets. (2) Analysis: identifying patterns in large data sets that can be used to establish economic, social, technical, and legal claims. (3) Mythology: the general idea that the greater the number of data sets we have, the higher the level of intellect and knowledge that may yield previously unattainable discoveries, all while maintaining the aura of truth, objectivity, and accuracy" [7].

Scientists usually mention that Big Data has four-dimensions. These are Volume, Velocity, Variety, and Veracity. It is believed by many that big Data consists of "high-volume, velocity and variety information assets that require innovative and cost-effective forms of information processing for enhanced insights and decision making" [8].

Big data computing is divided into two categories based on the processing requirements: large data batch computing and big data stream computing. When it comes to assessing real-time application scenarios, batch processing of big data is insufficient. Data that is generated in real time almost always necessitates real-time data analysis. The output must also have a minimal latency and be created in a matter of seconds. This needs big data stream analyses, which have grown increasingly common in recent years as a result of a variety of applications generating large amounts of data at a high rate. This made it very difficult for existing data mining tools, technologies, methods, and techniques to be applied directly on big data streams due to the dynamic characteristics of big data [9].

A log is a written record of what happens in an organization's systems and networks. Logs are made up of log entries, each of which contains details about a specific event that occurred in a system or network. Computer security records can be found in several logs inside a company. Security

software, such as antivirus software, firewalls, and intrusion detection and prevention systems; operating systems on servers, workstations, and networking equipment; and apps are all sources of computer security logs. [1] [3].

Log data, which contains information about the kind of events/attacks that occur within an organization's network, is extremely useful in the shifting circumstances. Log data can also be used to trace an intruder's behavior in the course of their day-to-day job and to provide evidence in the case of malicious behavior. Most network and system devices, including as operating systems, routers, switches, firewalls, wireless access points, VPN servers, antivirus (AV) systems, and cloud service platforms, generate log data for events on a regular basis. From performance data to problem detection to intrusion detection, logs can tell you a lot about what's going on with your system or network. After an incident, logs can be a helpful source of "forensic" information for identifying "what happened." They can also be used to answer a variety of other questions, such as “how well are we doing?”, “will something go wrong in the near future?”, and “did we do enough to protect anything?” as well as a bunch of audit-related queries like, "Who did what?" [10] [11] [12].

Streaming analytics refers to the analysis of massive amounts of data acquired at high speeds from a variety of sources in real time. It's a new paradigm necessitated by new data-generating conditions such as the abundance of location services, the pervasiveness of mobile devices, and the pervasiveness of sensors. It can manage high-speed data streams from sources such as the Internet of Things, sensors, market data, mobile, and clickstream in real time. The essential principle of this paradigm is that data's potential worth is determined by its freshness. As a result, contrary to batch computing, which requires data to be stored first before being reviewed; data is examined as soon as it enters in a stream to produce findings. Scalable computing systems and parallel architectures are in high demand. [9].

Companies benefit from streaming analysis when they need to extract crucial information from current events in a timely and effective manner so that they can respond quickly to problems or discover new trends that can help them improve their performance. However, scalability, integration, fault tolerance, timeliness, consistency, heterogeneity and incompleteness, load balancing, privacy concerns, and accuracy must all be addressed due to the nature of enormous data streams. [9].

The challenge is determining how an organization can ensure that effective logging is deployed in required locations, and that the logs are gathered and processed in order to derive information that can be used to add value. The information contained inside the massive collection of logs is frequently unavailable elsewhere. Because of the high velocity of logs, real-time ingestion,

processing, and visualization are extremely challenging to achieve with traditional data processing methods. This emphasizes the need of establishing effective log analysis discipline inside a business. [13].

This research assesses various studies and trending technologies of managing log data using big data technologies in order to design and demonstrate state-of-the-art efficient, scalable and effective big data platform capable of on-the-fly ingestion, processing and visualization of streaming logs from different data sources. It also presents an in depth analysis into current log analysis domains, common problems as well as latest trends of using big data technologies to efficiently process system and network device logs. Moreover It describes a centralized framework we developed for semi-structured log analysis with the view of providing a solution to open problems in the domain.

1.2 Statement of the problem

Nowadays, organizations are overwhelmed by external data sources on top of their internal data source which is called dark data. The challenges that organizations are facing is multifaceted such as the availability of big storage space (volume), the speed at which data creations takes place (velocity) and diversification of data types (variety).Tsunami of data from smarter planet whereby Internet of Things and people are introducing values extractions from big data using various analytics technologies and architectural frameworks [14].

The value of real-time data processing and monitoring various information security risks is typically overlooked by most small and medium-sized businesses (SMEs) and even large businesses. In the sphere of information security, it is always preferable to take a proactive rather than a reactive approach. These tools are usually simple to use, so they can be used by anyone with basic technical knowledge, and even a small business can use them to generate results and take appropriate action. [15].

An organization must always be aware of the overall status of its IS systems and IT infrastructure at all times. System logs are one approach to keep track of the condition of an environment at all times. Despite the fact that a network can have hundreds or thousands of nodes producing logs, most logs are rarely examined due to their complexity and volume. The volume and complexity of log files has grown to the point where manual analysis of system logs by administrators is no longer feasible. This fact necessitates the development of tools and procedures that will enable some type of automation in the management and analysis of system logs. [15].

The efficiency of big data architecture must match that of the organization's supporting infrastructure. Data from machines or sensors, as well as enormous public and private data sources,

are generating increasing amounts of data that are not well organized or simple. Previously, most businesses were unable to capture or store large amounts of data. Existing processing tools are also incapable of delivering entire results in a timely manner. The adoption of new big data technologies, on the other hand, has aided in performance enhancement, product and service innovation, and decision-making. The three major motives for big data technology implementation are to minimize hardware costs, check the value of big data before committing significant company resources, and reduce processing costs [16] [17].

The proper balancing of a limited quantity of log management resources with a continuous supply of log data is a fundamental challenge with log management that many organizations face. A vast number of log sources, inconsistent log content, formats, and timestamps among sources, and growing huge volumes of log data can all complicate log generation and storage. Log management also includes safeguarding the logs' confidentiality, integrity, and accessibility. Another issue with log management is making sure that security, system, and network managers execute effective log data analysis on a regular basis. [18].

Because of the booming of number and variety of academic and research institutes in our country in the past couple of decades, there is significant amount of research being published by students and researchers. Although there are some rare efforts in applying big data for various use cases in our country such as financial datasets, unstructured data management, mobile user data, cancer prediction from medical dataset etc..., almost all of them solely focus on the challenge of processing large volume of data (i.e. only Volume of the 3Vs) to recommend the other Vs for further research.

1.3 Research Questions

This research also raises the following research questions:-

1. How big data technologies can help real-time management of log data?
2. What are the current architectural trends for designing big data applications?
3. What are the challenges of applying big data technologies for log data management?
4. Which stream processing big data Frameworks and Technologies are more suited for log analysis and monitoring?

1.4 Objective of the study

1.4.1 General Objective

The main objective of this research is to design and demonstrate an integrated, service oriented big data architectural solution framework for efficient, resilient and fault tolerant log management and real-time monitoring for selected standard log types.

1.4.2 Specific Objectives

- ➢ To assess the current big data technology architectural trends and related researches
- ➢ To design a unified streaming big data solution pipeline to standard logging sources
- ➢ To explore efficient data processing and visualization techniques
- ➢ To experiment streaming log event data sets of Big Data
- ➢ To evaluate efficiency and effectiveness of the model
- ➢ To recommend further works in the domain

1.5 Scope and Limitation of the Study

The scope of this research is management of streaming logs from remote data sources to efficiently process and visualize in real-time using available open-source big data technology stacks. It is all about designing and experimenting a chain link of big data technologies to form a data pipeline that will collect target data source from selected remote machines to process and visualize in real-time. Metric data, Network flow data, Syslog file format and audit log are the selected data sources. It also thoroughly reviews existing trends and open source log management systems in the market. Managing velocity and variety characteristics of big data are the main focus areas of this research.

The limitations of this work include, big data platforms and frameworks that are proprietary and commercial are not considered. The designed log management solution, in its entirety; only based on open-source tools and frameworks. Data sources other than log files are not part of this research. The other is, volume characteristics of big data, is out of the scope of this study, since tools such as Apache Hadoop are solely designed to address the issue. Sub components of Hadoop, Hadoop distributed file system (HDFS) and Mapreduce are capable of efficiently storing and processing of high volume data respectively.

1.6 Significance of The study

Nowadays, every business, large or small, relies on information technology for some or all of its operations, and this necessitates the processing of data as it is generated. The usefulness of on-the-fly data processing and monitoring various information security risks is typically overlooked by most small and medium-sized businesses (SMEs) and even large businesses. In the sphere of information security, it is always preferable to take a proactive rather than a reactive approach. Once developed, tools like this are simple to use, so they can be used by anyone with minimal technical expertise, and even a small business may use them to create results and take appropriate action. **[15]**.

Thanks to deployment of various Information systems infrastructures in organizations, nowadays IT departments are tasked with monitoring and analyzing system failures and security incidents in real time in order to significantly decrease MTTR(Mean time to repair) time of systems and prolong MTTF(mean time to fail) of system. SOC(security operation centers) and system administers as well as some database administrators will greatly benefit from centralized log managements systems such as one designed in this research.

This research work also paves path for more academic research works in the area of big data analytics and specifically in streaming analytics and log management by serving as a reference material and initial point to dig more.

1.7 Organization of the Document

The rest of this Thesis is organized as follows:-

Chapter Two Relevant literature has been thoroughly reviewed. It starts by discussing fundamental concepts and terms and then progresses to the need for logging and how big data going to help manage logs. It also presents various architectural considerations for designing big data streaming systems. Various open source frameworks and technologies are also summarized in this section. Finally the chapter is concluded by presenting related research works.

Chapter three briefly describes the general research methodology used in this study. The proposed architecture for unified log management and monitoring systems is also revealed in this chapter. A selected DSR process model is followed and each step is discussed in a detailed manner.

Chapter four explains the step by step procedures and methods followed to develop the prototype system of unified log management and monitoring system. Results and evaluations are also part of this chapter.

In chapter five, Final conclusions and recommendations are discussed.

CHAPTER TWO

LITERATURE REVIEW

This chapter is intended to describe the basic concepts, ideas and related works relevant to the subject under investigation. The review is organized into three broad sections. Section one will thoroughly investigate various definitions, theories, and characteristics of the topic in question. Section two presents a survey of various trending big data architectures, frameworks and technologies currently used for stream processing in the big data community where as the last section will thoroughly assesses relevant related works.

2.1 Basic concepts & Overview

Information age, the economic industry in the 21st century, is highly dependent on data. But when we analyze the magnitude, how much data must be processed to be of meaningful use? A research work by [3] states only 0.5% of globally generated data is analyzed. A world where "every 2 days we create as much information as we did from the beginning of time up to 2003" , there is a need to bridge data analyzed with current trends to better business models. By the year 2020, it is estimated that, there will be approximately 100 billion connected devices leading to more abundance of data which in turn signifies the necessity for applying big data analytics. This brings the essence of investigating and understanding Big Data [3].

Although the term "Big Data" is trending in the 21st century, there is no a standard definition for it still now. The concept of big data analytics is continually growing as organizations remodel their operational processes to rely on live data hoping to drive effective marketing techniques, improve customer engagement, and to potentially provide new products and services [3].

2.1.1 Basics of Big data & Big data Analytics

The term "Big Data" was first introduced to the computing world by Roger Magoulas from O'Reilly media in 2005, in order to define a large amount of data that traditional data management techniques cannot manage and process due to the complexity and size of this data. Madden define the Big Data as: "data that's too big, too fast, or too hard for existing tools to process." Despite the fact that there is no universally accepted definition for big data, it refers to large sets of complex data, both structured and unstructured which traditional processing techniques and/or algorithms are unable to operate on. The primary aim is being to reveal hidden patterns and has led to an evolution from a model-driven science paradigm into a data-driven science paradigm [19].

According to a recent study, big data rests on the interplay of Technology, Analysis and Mythology. Big data Relies on Technology to maximize computational power and algorithmic accuracy to collect, analyze, link, and compare large data sets in timely and efficient manner. Whereas, Analysis will help draw on large data sets to identify patterns in order to make economic, social, technical, and legal claims. Mythology, on the other hand, is the widespread belief that large data sets usually offer a higher knowledge that can generate insights that were previously not possible, with the aura of truth, objectivity, and accuracy [7].

IBM scientists mentioned that Big Data has four-dimensions: Volume, Velocity, Veracity and Variety. Gartner stipulates on the idea of IBM by stating that Big Data consists of "high-volume, velocity and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making" [19].

The term big data is commonly used for a gathering of data sets so large and dimensionally complex that it gets difficult to process them accurately and in timely fashion, as the information comes from multiple, autonomous sources with complex and evolving relationships, and keeps growing. The datasets that is used in this case are very big, they are measured in Exabyte. With this tremendous increase of huge global data, the notion big data is mainly used to describe gigantic datasets. When we compare it with traditional datasets, big data usually includes masses of mainly unstructured data that need more real-time analysis. It is about growing challenges that corporations face as they deal with huge and enormously growing sources of data. The data could come in various formats such as numbers, maps, videos, pictures, words and phrases, and many more. Some Examples of big data include customer reviews on commercial websites, photos and videos posted online, comments on social networking websites, electronic medical records, and bank records [20].

2.1.2 Stream computing

Stream computing generally refers to the processing of enormous amount of data generated at high-velocity which comes from multiple sources with low latency and computed in real-time. It is a new paradigm which came to existence because of new sources of data generating scenarios which include ubiquity of location services, mobile devices, and sensor pervasiveness. It can be applied to the high-velocity flow of data from real-time sources such as the Internet of Things(IOTs), Sensors data, market data, various ubiquitous computing platforms, Mobile and clickstream data [21].

A more Formal definition for Stream processing could be the in-memory, record-by-record analysis of data in motion with an objective of extracting actionable intelligence as streaming analytics, and

to react to operational exceptions through real-time alerts and automated actions averting and or correcting problems. Stream processing is designed to manage real-time streaming. It supports applications that generate data from heterogeneous sources and are pushed to processing servers. These applications are normally deployed as continuous jobs that run from inception to the time they are canceled [19] [22].

A stream processing system is generally designed to handle data streams and to manage continuous queries. It executes continuous queries that are not only once performed, but are continuously executed until they are explicitly terminated or uninstalled. As long as new data arrives in the system, it will not stop producing results and data is processed on the fly without the necessity of storing it first. But that does not necessarily mean Data is not usually stored after processing. Stream processing systems differ from batch processing systems, due to the requirement of real time data processing. Real-time processing guarantees that a certain process will be executed within a given period, maybe a few seconds or microseconds, depending on the quality of service constraints. The term "real-time" is a bit redundant but many systems use the term to describe themselves as low latency systems [2] [7].

The fundamental assumption of Stream processing is the fact that the potential value of data lies in its freshness. Hence, data are analyzed as soon as they arrive in a stream to produce result as opposed to what obtains in batch computing where data are first stored before they are analyzed. There is a crucial need for parallel architectures and scalable computing platforms. With stream computing, organizations can analyze and respond in real-time to rapidly changing data. The principle of using streaming data into decision-making process signifies a programming paradigm called stream computing [9].

Stream processing solutions are required to handle a high volume of data, coming in real-time from diverse sources putting into consideration availability, scalability and fault tolerance. Big data stream analysis involves assimilation of data as an infinite tuple, analysis and production of actionable results usually in a form of stream In a stream processor, applications are represented as Data Flow Graph(DAG) which are made up of operations and interconnected streams [9].

In a streaming analytics system, applications mostly comes in a form of continuous queries, data are ingested continuously, analyzed and correlated, and stream of results are generated. Usually a set of operators connected by streams, Streaming analytic systems must be capable of identifying new information, incrementally build models and access whether the new incoming data deviate from model predictions [9].

As mentioned earlier, the main logic behind Streaming analytics is that each of the received data tuples is processed in the data processing node. The process may be a kind of filling missing data, removing duplicate files or records, parsing, feature extraction and data normalization which are usually done in a single pass because of the high data rates of external feeds. Whenever a new tuple or record arrives, this node is triggered, and then it expels tuples or records older than the time specified in the sliding window. A window is basically a logical container for data tuples received. It defines the rate at which data is refreshed in the container or window as well as when data processing is triggered [21].

Streaming applications can be represented using a Directed Acyclic Graph (DAG), in which vertices are the operators and the edges are the channels for dataflow between operators. The processing then can go through operators in a particular order, where the operators can be chained and pipelined together, but the processing may never go back to an earlier point in the graph (it should not traverse back to earlier points). So far there are two execution models used in stream processing:

- The first is The Stream-Dataflow Approach, in which an application is viewed as a dataflow graph with operators and data dependencies between them (where sometimes mentioned as operator-based approach). A task encapsulates the logic of a predefined operator like filter, window, aggregate or join or even a routine with user-specified logic. A data stream between two entities represents an infinite sequence of data. Data is delivered and consumed in respective order by applying parallel. The whole scenario works by automatically pipelining everything.
- The Micro-Batch Approach, on the other hand, enables processing of data streams on batch processing systems. With micro-batch approach, one can treat a streaming computation as a sequence of transformations on bounded sets by discretizing a distributed data stream into a series of batches, and then scheduling them sequentially in a cluster of distributed worker nodes [2].

[22] Asks the obvious questions regarding stream processing solutions is that what kinds of challenges must it solve?

- Process large amounts of streaming events
- Integration with existing infrastructure
- Performance and scalability due to increase of datasize and complexity
- Automated alerts and reactions.

- Live data discovery and monitoring
- Quick time-to-market for application development and deployment
- Real-time responsiveness to changing market conditions
- Developer productivity throughout the software development life cycle (SDLC)
- Continuous query processing
- End user ad hoc continuous query access
- Push based visualization.

2.1.3 What is log and log management?

[23] defined log as "*A log is a record of the events occurring within an organization's systems and networks. Logs are composed of log entries; each entry contains information related to a specific event that has occurred within a system or network. Originally, logs were used primarily for troubleshooting problems, but logs now serve many functions within most organizations, such as optimizing system and network performance, recording the actions of users, and providing data useful for investigating malicious activity. Logs have evolved to contain information related to many different types of events occurring within networks and systems. Within an organization, many logs contain records related to computer security; common examples of these computer security logs are audit logs that track user authentication attempts and security device logs that record possible attacks. This guide addresses only those logs that typically contain computer security-related information.*" [23].

Log management is a broad term that refers to a collection of activities and processes for gathering, generating, centralizing, transmitting, parsing, storing, archiving, and disposing of massive amounts of computer-generated log data. Those tools are used to manage all of the logs created by various systems, applications, software, networks, or users, and to deal with them in the most efficient way possible for an enterprise or organization. Not only among system administrators and security operators (SOC operators), but also among developers and researchers, log management is a hot subject. The primary reason for this is that logs are used for security, performance improvement, and troubleshooting through many IT segments and job functions, and it is still increasing and expanding [23].

The number, volume, and variety of computer logs has significantly increased in recent years, necessitating the need for computer log management, which is critical for ensuring that computer security records are maintained in adequate detail for a sufficient period of time. Security incidents, policy breaches, illegal behavior, and organizational issues can all be identified with regular log

review. Auditing and forensic research, assisting internal audits, setting baselines, and detecting organizational patterns and long-term issues all benefit from logs. Certain logs can also be stored and analyzed by organizations in order to comply with federal laws and regulations such as the Federal Information Security Management Act of 2002 (FISMA), the Health Insurance Portability and Accountability Act of 1996 (HIPAA), the Sarbanes-Oxley Act of 2002 (SOX), the Gramm-Leach-Bliley Act (GLBA), and the Payment Card Industry Data Security Standard (PCI DSS) [23].

The effective balancing of a finite quantity of log management resources with a continuous supply of log data is a fundamental problem with log management that many organizations face. A large number of log sources, inconsistent log content, formats, and timestamps among sources, and increasingly large volumes of log data can all complicate log generation and storage. Log maintenance also entails safeguarding the logs' confidentiality, integrity, and accessibility. Another issue with log management is making sure that protection, system, and network administrators analyze log data effectively on a regular basis [23].

Log messages, or logs, are at the core of log data. In the world of computing, a log is a file that is created automatically any time such events occur in the system, is normally time-stamped, and can record virtually everything that happens behind the scenes in operating systems or software applications - in a nutshell, they record anything that the server, network, OS, or application feels is necessary to keep track of. Messages and transactions between various users, what occurred during a backup, and failures that prevented an application from running can all be recorded in logs or the files that have been requested by users from a website. A log message is the production of a computer system, unit, program, or other device in response to a stimulus. What the triggers are depends a lot on where the log message came from. UNIX systems, for example, will produce log messages for user login and logout, firewalls will generate ACL approve and reject messages, and disk storage systems will generate log messages when errors occur or, in some cases, when the device detects an imminent failure [24].

A log message's intrinsic value is referred to as log data. Alternatively, log data is the information extracted from a log message that explains why the log message was created. When anyone accesses a resource (image, file, etc.) on a Web page, for example, a Web server will often record it. The user's name would appear in the log message if she had to authenticate herself to access the website. This is an example of log data: the username can be used to figure out who accessed a resource. The word logs actually refers to a list of log messages that would be used together to create an image of an event [25].

There are different kinds of logs: some can be opened and read by humans, while others are only held for auditing purposes and are not readable by humans. Audit logs, transaction logs, event logs, error logs, and message logs are only a few examples of log files, each with distinct function. They have a wide range of extensions, including .log, and other proprietary extensions. They can be opened with a regular text editor (such as Notepad) or word processing software (Open Office or Microsoft Word) depending on the extension and readability, or they can require the installation of specific applications [23] [24].

The following general categories can be used to classify log messages:

- Informational: - Messages of this type are designed to let aware that something benign has occurred.
- Debug:-are generally generated from software systems in order to aid software developers troubleshoot and identify problems with running application code.
- Warning:-are concerned with situations where things may be missing or needed for a system, but the absence of which will not impact system operation.
- Error:-are used to relay errors that occur at various levels in a computer system.

Log management (LM) is a method for dealing with large quantities of log messages created by computers (also known as audit records, audit trails, event-logs, etc.).Log compilation, centralized log aggregation, long-term log storage and retention, log rotation, log review (in real-time and in bulk after storage), and log search and reporting are all examples of log management [25].

Figure 1 Log management life cycle [28]

2.1.4 How is Log Data Transmitted and Collected?

The transmission and processing of log data is conceptually straight forward. A logging subsystem is implemented in a machine or device so that it can produce a message whenever it sees fit. The method used to make the determination varies depending on the unit. You can, for example, be able to customize the device or the device may be hard-coded to send a pre-determined list of messages. On the other hand, you must have a location where log messages are sent and received. A log host is the term used to describe this location. A log host is a computer device that collects log messages in one place. The following are some of the benefits of using a central log collector:

- It's a place for storing backup copies of your logs.
- It's a place where analysis can be performed on you log data.
- It's a centralized place to store log messages from multiple locations.

However, one might wonder how log messages are transmitted in the first place. The Syslog protocol is the most popular method. The Syslog protocol is a log message interchange standard. It's most popular on UNIX systems, but it's also available on Windows and other non-Unix platforms. While several open source and commercial Syslog implementations also support the Transmission Control Protocol (TCP) for guaranteed delivery, there is essentially a client and server portion implemented over the User Datagram Protocol (UDP).The actual device or computer system that produces and sends log messages is referred to as the client component. A log collection server is usually where the server side is found. Its primary contribution is to take a note of Syslog- based log messages and store them to local disk storage where they can be analyzed, backed up and stored for long-term use [26].

The Simple Network Management Protocol (SNMP) is a protocol for managing networked devices that is based on industry standards. Traps and polling are the foundations of the protocol. A trap is a type of log message that a device or computer system sends out when something unusual occurs. An analogous to a log host, a trap is sent to a management station. SNMP-based systems are managed using a management station. Polling is the process of using SNMP to query a system for pre-defined variables such as interface from a management station [27].

Databases have recently become a popular option for storing log messages in applications. An application can write its log messages to a database schema instead of creating a Syslog message. In

certain instances, the Syslog server can write a relational database directly. This has a lot of benefits, particularly in terms of storing, analyzing, and reporting log messages in a structured way [28].

Finally, there are proprietary logging formats. These are third-party devices and software that create and retrieve log messages using their own proprietary mechanisms. In this case, the provider either offers an Application Programming Interface (API) in the form of C or Java libraries, or you are left to your own devices to implement the protocol. Although the Windows Event Log is a proprietary format, it is often regarded as an unofficial logging standard, similar to Syslog, due to its widespread use [24].

Some of the more common protocols we have discussed can be summarized as follows:

#	**protocols**	**description**
1	Syslog	UDP-based client/server protocol. This is the most common and prevalent mechanism for logging.
2	SNMP	It was originally created for use in managing networked devices. However, over the years, many non-networked systems have adopted SNMP as a way to emit log message and other status type data.
3	Windows Event Log	Microsoft's proprietary logging format.
4	Database	Structured way to store and retrieve log messages.
5	Common Proprietary Protocols:	Different proprietary systems are shipped with custom log collection protocols, systems and/or APIs.

Table 1 Major Protocols for Log collection

2.1.5 Benefits of Logging

Security, compliance & audit, IT operations, DevOps, and MSSP are just a few of the use cases for log analysis tools. Resource management, device troubleshooting, regulatory enforcement & SIEM, business analytics, and marketing insights all benefit from log management [24].

Here let's study three primary reasons for logging across the entire spectrum of log sources.

First and foremost, security logging is concerned with detecting and reacting to threats, malware infections, data theft, and other security concerns. The recording of user authentication (login) and other access decisions with the aim of analyzing whether someone has access to the resource without proper authorization is a classic example of security based logging [28].

The other is to conduct operational logging in order to provide valuable information to device managers, such as malfunction notifications and potentially actionable situations. Service provisioning and even financial decisions can benefit from operational logging (access-based pricing, Web server access logs are a common example of logs used for business not just for IT operations) [23].

Furthermore, since regulations are often written to enhance the protection of systems and data, compliance logging often overlaps with security logging. It's crucial to understand the distinction between two forms of compliance logging: regulations and mandates that apply to IT, such as PCI DSS, HIPAA, ISO, ITIL, and others, and system regulations, such as common requirements and other mandates on system design and security [23] [24].

Finally, application debug logging is a form of logging that is useful to developers of applications and systems, not to system operators. In most production systems, such logging is disabled, but it can be allowed upon request. Many of the messages in debugging logs can be analyzed by an application developer who has complete knowledge of the application's internals and, in some cases, even has access to the source code.

These four types of logging are produced by all log sources (event producers), but are analyzed and consumed differently and by different systems (event consumers).Starting from the days of early Unix send mail program in the 1980s,Standard Unix systems have a logging technology called syslog. The operating system itself and some major system and user applications write message to syslog. If the system administrator has not disabled logging or removed the logs files (“because the disk was getting full”), he/she might see entries like the following:

Nov 11 22:41:00 ns1 named[765]: sysquery: findns error (NXDOMAIN) on ns2.example.edu?

The entry contains a date, host name, name of the service that generated the message as well as the message it-self. This particular message indicates an attempt to do a DNS lookup on a non-existent domain.

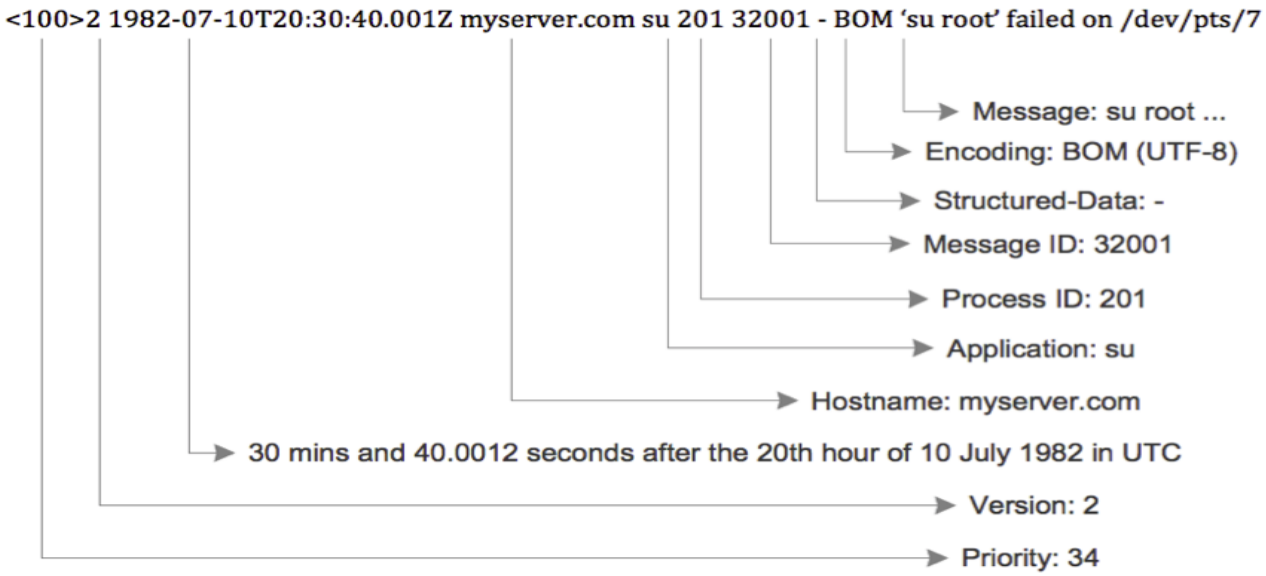


Figure 2 Syslog Format Syntax Example

2.1.6 Why is log management so challenging using traditional log analysis tools?

Despite an organizations size, whether Large or Medium, Most organizations face similar log management-related challenges, which have the same underlying problem: effectively balancing already limited amount of log management resources such as log management systems and its entire infrastructure it needs ,with an ever-increasing and overwhelming supply of log data. When we discuss the most common types of challenges, they are mostly divided into three groups. First, there are several potential problems with the initial production of logs that arises due to variety and prevalence of the log data. Secondly, the confidentiality, integrity, and availability (CIA) of generated logs could be breached inadvertently or intentionally. Finally, the people responsible for performing log analysis are often disappointingly not prepared enough and supported well. In a typical organization, many hosts' OSs, security software, and other applications generate and store logs. This complicates log management in the following ways:

Many Log Sources: - Logs are located on enormous hosts throughout the organization, which necessitates log management. In addition, a single log source can generate multiple logs—for example, an application storing authentication attempts in one log and network activity in another log.

Inconsistent Log Content: - In its log, each log source records those entries, such as usernames and host IP addresses. Log sources also record only the pieces of information that they consider most relevant for the sake of efficiency. Since they may not have any common values recorded, this may make it difficult to connect events recorded by different log sources (e.g., source 1 records the source IP address but not the username, and source 2 records the username but not the source IP

address). Each type of log source may also represent values differently; these differences may be slight, such as one date being in MMDDYYYY format and another being in MM-DD-YYYY format, or they may be much more complex, such as use of the File Transfer Protocol (FTP) being identified by name in one log ("FTP") and by port number in another log. This further complicates the process of linking events recorded by different log sources.

Inconsistent Log Formats: - Comma-separated or tab-separated text files, databases, syslog, Simple Network Management Protocol (SNMP), Extensible Markup Language (XML), and binary files are among the log source types that use various formats for their logs. Some logs are intended to be read by humans, while others are not; some logs follow common formats, while others follow proprietary formats. Some logs are generated for transmission to another device for processing, rather than for local storage in a file; SNMP traps are a good example of this. There are several options for the sequence of the values in and log entry and the delimiters between the values in some output formats, especially text files (e.g., comma-separated values, tab-delimited values, XML).

Inconsistent Timestamps: - When setting a timestamp for each log entry, each host that generates logs usually refers to its internal clock. If a host's clock is off, the timestamps in its logs will be off as well. This can make log analysis more difficult, particularly when multiple hosts' logs are being analyzed. For example, timestamps can show that event A occurred 45 seconds before event B, when in fact, event A occurred two minutes later.

Log retention: - Generally speaking, log files are kept in the system for significant amount of time for future and further analysis. Despite the fact that log lines are usually small bytes of text data, they may accumulate over time and become too large and storing in local disk space could become increasingly difficult.

Log Protection:- inadvertently storing sensitive information such as users' passwords and the content of e-mails also may raise security and privacy concerns involving both the individuals that review the logs and others that might be able to access the logs through authorized or unauthorized means [29].

2.1.7 Why we need big data?

Big data analytics in general, examines large amounts of data to uncover hidden patterns, correlations and other useful insights. With the help of current technologies, it has become feasible to analyze your data and get answers out of it in a very short period of time, an effort that is much slower with more traditional business intelligence (BI) solutions. Big Data analytics can assist companies in better understanding the data and identifying the data that is most vital to the

organization and future business decisions. Analysts dealing with Big Data are frequently interested in the information gained from data analysis. Tom Davenport, IIA Director of Research, interviewed more than 50 businesses for his study Big Data in Big Companies to learn how they used big data. He discovered that they were valuable in the following ways.

- **Cost Reduction.** When it comes to storing large amounts of data, Big Data technologies like cloud-based analytics provide substantial cost savings. They can also find more effective ways to do business.
- **New Products and Services**. With the ability to use analytics to gauge consumer desires and satisfaction comes the ability to offer consumers exactly what they want. According to Davenport, more businesses are using big data analytics to create new products to meet the needs of their consumers.
- **Faster, Better Decision Making**. Businesses can evaluate information quickly –and make decisions based on what they've discovered – thanks to hardtop's pace and in-memory analytics, as well as the ability to analyze new sources of data.

Big Data Analytics is usually conducted using advanced software tools and applications for predictive analytics, data mining, text mining, forecasting, and data optimization to analyze such a large amount of data. These processes are distinct but highly integrated features of high-performance analytics when viewed as a whole. Using Big Data tools and software allows a company to process incredibly large amounts of data in order to decide which data is important and can be analyzed in order to make informed business decisions in the future [30].

2.1.8 Characteristics of Big data

Big Data is vital because it enables organizations to collect, store, manage, and manipulate vast amounts data at the right speed, at the right time, to gain the right insights. In addition, Big Data generators must create scalable data (Volume) of different types (Variety) under controllable generation rates (Velocity), while maintaining the important characteristics of the raw data (Veracity), the collected data can bring to the intended process, activity or predictive analysis/hypothesis. Indeed, there is no clear definition for 'Big Data'. It has been defined based on some of its characteristics [31].

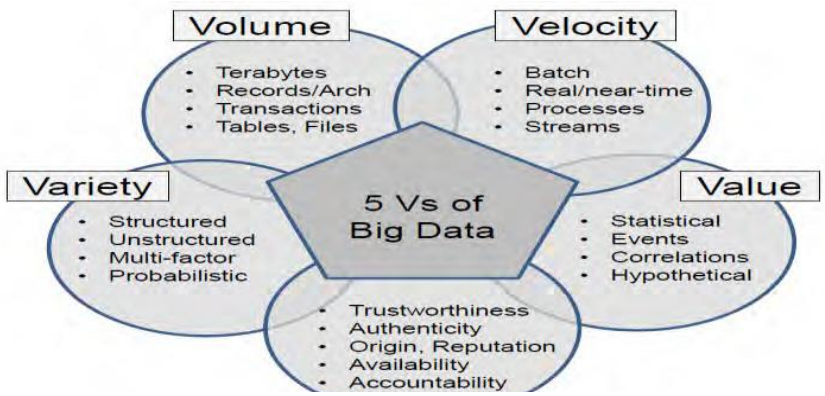


Figure 3 The 5Vs of Bigdata [32]

The quantity of data collected by an organization is referred to as volume. This data must be used in order to obtain useful knowledge. Enterprises are awash in ever-growing data of all kinds, with terabytes and even petabytes of data easily amassing (e.g. turning 12 terabytes of Tweets per day into improved product sentiment analysis; or converting 350 billion annual meter readings to better predict power consumption).Furthermore, according to some studies, volume is the most significant and distinguishing characteristic of Big Data, placing strict criteria on all commonly used conventional technologies and resources [32].

Variety on the other hand refers to the various types of data that Big Data can comprise. This data may be either structured or unstructured. Structured and unstructured data, such as text, sensor data, audio, video, click streams, log files etc., is classified as big data. Monitoring hundreds of live video feeds from surveillance cameras to target points of interest, using the 80 percent data increase of photographs, video, and documents to enhance consumer loyalty are just a few examples of how combining data types can lead to new issues, situations, and so on [32].

Velocity refers to the time in which the incoming Data can be processed. Fast processing maximizes productivity because certain tasks are extremely critical and require immediate responses. Big Data flows must be processed and used as they flow into organizations for time-sensitive processes like fraud detection in order to optimize the value of the information (e.g. scrutinize 5 million trade events created each day to identify potential fraud; analyze 500 million daily call detail records in real-time to predict customer churn faster) [32].

Value refers to the important feature of the data which is characterized by the added-value that the gathered data can bring to the intended activity, process or predictive analysis. The value of data is determined by the events or processes it represents, which can be stochastic, probabilistic, regular, or random. Depending on this, conditions may be placed to collect all data, store for a longer period of time (for any potentially interesting event), and so on. Data volume and variety are closely related to data importance in this regard [32].

Last but not least, Veracity refers to the degree in which the end user or analyst trusts information in order to use it for decision making. As a result, making the right Big Data connections is critical for the company's potential success. However, with one-third of business leaders distrusting the

information used to make decisions, establishing confidence in Big Data is becoming increasingly difficult as the number and variety of sources increases [32].

2.1.9 Big data Taxonomy

Big Data Types

Big data types can be usually classified into 3 different categories.

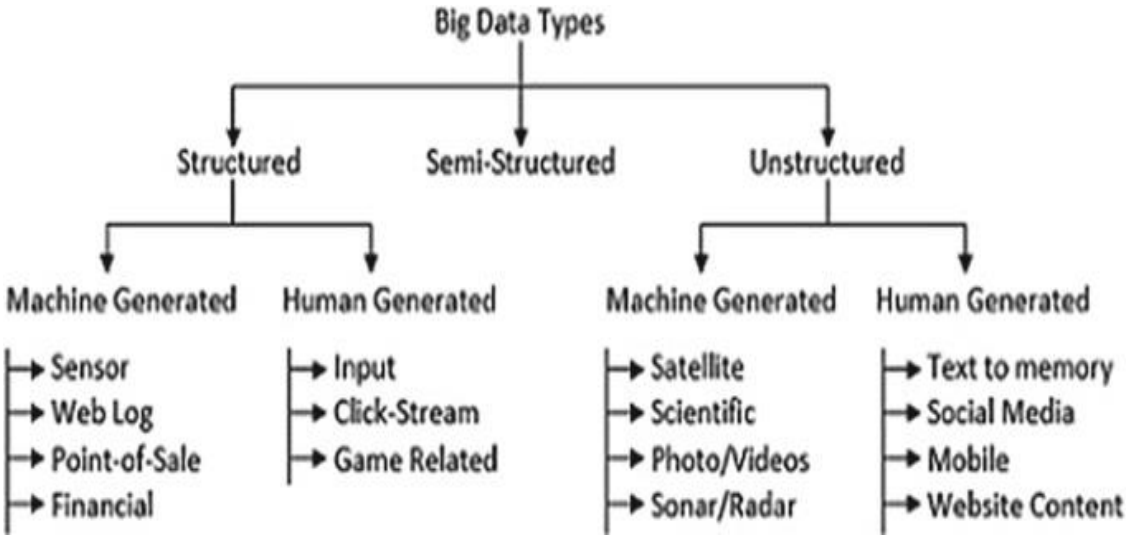


Figure 4 Big data Types and Classification [75]

Structured data: Numbers and words are examples of data that can be easily classified and analyzed hence structured data. It is primarily generated by network sensors embedded in electronic devices, such as smartphones and global positioning system (GPS) devices. Transaction records, revenue statistics, account balances, and other structured data are examples of structured data. This data form represents data that is organized into rows and columns in a standard database, i.e. a relational scheme. Based on an organization's requirements and organizational needs, the data configuration and consistency allows it to respond to simple queries to arrive at usable information.

Semi structured data: It's a form of structured data that doesn't follow a rigid and explicit schema.The data is inherently self-descriptive, with tags or other identifiers enforcing hierarchies of records and fields within it. Weblogs and social media feeds are two examples.

Unstructured data: It includes more complex data such as consumer feedback from commercial websites, images and other multimedia, and social networking site comments. These data are difficult to divide into groups or numerically evaluate. Big data consultancy vice president Tony

Lewitt of Plano, Texas, defines unstructured big data as "what humans are saying. “It speaks in a natural tone. This data is in formats that are difficult to index into relational tables for analysis or querying. Photos, audio, and video files are examples [33].

2.1.10 Types of Big data Analytics

Big data analytics or data analytics in general, based on the nature and type of the analysis used, can be classified into four categories.

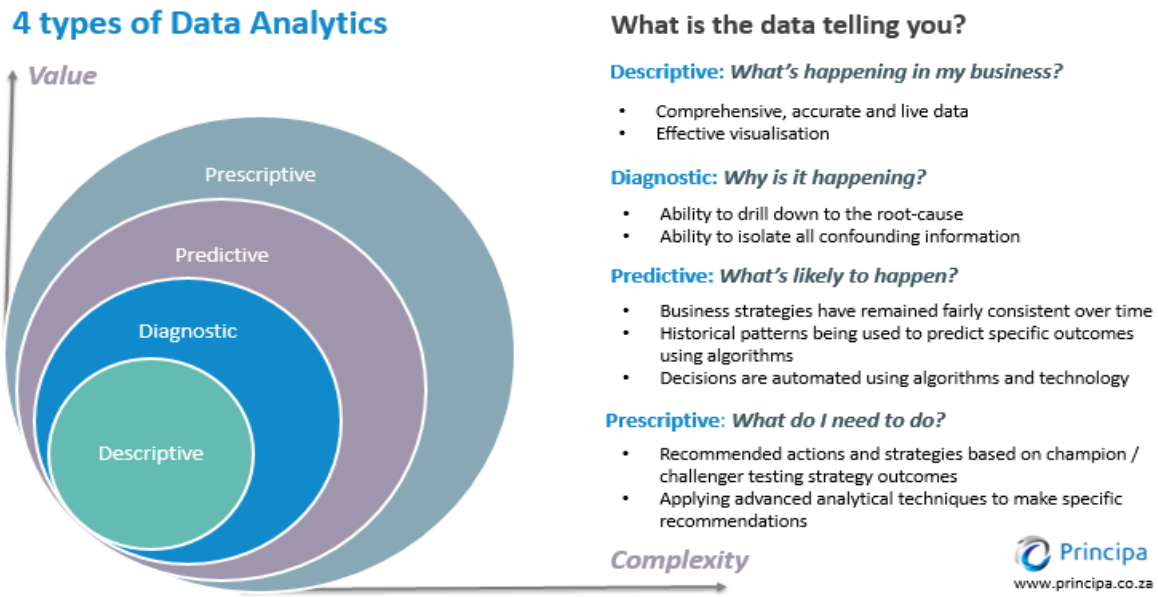


Figure 5 Big data Analytics types [76]

(a)Descriptive Analytics consists of asking the question: What is happening? It's a stage of data processing process that results in a collection of historical data. Data mining techniques organize data and aid in the discovery of trends that provide insight. Descriptive analytics forecasts potential odds and patterns, as well as what might occur in the future.

(b)Diagnostic Analytics consists of asking the question: Why did it happen? The aim of diagnostic analytics is to find the source of a problem. It's used to figure out why something happened the way it did. This approach seeks to identify and comprehend the causes of events and behaviors.

(c)Predictive Analytics consists of asking the question: What is likely to happen? It forecasts the future using historical evidence. It's all about predicting the future. Predictive analytics analyzes existing data and creates simulations based on what could happen when using a variety of techniques such as data mining and artificial intelligence.

(d)Prescriptive Analytics consists of asking the question: What should be done? It is committed to determining the best course of action. Predictive analytics helps forecast what could happen, while descriptive analytics offers historical data. These parameters are used by Prescriptive analytics to find the best solution [14] [20].

2.2 Architecture of Big data Applications

Generally speaking big data follows a layered architecture. Different layers present in big data architecture are explained below. The layered view of the architecture of big data is shown below.

- Data Source Layer: - It is the layer that accepts data from relational databases as well as structured, unstructured, and semi-structured data.
- Ingestion Layer: - It's the layer that imports and processes data, filters out noise, and stores it for later use by tools like Flume, Kafka, and Storm.
- Visualization Layer: - It is the layer that allows data scientists and analysts to gain insights and develop their ability to look at various aspects of big data using Hadoop Administration, Data Analyst IDE/SDK, and Visualization software.
- Analytics Engine Layer: - Statistical Analytics, Text Analytics, Search Engines, and Real Time Engines are all used in this layer to manage Big Data.
- Hadoop Platform Management Layer—Pig, Hive, Oozie, Map Reduce, Zookeeper, Impala, Spark, and Shark are used in this layer to handle the massive amount of data stored in HDFS.
- Storage Layer: - This layer stores data in a low-cost No-SQL database and uses Hadoop Distributed File System (HDFS) for high-speed distributed algorithm processing.
- Data Warehouse—It contains some analytic appliances and manages RDBMS-based data in a centralized environment.
- Memory: - The layers in which nodes are present where the data is stored or processed in Rack, Disk and CPU.
- Infrastructure Layer: - It includes bare metal clustered workstations, a virtualized cloud service, and Hadoop Storage Layer support. Data is stored in several different locations in a distributed model and connected together in this layer. The key goal is to verify the data redundancy.
- Security Layer: - Tasks such as Data Encryption, Data Access, Threat Detection, Nodes Authentication and tools such as Big-IP, Apache Sentry, and Apache Knox are used to maintain privacy and to meet the compliance requirements for proper authorization and authentication.
- Monitoring Layer: - The distributed clusters are monitored in this layer using tools such as Chukwa, Hue, Ganglia, Open TSBD, and Nagios, and large amounts of data are treated.

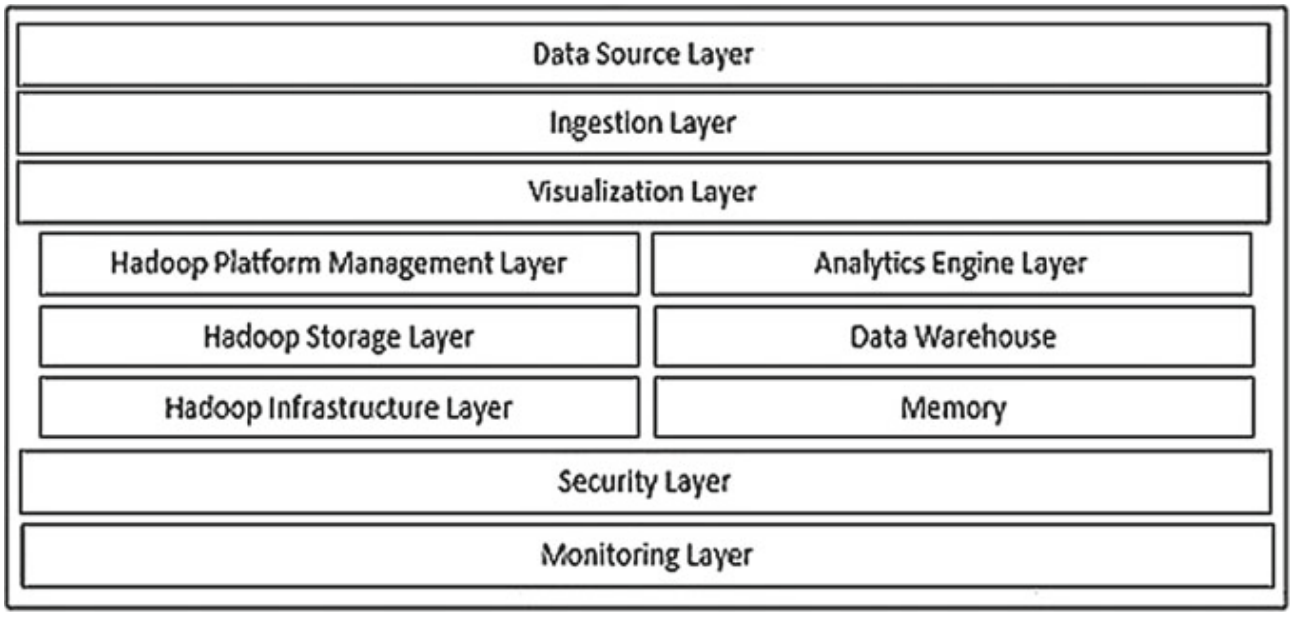


Figure 6Big data Architecture Layers [34]

Big Data architecture has seen rapid growth in recent years, and Gartner predicts that companies will continue to spend more in IT in 2018 and 2019, with an emphasis on IOT, Block chain, and Big Data. Data Center Systems cost 178 billion dollars in 2017, and this figure is projected to rise in the coming years. Given the large sums of money that businesses spend in Big Data solutions, it seems clear that proper preparation should be performed ahead of time before a solution is implemented. However, according to the McKinsey Institute, many companies are currently experiencing difficulties as a result of their data processing solutions' lack of architectural planning. Since they typically design technology-driven solutions, they develop conflicting functionalities and are unable to achieve sustainability [34].

Early and careful big data system architecture takes into account a comprehensive data approach when concentrating on real business goals and requirements. It's critical to write down not only current but also potential requirements in order to factor in scalability concerns from the start of the big data system's growth. If an organization has a clear list of use cases and specifications, it can step forward and choose the best Big Data architecture for its needs from among the many options available.

The Lambda architecture was one of the first to be proposed for Big Data processing, and it has since become the industry standard. The Kappa architecture was next designed to address some of the limitations of the lambda architecture for use cases where the former standard failed to provide satisfactory results. In the following section, we'll go over the various architectures and their optimal use cases, as well as some of the factors to consider when making the best decision [34].

Big Data architectures are designed to handle ingestion, processing, visualization, and analysis of data that is too massive or complex for conventional tools to handle. The way data is collected,

stored, analyzed, and visualized is determined by the Big Data architecture. We also use it to describe how to analyze and report on structured, unstructured, and semi-structured data.

We discuss in this section, five of the most prominent Big Data architectures that have gained recognition in the industry over the years.

A. **Lambda Architecture**

The lambda architecture is a big data processing technique that strives for low latency updates while retaining the highest accuracy possible. There are three levels to it. The first layer, called "the batch layer," is made up of a distributed file system that stores all of the collected data. The same layer also contains a collection of predefined functions that can be run on the dataset to generate a batch view. Those views are saved in a database, which serves as the "serving sheet," from which the user can query them interactively. The "speed layer" is the third layer, which computes incremental functions on new data as it enters the device. It only re-computationally processes data produced between two consecutive batch views, and it produces real-time views that are also stored in the serving layer. To achieve the most reliable results, the various views are queried together. Figure below shows a picture of this architecture [34] [35].

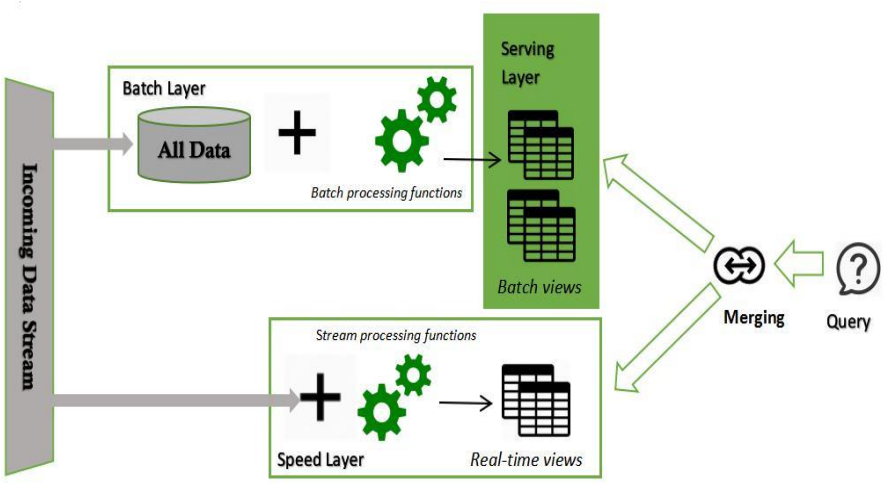


Figure 7 Lambda Architecture [34]

Advantages and Disadvantages of Lambda Architecture	
Advantages	**Disadvantages**
Better accuracy, higher throughput and lower latency for reads and updates simultaneously without compromise on data consistency	Maintaining the synchronization of the batch and speed layers is hard.
More resilient thanks to the Distributed File System used to store the master dataset	Only analytical operations are possible from the serving layer; no transactional operation is possible.
Less subject to human errors (such as unintended bulk deletions) than a traditional RDBMS.	The need to maintain two similar code bases one in the speed layer and another in the batch layer to perform the same computation on different sets of data.
Robustness and fault tolerance provided through the batch layer.	
Provides both real-time and batch processing	

Table 2 Advantages and Disadvantages of Lambda Architecture

Several businesses from various industries have implemented the Lambda Architecture over the last few years. Log ingestion and analytics are two areas where the Lambda architecture is especially useful. The explanation for this is that log messages are immutable and are frequently generated at a high rate in systems that need high availability. When there is a need for both real-time/fluid analysis of incoming data and periodic analysis of the entire repository of data collected, the Lambda Architecture is preferred. Analysis of social media, especially tweets, is an excellent example of such an application. Where data loss or manipulation is not an option and a large number of clients demand immediate input, such as in the case of a fraudulent claims processing system, the lambda architecture is a good choice [34] [36].

Some literature discovered that the batch layer's specifications make Hadoop the most suitable framework to use for its implementation when looking at suitable software requirements for Lambda architecture. To fit the master dataset, HDFS provides the ideal append-only technology. The batch functions can be developed using MapReduce, PIG, and Hive.

Real-time processing tools like Storm or S4 can be used to implement the speed layer. Spark Streaming can also be used, but it only works for micro-batches of data rather than actual streams. The Spark code can be reused in the batch layer, which is an advantage [34].

Any random-access NoSQL database will host the real-time and batch views in the Serving layer.HBase, CouchDB, Voldemort, and even MongoDB are some examples. Cassandra is especially popular because of the write-fast choice it offers. Moreover, to ensure asynchronous and

fault-tolerant transmission of real-time data to the batch and speed layers, a queuing system is needed. Apache Kafka and Flume are two popular alternatives [34].

B. Kappa Architecture

The Kappa architecture was proposed to reduce the lambda architecture's overhead that came with managing two different code bases for stream and batch processing. Its creator, Jay Kreps, observed that the need of a batch processing system came from the need to reprocess previously streamed data again when the code changed [37].

In Kappa architecture the batch layer was removed and the speed layer improved to provide reprocessing capabilities. By using specialized stream processing tools such as Apache Kafka, it is henceforth possible to store streamed data over a period of time and create new stream processing jobs to reprocess the data when it's required replacing batch processing jobs. The working mechanism is shown below [34].

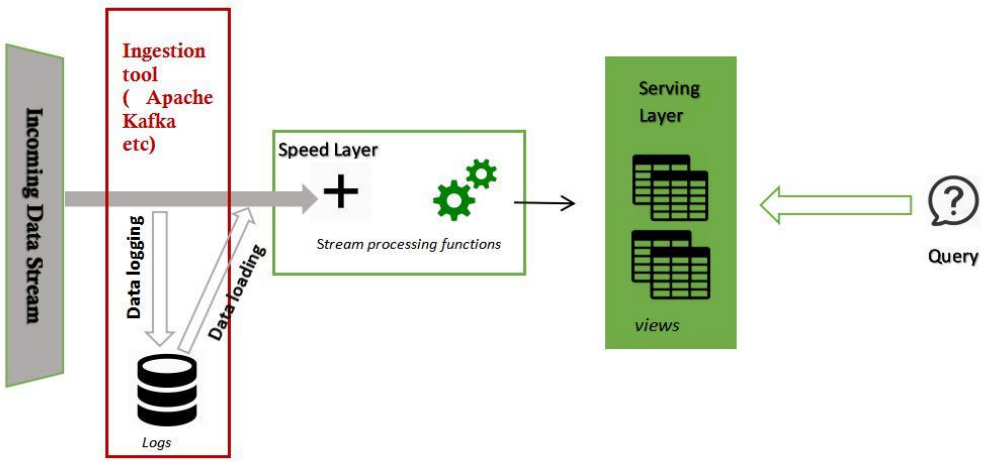


Figure 8 Kappa Architecture [34]

The advantages and disadvantages of the kappa architecture are summarized below.

Advantages and Disadvantages of Kappa Architecture	
Advantages	**Disadvantage**
Simple data processing	only analytical operations are possible not transactional ones
Allows historical data querying and analysis through replays.	Not native to cloud services
It has fewer moving parts than the Lambda architecture	they do not support streams with a long Time to live (TTL)
it allows replaying streams and re-computing results in case the processing code changes.	Short data retention policy by design

single code base	

Table 3 Advantages and Disadvantages of Kappa Architecture

The Kappa architecture is particularly suited for real-time applications because it focuses on the speed layer. Some authors introduced a comprehensive implementation of Kappa architecture for real time analytics of subscribers, network and social data collected by a telecom operator [34].

When it comes to suitable software technologies for kappa architecture, they are very similar to those for Lambda Architecture, with the exception of the Hadoop platform, which is not used to implement the batch layer. Apache Kafka is the preferred ingestion tool due to its ability to retain ordered data logs, allowing for data reprocessing, which is critical to the Kappa architecture. Because it allows for the creation of time windows for computations, Apache Flink is particularly well suited for implementing the Kappa architecture. Apache Samza is a popular alternative [34].

c. **The Zeta Architecture**

On the other hand, the Zeta architecture proposes a novel approach in which a company's technological solution is directly integrated with the business/enterprise architecture. Any business application can be "plugged in" to this architecture. It offers containers, which are isolated environments in which software can run and interact without regard for platform incompatibilities. As a result, zeta architecture can accommodate and run a wide range of applications [34] [38].

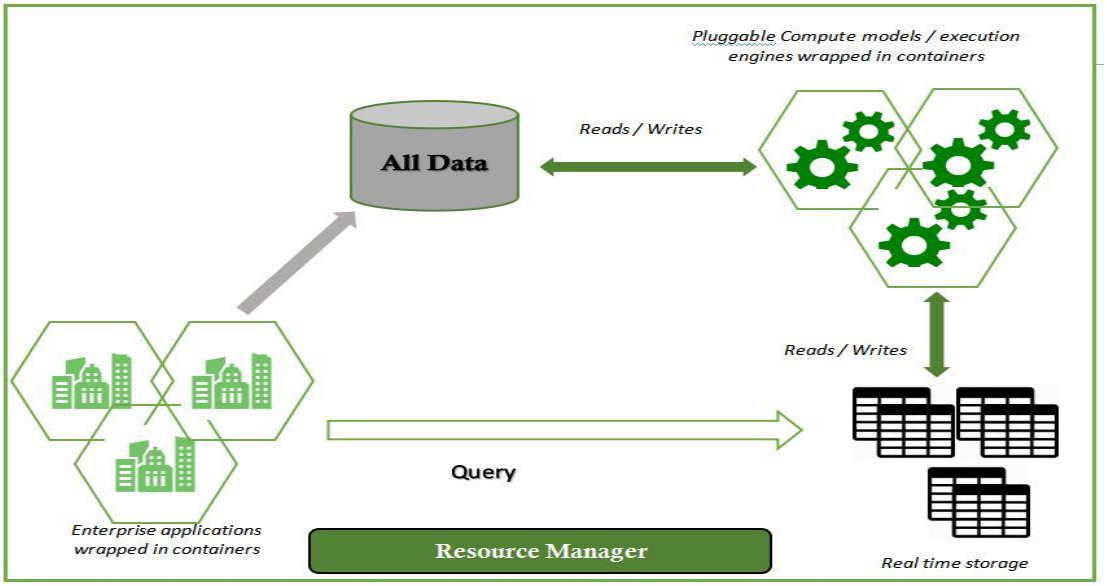


Figure 9 Zetta Architecture [34]

The major advantages and disadvantages of zetta architecture are summarized as follows.

Advantages and Disadvantages of Zetta architecture	
Advantages	**Disadvantages**
better utilized and it can be allocated to serve the most pressing need at any moment	Still on experimental phase and the big data community is actively investigating.
near real time backups also help avoid over extended recovery periods from failures	Segregated processes may lead to under-utilized resources
helps to discover issues more quickly	
It facilitates the testing and deployment phases by allowing the creation of binaries that can be deployed seamlessly in any environment without the need to modify them	

Table 4 Advantages and Disadvantages of Zetta Architecture

Organizations that handle real-time data processing as part of their internal business operations will benefit from the zeta architecture. For instance, a good use case that has been evoked is the dynamic allocation of parking lots based on data from sensors. It is the architecture that Google uses for systems like Gmail. The zeta architecture is also well-suited to complex data-centric web applications, machine learning-based systems, and big data analytics [38].

When it comes to the software stack for the zetta architecture, Hadoop Distributed File System is commonly used to host the master data, while NoSQL or NewSQL databases can be used for real-time storage (HBase, MongoDB, VoldDB etc.).The enterprise applications depicted in the diagram are typically web servers or other business applications (varying from one business to the other). All analytics operations will be performed by the compute model/execution engine. Any pluggable data processing tool can be used for this, including MapReduce, Apache Spark, and even Apache Drill. Apache Mesos or Apache YARN can serve as global resource manager. The container management system can be chosen among Docker, Kubernetes or Mesos [4] [34].

d. The IoT Architecture

Because the Internet of Things (IoT) domain is so vast, no standard architecture has yet been established in the field. Scholars have, however, proposed a number of architectures over time. Based on the requirements of an IOT data processing system, Michael Hausenblas has proposed high abstraction architecture for all IOT projects. The architecture is known as IoT and is depicted below.

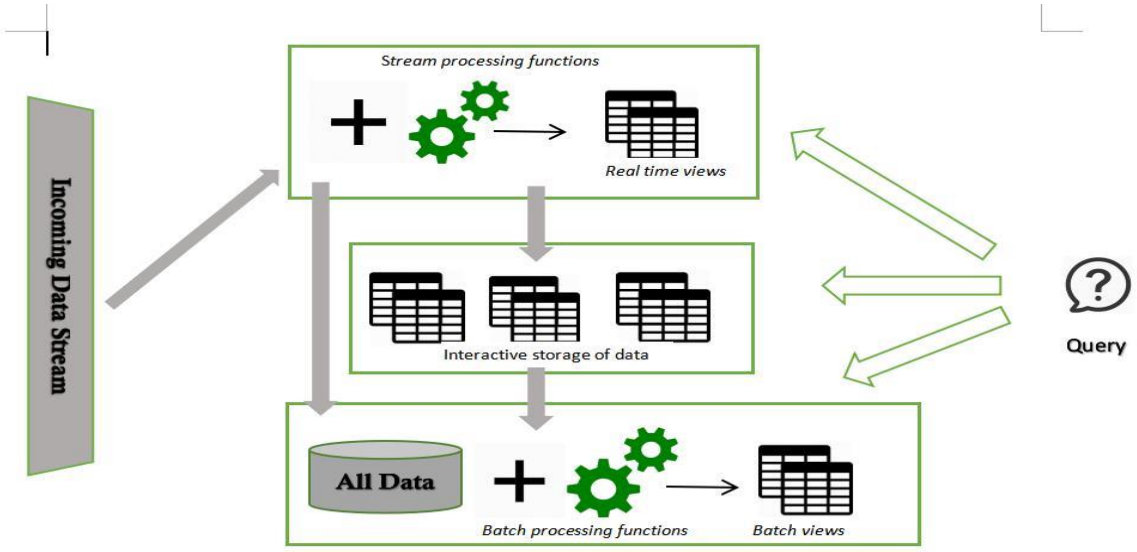


Figure 10 IoT-A architecture [34]

It's difficult to discuss the architecture's benefits and drawbacks because there hasn't been enough feedback on projects that have used it to provide a thorough assessment of its performance and, eventually, its flaws [34] [39].

The architecture being discussed is one that is intended to be a good fit for use cases such as smart homes and smart cities. In the automotive industry, for example, the Message Queue/Stream Processing layer can help alert a car user in real-time about failures, preventing accidents. The Database layer is used to query the system and obtain information about a car's status in order to perform a checkup or develop a repair strategy. Finally, the Distributed File System layer can enable a car owner to assess the overall metrics and performance of his vehicle on a weekly or monthly basis, potentially identifying problems [34] [4] [39].

When it comes to software stacks that are suitable for this architecture, the MQ/SP (Message Queuing and Stream Processing) layer could use Apache Kafka or Fluentd for data collection and Apache Spark or Storm for processing. Any NoSQL database can be used to enforce the interactive storage layer, and tools like Apache Drill can be used to communicate with it. For machine learning over the master dataset, the DFS layer may use HDFS, Hive, and Apache Mahout [34] .

2.3 Tools, Technologies, Solutions and frameworks

In this section we will thoroughly investigate widely used open source Big data Stream processing and Log management tools, technologies, solutions and frameworks.

2.3.1 Big data Stream processing tools, Technologies, solutions and frameworks

Apache Spark

Apache Spark is a multi-purpose open-source big data analytics framework originally developed at the University of California at Berkeley and later handed to the Apache Foundation to be open-sourced and contributed by the community, which has maintained it ever since. The original purpose of spark's design was to resolve some of the MapReduce model's limitations, particularly the need for quick processing of large datasets. By employing Resilient Distributed Datasets (RDDs), which are specifically designed to support in-memory computation, the MapReduce system, which runs computations on disk, was found to be more effective. Spark is a project that schedules, distributes, and monitors applications with multiple tasks across nodes in a computing cluster [40].

Spark, on the other hand, supports its native Spark cluster (standalone), Apache YARN, and Apache Mesos for cluster management. The RDD abstraction is also at the heart of the method. RDDs are collections of data objects that can be manipulated in parallel and are spread across cluster nodes. It supports several closely integrated modules for managing different forms of workloads such as SQL, streaming, and machine learning at a higher level.

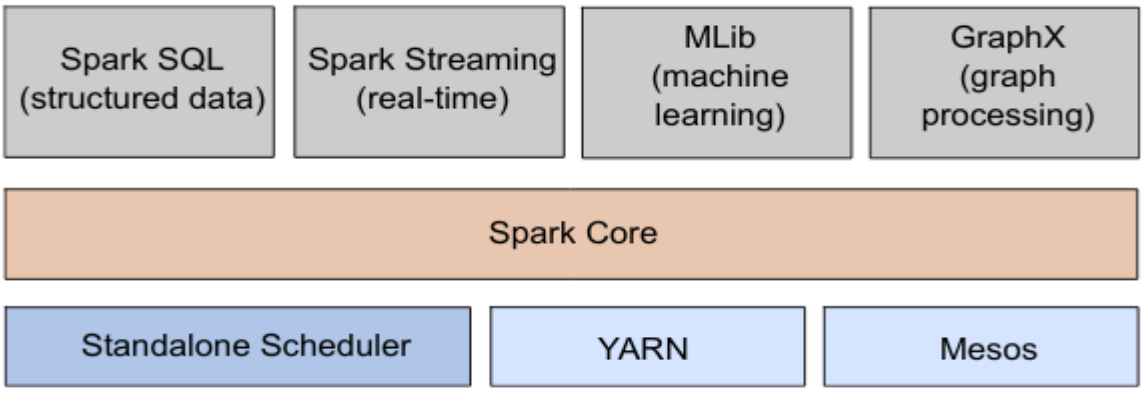


Figure 11 Apache Spark Architecture [77]

Since this research focuses on stream analytics, it focuses on modules used for stream Management (Spark streaming).

Spark Streaming is a micro-batch-based data stream processing system which Ingests data from a variety of sources and systems, including Apache Kafka, Apache Flume, and Amazon Kinesis. It is arguably difficult to achieve fault tolerance and manage stragglers using the conventional stream processing approach based on a graph of continuous operators that process tuples as they arrive (i.e. the dataflow model) [3] [40].

The discretized stream, or DStream, is a high-level abstraction used for Spark Streaming. DStreams employs a micro-batch approach, in which stream processing is organized as a series of batch computations carried out at regular intervals over short time windows. D-Streams stores the obtained data for a limited time period, which the cluster resources then use as an input dataset for parallel computations once the interval has passed. These computations result in the development of new datasets that represent a state in between or computation outputs. The RDDs that DStreams processes, as well as the datasets processed over the next interval, make up the intermediate state.

The Apache Spark Mllib module, on the other hand, is a library that contains common Machine Learning (ML) functionality such as learning algorithms for classification, regression, clustering, and collaborative filtering; and feature extraction, transformation, dimensionality reduction, and selection. MLlib also allows for the construction of machine learning pipelines as well as the storage of algorithms, models, and pipelines .Using Spark's core engine, these features are built to scale out across large computing clusters [2].

Apache Storm

Apache Storm is a distributed real-time big data processing and computation platform which is open source and fast. Clojure and Java are used to build it. Storm's cluster is similar to Hadoop, and it fills the gap between Hadoop and MapReduce, which provide a general framework for batch processing, and Storm, which provides a real-time stream processing framework [41].

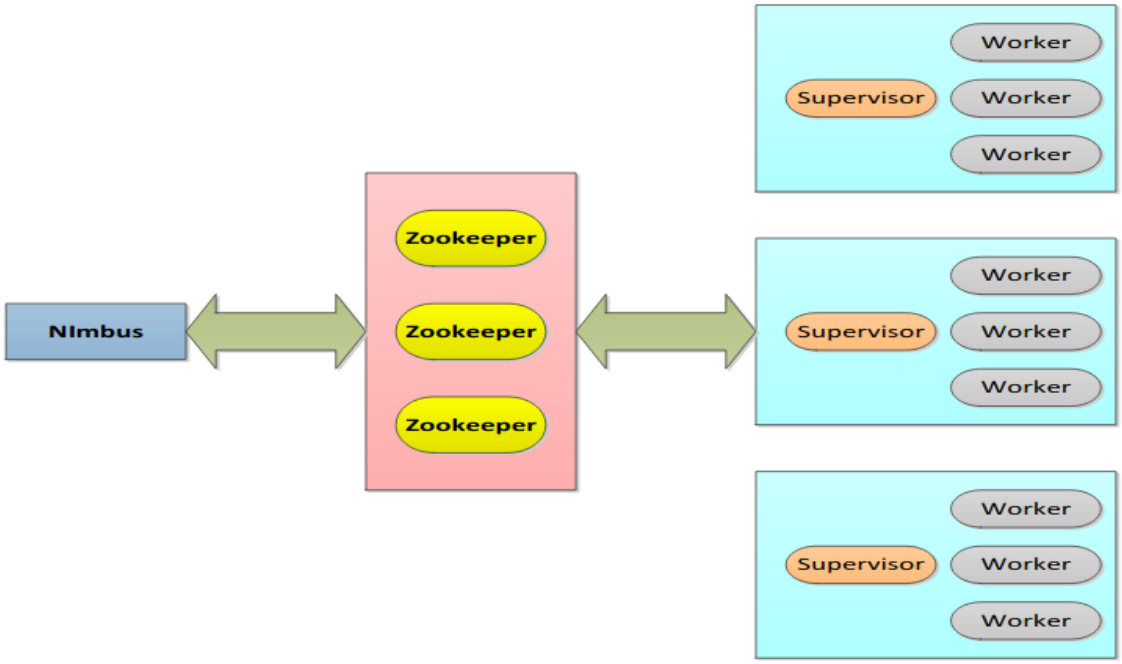


Figure 12 Apache Storm Architecture [41]

Storm is in favor of putting the dataflow model into use. Spouts and bolts refer to data sources and elements that process data in the form of tuples, respectively, and are used to define the topology of an application. Storm has sub-second latency since each tuple is processed when it arrives. Because of its acknowledge mechanism, its mechanism supports low throughput. Each record processed by an operator sends a confirmation to the previous operator, indicating that it has been processed [41]. This mechanism has the potential to incorrectly identify a large number of documents as non-acknowledged. As a result, these documents would have to be reprocessed, resulting in a decrease in throughput. It does not provide state recovery, but it does guarantee data transmission and processing through the use of upstream backups and record acknowledgements. In the event of a mistake (for example, a worker failure), the records are replayed by the spout if not all acknowledgements have been issued. There will be no data loss as a result of this method, however duplicate records can pass through the system. It's possible that mutable states are changed incorrectly twice. That is why it guarantees delivery at least once [2] [41].

Trident Storm provides micro-batch processing for increased throughput. The state in Storm Trident can be handled automatically, which ensures state continuity (exactly-once delivery), but the state is stored in a replicated database, which is costly since all changes are replicated across the network. Apache Storm is a compositional framework that requires the user to specify the application DAG directly [2] [41].

Apache Samza

Apache Samza is a Publish/Subscribe-based open-source stream-processing platform. It listens to a data stream, processes messages that are its stream primitives one by one as they arrive, and then outputs the results to another stream. It is closely integrated with the Apache Kafka messaging framework for data streaming between tasks and Apache YARN for task distribution across cluster nodes. Kafka provides replicated data storage that can be accessed with low latency, allowing Samza jobs to run with latency as low as a few milliseconds [22] [42].

Samza allows tasks to keep track of their current state by saving it on disk (typically using Kafka). To prevent performance issues, this state is stored on the same computer as the processing operation. Samza is able to achieve high throughput by combining storage and processing on the same computer. It also keeps track of whether a message has been sent and, in the event of failure, redelivers it to prevent data loss using a check pointing mechanism, although it can only guarantee delivery once. If a Samza task fails and is restarted, it is likely that certain messages consumed after the last checkpoint will be counted twice. The user's code determines the topology of a Samza job specifically [22] [42].

Apache Flink

Flink is a hybrid solution, and the need to handle multiple workloads within a single architecture led to the creation of many design trends, the most common of which is the "Lambda Architecture". The difficulty of combining batch and streaming architectures led to the "Kappa architectural" pattern, which fuses the batch and stream layers together. Flink is a manifestation of the Kappa design. Despite the fact that it uses a streaming execution model, two APIs running on the same distributed streaming execution can process both bounded and unbounded data [2] [43].

DataStream is the most common data abstraction for stream processing. It performs data-parallel and pipelined execution of arbitrary dataflow programs, resulting in low latency. The dataflow programming model in Apache Flink allows for event-at-a-time processing. Before being sent to the next operator to switch the knob between throughput and latency, multiples can be collected in buffers with an adjustable timeout. Based on current benchmarks, it performs well at a large scale, operating on thousands of nodes with excellent throughput and latency characteristics. It guarantees exactly once semantics by using state full computations [43].

A lightweight fault tolerance framework based on distributed checkpoints is also included in Apache Flink. Its algorithm creates consistent snapshots of the current state of the distributed system on a regular basis, without missing data or recording duplicates. These snapshots are saved

to a long-term storage unit. In the event of a loss, the most recent snapshot is restored, the stream source is rewound to the moment the snapshot was taken, and the stream is replayed. Flink is a one-of-a-kind solution in the processing platform world right now, but it's still a young project with little research into its scaling limitations. It's a declarative framework that gives users like Spark higher-level abstractions. The DAG is implied by the transformations' ordering, and the engine will reorder the transformations if necessary [2] [43].

Comparisons between Stream processing Frameworks

All streaming solutions examined in this section use a parallel distributed architecture that allows portioning of data streams and parallelization across a cluster of machines. The attributes or criteria that are also examined are the following:

- **Processing Model**: Batch processing, micro-batch processing, and stream processing tuple by tuple are all options for a system's processing model. In this part, we're not interested in batch processing systems like MapReduce.
- **Stream Primitive**: This usually Refers to the primary data structure in a streaming system. These systems use various words for such concepts.
- **Throughput**: Refers to the average number of jobs or tasks or operations per time unit.
- **State Management**: Stateless or stateful streaming computations are also possible. “A stateless program examines each individual event and generates a result based on the previous event.” For example, a streaming program might receive traffic data and send out a warning if a traffic light is broken. “A stateful program produces output based on a set of events.” Stateful state machines are used in complex event processing (CEP) in all types. A stateful computation is, for example, generating a warning after receiving two traffic light violations separated by less than 5 minutes. The structures under consideration use a variety of techniques to store states, or they may not store states at all.
- **Latency**: Refers to the elapsed time from job submission to receiving the first response.
- **Delivery Guarantee**: When comparing the results produced after a failure and a full recovery of the system to the results produced without any failures, this term refers to the "degree of correctness" of the results produced. The three semantics that define stream processing systems are as follows:
    - At-most-once delivery, which drops messages in case they are not processed correctly, or in case the processing unit fails. This is usually the least desirable outcome as messages may be lost.

- At-least-once delivery, which keeps track of whether each input was processed successfully within a specified timeout. This ensures that messages are redelivered and reprocessed in the event of a malfunction. No messages are lost if a mission fails, but certain messages can be redelivered. If a message's impact on state is idempotent, processing the same message several times causes no problems. A duplicate update would have no impact on the outcome. At-least-once delivery promises, on the other hand, will yield incorrect results for non-idempotent operations like counting. For certain use cases, this method is adequate, but it may result in duplicates
- Exactly-once semantics, the failure detection mechanism is the same as in the at-least-once mode. While messages are processed at least once, duplicates can be prevented using a variety of techniques. Such systems guarantee that the final outcome will be identical to that of a failure-free scenario. This is the most desired element, but it is difficult to ensure in all circumstances.

- **API languages:** Refers to the languages that someone can use in order to develop an application for this framework.
- **Programming Model:** Systems can be either compositional when users have to model the streaming application as graph explicitly or declarative when users are provided with higher level abstractions.
- **Contributors**: Refers to the respective community of contributors based on Github [44].

So if we compare Spark Streaming, Flink, Samza and Storm based on those criteria we get the following.

Platform/Criteria	Processing Framework	Stream Primitive	Latency	Throughput	Stateful Operations	Guarantee	Programming Model	API languages	Contributers (Github)
Storm	Streaming	Tuple	Subsecs	Low	No	At least once (exactly with Trident)	Compositional	Any	294
Spark Streaming	Micro-Batch	Dstream	Few Secs	High	Yes	Exactly once	Declarative	Java, Scala, R, Python	1287
Samza	Streaming	Message	Subsecs	High	Yes	At least once	Compositional	JVM languages	93
Flink	Hybrid	DataStream	Subsecs	High	Yes	Exactly once	Declarative	Java, Scala, Python	444

Figure 13 Comparison of analysis frameworks [2]

	Spark	Storm	Flink	Samza
Data format	DStream	Tuples	DataStream	Message
Data sources	HDFS, DBMS, and Kafka	Spoots	HDFS, DBMS, and Kafka	kafka
Programming model	Transformation and action	Bolts	Actions functions (map,groupby,..)	Mapreduce Job
Programming languages	Java, Scala and Python	Java	Java	java
Cluster manager	Hadoop YARN, Apache Mesos	Zookeeper	Hadoop YARN, Apache Mesos	YARN
Latency	Few seconds	Sub-second	Sub-second	Sub-second
Messaging	Exactly once	At least once	Exactly once	Exactly once
Machine learning compatibility	SparkMLLIB	Compatible with SAMOA API	FlinkML	Compatible with SAMOA API
Elasticity	Yes	Yes	No	No
Sliding win-dows/Windowing	time based	time based and count based	time based	time based and count based
Auto-parallelization	On demand	Pipelined processing	Pipelined processing	On demand
Streaming query	SparkSQL	No	No	Yes (Samza-SQL API)
Data Partitioning	Yes	No	No	Yes
API	Declaratif	Copositionnel	Declartaif	Copositionnel
Data transport	RPC	RPC	RPC	Kafka

Figure 14Summary of analysis frameworks [44]

Apache Metron

Apache Metron is an open source platform for security monitoring and analysis that incorporates a number of other open source big data technologies. As a result, Metron's capabilities can be divided into four categories: platform for capturing, storing, and normalizing any sort of data at extremely high rates, real-time processing and application of enrichments, effective information storage, and a user interface for viewing data and notifications that have passed through the device. Apache Kafka, Apache Storm, and Apache Hadoop are the three major open source resources at the heart of Metron. Metron is depicted in a high-level summary diagram below [45].

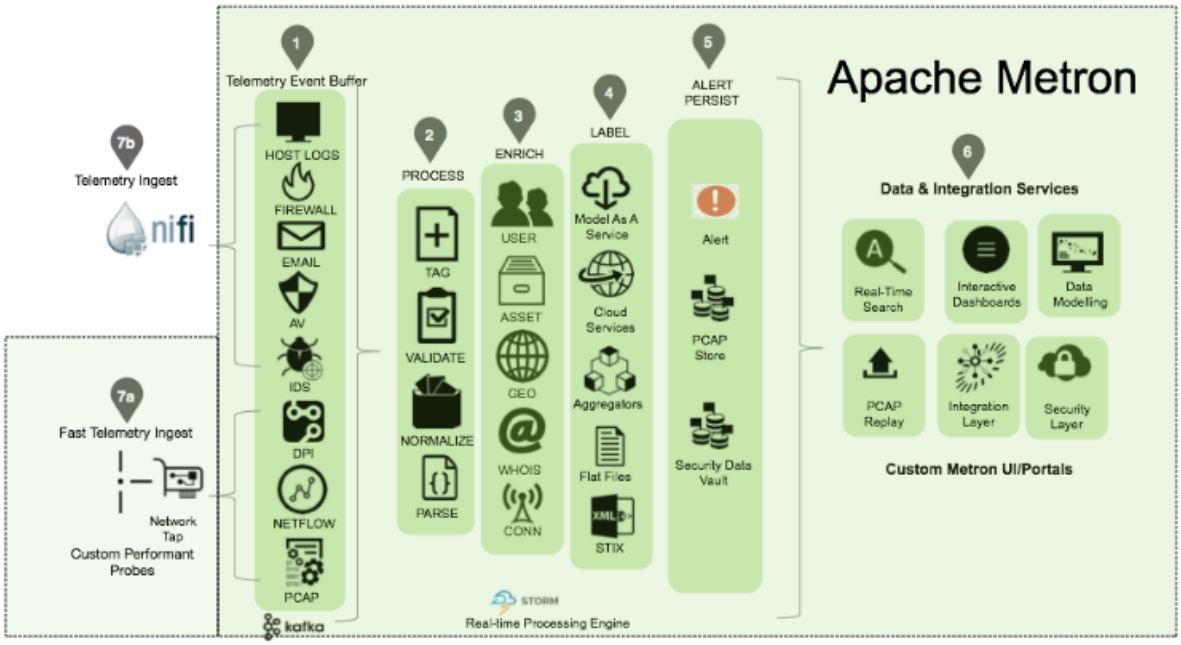


Figure 15Apache Metron Architecture [78]

Advantage and disadvantages of apache Metron

Advantages	**Disadvantages**
• Centralized Alerts Console • Fast telemetry ingest from various data sources such as Netflow, DPI,syslog,AV,Email etc • Easily customizable and interactive Kibana based Dashboard • modular and highly customizable design • Uses known fellow apache based open	• Too many Configurations and amalgamation of various technologies made is less user friendly • All or nothing:- failure of single framework such as Kafka, Storm, Hadoop usually interrupts the whole flow • Requires a vast amount of computing infrastructure

source frameworks such as Kafka,Storm,Nifi, Hive etc. • Real-time search + UI - Metron indexes all events and alerts and provides UI dashboard to perform real-time search. • Alerts labeled with threat Intel data - Viewing alerts labeled with threat Intel from third party feeds allows the analyst to decipher more quickly which alerts are legitimate vs false positives.	• Compatibility between the frame woks is highly sensitive to versions(i.e. upgrading one framework most likely will break the whole system) • Still in its infancy stage(as of the timing of writing of this paper it is on version 0.7 • Although the project is accelerated by vast contribution from the open source community, it's documentation is still inadequate for developers and researchers to work with.

Table 5Advantages and Disadvantages of Apache Metron

2.3.2 Log management tools, Technologies, Solutions and frameworks

ELK Stack

Elastic search, Log stash, and Kibana are three open-source products developed, managed, and maintained by Elastic Co. The ELK Stack is a series of three open-source products developed, managed, and maintained by Elastic Co. The Elastic Stack was renamed after the implementation and eventual inclusion of Beats data shippers, which turned the stack into a four-legged project. Based on the Apache Lucene search engine, Elastic search is an open source full-text search and review engine. Log stash is a log aggregator that gathers data from a variety of sources, applies various transformations and improvements to it, and then sends it to one of the several supported output destinations. Kibana, on the other hand, provides a visualization layer that interacts and works on top of Elastic search engine, providing users with the ability to analyze and visualize the data [46].

These various components are most widely used for tracking, troubleshooting, and protecting IT environments when used together (though there are many more use cases for the ELK Stack such as business intelligence and web analytics).Log stash is in charge of data collection and processing, Elastic search is in charge of indexing and storing the data, and Kibana is in charge of querying and visualizing it [46] [47].

SIEM Technologies

SIEM (security information and event management) incorporates the previously separate product types of SIM (security information management) and SEM (security event management) (security

event management).SIEM technology analyzes security warnings created by network, hardware, and applications in real time. They transform log entries and security device events into actionable data using rules and statistical correlations. This data will aid security teams in detecting threats in real time, managing incident response, conducting forensic investigations into previous security incidents, and preparing enforcement audits [48].

The word SIEM was first coined by Gartner, a research and consulting firm, in 2005.SIEM evolved from the two types of systems that had previously existed: Security Information Management (SIM) and Security Event Management (SEM) systems. On the basis of their log lists, SIM systems offered long-term storage in a centralized archive, trend analysis, and automated reporting. SEM systems capture real-time events, interpret them in near real-time, submit alerts, and view information at an operator's console so that defensive steps can be taken more rapidly. As a result, SEM is more concerned with the present, while SIM is more concerned with the past. A combined SIEM system collects logs and other IS-related information for analysis. The key functions of these three systems can be summed up as follows:

- SIM – log collection, archiving, historical reporting, forensics;
- SEM – real-time reporting, log collection, normalization, correlation, aggregation;
- SIEM – log collection, normalization, correlation, aggregation, reporting.

Data forensics software and compliance reporting standards are both supported by logging event reports from different intranet sources. Normalization transforms log messages from a variety of systems into a standard data model, allowing organizations to connect and evaluate similar events despite the fact that their log source formats were initially different. Correlation binds logs and events from various systems or applications, making it easier to identify and respond to cyber-threats. By consolidating duplicate event information, aggregation reduces the amount of event data. In real-time tracking and long-term summaries, reporting presents the correlated, aggregated event data. The basic event selection and record preservation capabilities are shown at the bottom of the figure. This middle data is then used for tracking and correlation purposes. The top demonstrates how the analyzed data can be interpreted as security details or event-driven reports [48] [49].

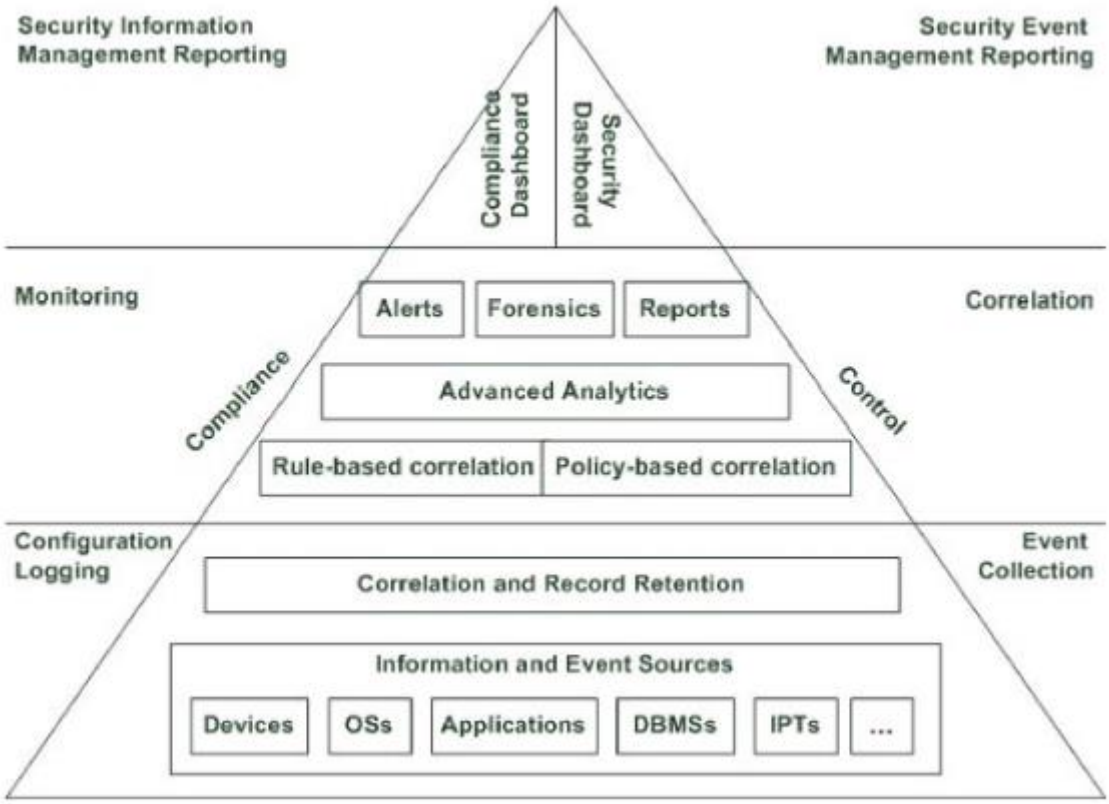


Figure 16 Typical SIEM system Architecture [48]

How does SIEM Technologies work?

SIEM software gathers log and event data from an organization's programs, security devices, and host systems and consolidates it into a single unified platform. SIEM collects data from antivirus incidents, firewall logs, and other sources, categorizing it as malware operation, failed and active logins, and so on. SIEM produces a warning and assigns a threat level based on predetermined rules when it detects a threat through network security monitoring. For example, someone trying to log into an account 15 times in 15 minutes could be considered as Ok, while 150 times in 10 minutes might be flagged as an attempted attack. In this way it detects threats and creates security alerts. SIEM's custom dashboards and event management system improves investigative efficiency and reduces time wasted on false-positives [49].

OSSIM

OSSIM is a SIEM technology that is open source. OSSIM is a C++ software library with advanced remote sensing, image processing, and geospatial capabilities at its heart. Ortho-rectification, precision terrain correction, rigorous sensor models, very large mosaics, and cross sensor fusions, a wide range of map forecasts and datums, and a large range of commercial and government data

formats are only a few of the features of OSSIM. The library's architecture supports parallel processing through MPI, a dynamic plugin architecture, and dynamically connectable artifacts, allowing for rapid prototyping of custom image processing chains [49].

The software distribution contains a wide range of command line utilities that can be conveniently scripted for batch production systems, as well as higher level GUI applications image linker and iview, which are designed around the heart of the OSSIM library. In addition, bindings for other languages have been made [49].

WAZUH

Wazuh is a threat detection, integrity monitoring, incident response, and compliance security monitoring solution that is free, open source, and enterprise-ready. It secures workloads on-premises, in virtualized, containerized, and cloud settings. The Wazuh system is used by millions of organizations worldwide, from small businesses to huge corporations, to gather, aggregate, index, and analyze security data, assisting organizations in detecting intrusions, threats, and behavioral anomalies. Wazuh is made up of an endpoint security agent that is installed on monitored systems and an intelligent management server that handles threat intelligence and data analysis. As cyber threats become increasingly sophisticated, real-time monitoring and security analysis are required to detect and remediate problems quickly. Our server component provides security intelligence and performs data analysis, while our lightweight agent provides the essential monitoring and response capabilities [50].

2.4 Related Works

Although there are numerous studies in the literature about the application of big data technologies for efficient management of logs of various kinds in the international community, there is no significant contribution to the area among the Ethiopian data enthusiasts. Most Literature in These studies proposed different Architectural and Technological solutions to ingest, store, process, visualize and query logs. They experimented with combinations of various open-source and proprietary big data technologies and evaluated their efficiency with metric such as performance, latency, Accuracy, time consumption, Speed, Bandwidth usage, scalability, integration, fault tolerance, timeliness, consistency, heterogeneity and incompleteness, load balancing, privacy issues etc. This section thoroughly reviews related research works on implementation of various big data analytics frameworks to analyze and mine live streaming log files.

Log analyzers are widely used in many web based applications and there are several) research work available on traditional static log analyzers. Few recently developed log analyzers are also became popular and adopted to overcome the drawbacks of static log analyzer. A research work performed

by Joshila et.al. [51] , Provides a detailed discussion about static web log analysis, Log types, log formats, access mechanism and their usefulness for web mining. The authors of the paper also discussed about creating an extended log file for user profiling. It proposed a hybrid web log analyzer using both IIS and tomcat server log. They proposed an ontology mapping approach to get required information. Data mining technique was also used to match user profiles and their interest.

The Paper gives a detailed look about the web log file, its contents, its types, its location etc., Added to these information it also gives a detailed description of how the file is being processed in the case of web usage mining process. The various mechanisms that perform each step in mining the log file is being discussed along with their disadvantages. The additional parameters that can be considered for Log file entries and the idea in creating the extended log file is also discussed briefly. The extended work is to combine the concept of learning the user's area of interest [51].

Agarwal & Prashad [52] proposed a framework for high speed streaming data analysis of web log streams that can resourcefully handle the challenging issues associated to manage multiple web based log streams that are distributed across a fleet of web based applications and present a summarized view of statistical profile of web based applications which may be useful for web usage mining. The proposed framework for high speed analysis of web access log streams is based on integration of Kafka and Spark streaming module. Apache Spark processing utilizes RDDs (Resilient Distributed Data) which are distributed collection of data and considered as a key abstraction provided by the Apache Spark. the proposed framework has various components assembled in way to facilitate the stream analytics task on sequence of high speed streams of web access logs. The streams of web logs are simulated from the web access log file to work as a live input streams from server [52].

Euginio etal in their paper [46] reported the implementation of ELK stack (Elastic Search Log stash and Kibana) framework to analyze streaming meteorological data logs from real life sensors. They used a data acquisition system (data logger) to collect and store data from a thermo-baro-hygrometer, and a pyranometer, which were calibrated previously in the laboratory. This paper aimed to analyze the open source Elastic search, Log stash and Kibana(ELK) stack to capture, transform, enrich, store, index, select relevant time slots and generate graphs that were integrated in a dashboard for combined visualization and analysis. Additionally, they explored its capacity to embed metadata from sensors and correct data based on a calibration certificate, also showing some relevant graphics. In this weather application, they observed that these sets of computational tools are well suited to manage the daily difficulties in handling meteorological data and metadata. The writers stipulate that, The open source Elastic search, Log stash and Kibana (ELK) stack has great potential to analyze historical and near real-time data: geo-identify the website users, real-time

analytics on Electric Vehicle (EV) mobility and behavior [6], collection and indexing of Tweets with a geographical focus, log management [46].

Another research work [47] tries to compare the Elastic's Elastic search Log stash Kibana(ELK) Stack, an open source search engine, analytics and Log Management Software (LMS)under Apache v2 license, and other Enterprise licensed software, mainly Splunk from high-level viewpoint. To present in details the real-world usability and best practice compliance of ELK Stack and how ELK was able to convert big companies to use their stack. Uncover the potential growth of development for ELK given the conversion of these company and possible contribution.

The writers conclude that If interested in trying or switching to ELK Stack be do reminded that it is a complicated LMS. They added that for the interested users to at least read the guides of the ELK Stack to have some knowledge about the software and on how to use it, or have an expert taught the administrator on how to use it. But It is very hard to conclude on whether to go with ELK or other enterprise LMS due to feature differences and implementation because the negative effect of plugins is that we need to set it up unlike in other LMS it is only one application but contains all the features needed. Not considering the organizational requirements and resource availability which includes but not limited to human resource and total cost of ownership.

Park et.al, in their paper [53] utilized a Korean web search engine named as NAVER for transaction logs. They proposed innovative methods of log cleaning, log sessions and classification of logs. . The goal of their research is to examine user queries in a variety of ways in order to improve search engine performance. His research looks at current transaction log analysis methods and provides a way for cleaning logs, defining sessions, and classifying queries. A term definition approach is also discussed, which is required for Korean transaction log analysis. Users behave in a straightforward fashion, according to the findings of this study: they write in short inquiries with a few query phrases, rarely use sophisticated features, and only look at a few results pages. Users are also passive in their behavior, rarely changing the search surroundings that the system has set for them. It's worth noting that consumers prefer to completely change their searches rather than simply adding or removing words to alter earlier inquiries. The findings of this study could help to improve the efficiency and effectiveness of Web search engines and services.

Wang et al. [53] presented a web log analysis method for the E Governance website. The authors developed a synthetic index that is made up of subjective and objective indices. A citizen-centric survey is used to create these indices. Evaluating the functioning of E-Government websites is beneficial.

Recent research work has been carried out by Madhury Mohandas on Hadoop Log Analysis Tools [54]. In the paper the need of activity monitoring in large scale computing has been highlighted. Several log analyzers have been discussed to perform assessment of failure detection and monitoring. Since log analysis supports optimization of the Performance of the system and resource utilization, the paper has summarized the recent developments that have been reported in the area related to log analysis and activity monitoring. It provides detailed scenarios of using selected technologies for log analysis such as vaidiya, Salsa & Mochi and chukwa.

The writers emphasized that Hadoop being the most popularly used methodology for storage and processing of Big Data, has several subprojects for failure monitoring and analysis. The majority of these tools seize the log files to capture the behavior of the cluster and the running application. They process the logs to retrieve the necessary information required for failure diagnosis, and some of the tools even support the failure recovery. An expository survey of some of these log analyzers shows that most tools try to capture only the application failure diagnosis aspect ignoring the hardware and network failures. In such clusters, the failures are not an exception and so diagnosis of failures must be extended to all possible levels of failures ranging from node failure to application failures [54].

Another work by Xia etal [55] which tries to design Online Visualization System for Streaming Log Data of Computing Clusters presents an integrated visualization system that employs a two-stage streaming process mode. Visualizing the log data of a computing cluster is a challenging problem due to the complexity of the underlying dataset: it is streaming, hierarchical, heterogeneous, and multi-sourced. Recently, visualizing heterogeneous streaming data has received much attention.

This paper focuses on the visual monitoring of computing cluster logs, “cluster log monitoring” for short. In High Performance Computing (HPC) cluster environments; although large volumes of performance log data are produced and consumed, there is a low ratio of valuable information content [55].

This paper presents an integrated visualization system that employs a two-stage streaming process mode. Prior to the visual display of the multi-sourced information, the data generated from the clusters is gathered, cleaned, and modeled within a data processor. The visualization supported by a visual computing processor consists of a set of multivariate and time variant visualization techniques, including time sequence chart, tree map, and parallel coordinates. Novel techniques to illustrate the time tendency and abnormal status are also introduced. We demonstrate the effectiveness and scalability of the proposed system framework on a commodity cloud-computing platform.

Liu [56]proposed a novel analysis method using web logs in genealogy system for drawing user profile and providing personalized recommendations. More specifically, firstly, all web logs are extracted from genealogy system based on the enhanced logs which record the behaviors and data of user. Secondly, the extracted logs are formatted for analysis, and we analyze the logs to extract various user-related attributes for drawing user profile. Finally, personalized recommendations like interested people and family tree are provided for each user in system. The high availability and effectiveness of our proposed method is verified on genealogy system-Huapu system.

A survey work by Bavaskar [28] provides information which give the knowledge about tools for log analysis like ELK Stack tool as well as different scenarios where to use log analysis. ELK stack is combination of Elastic search which accepts the raw data to form different indexes, Log stash which use to collect the logs and Kibana is used for visualization purpose. The paper explains that in the present world there are many tools that are used in different scenarios in collecting the logs and analyzing those logs to detect the malicious activity. There are many commercial tools to give the higher accuracy. Here, our main aim is to cover the survey on abnormality detection in log analysis.

#	Topic	Author	Year	Architecture & Technologies used	Type
1	ANALYSIS OF WEB LOGS AND WEB USER IN WEB MINING	L.K Joshila Grace, V.Maheswari, Dhinaharan Nagamalai ,	Jan 2011		Survey
2	High speed streaming data analysis of web generated log streams	Sonali Agarwal,Bakshi Rohit Prasad	Dec 2015	Lambda architecture Apache spark Apache kafka Apache Hadoop	Journal Article
3	Exploratory study of the ELK stack for meteorological observation system data analysis	Eugenio Almeida, Ivo Koga Márcio A. Santana Patricia Guimaraes	Jan 2017	ELK stack,	Journal Article
4	The Rise of Elastic	Michael Jade Mitra,	Nov, 2016	ELK stack, LMS	Journal

	Stack	Davvid Paulo Sy			Article
5	End user searching: A Web log analysis of NAVER, a Korean Web search engine	SoyeonPark Joon Ho Lee Hee Jin Bae	Apr, 2015	Custom	Journal Article
6	An Exploratory Survey of Hadoop Log Analysis Tools	Madhury Mohandas, Dhanya P M	Aug, 2013	Apache Hadoop Apache Chukwa Vaidiya Salsa & Mochi	Journal Article
7	An Online Visualization System for Streaming Log Data of Computing Clusters	Jing Xia Feiran Wu Fangzhou Guo Cong Xie Zhen Liu	Apr, 2013	Custom	Journal Article
8	A SURVEY ON: "LOG ANALYSIS WITH ELK STACK TOOL"	Pranita P. Bavaskar Onkar Kemker Aditya Kumar Sinha	Aug, 2020	Elastic Stack tools	Journal Article

Table 6 Summary of Related Works

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Overview

The research design and methodology chapter lays out the steps for gathering data and linking it (along with the conclusions drawn) to the study's initial questions. The primary goal of this chapter is to go over the research design and methods used in this study. It describes the phases of the research design such as the data collection techniques used in the research, data analysis, data storage and presentation, as well as evaluation techniques.

3.2. Research Design

Design Science Research (DSR) is used in this research. In Information Systems (IS) research, Design Science Research (DSR) aims to create new generalizable knowledge about design processes, design products, and designed artifacts while also solving organizational challenges through new work practices focused on Information Technology (IT) [57].

Design science research is a valuable paradigm for performing applicable, yet rigorous, research. It is ideal for studies that seek to produce artifacts at the end, such as prototypes, algorithms, models and instantiations. This paradigm was chosen because it offers a workflow for achieving the research's goal of developing a layered high-end technology stack data management model [58].

Design science is, in reality, becoming a part of mainstream science. Since the concept of design is fundamental to the field of information systems, this should come naturally. One of the most common modes of valid information generation in the IS discipline is the creation of new artifacts that push the boundaries of possibility [59].

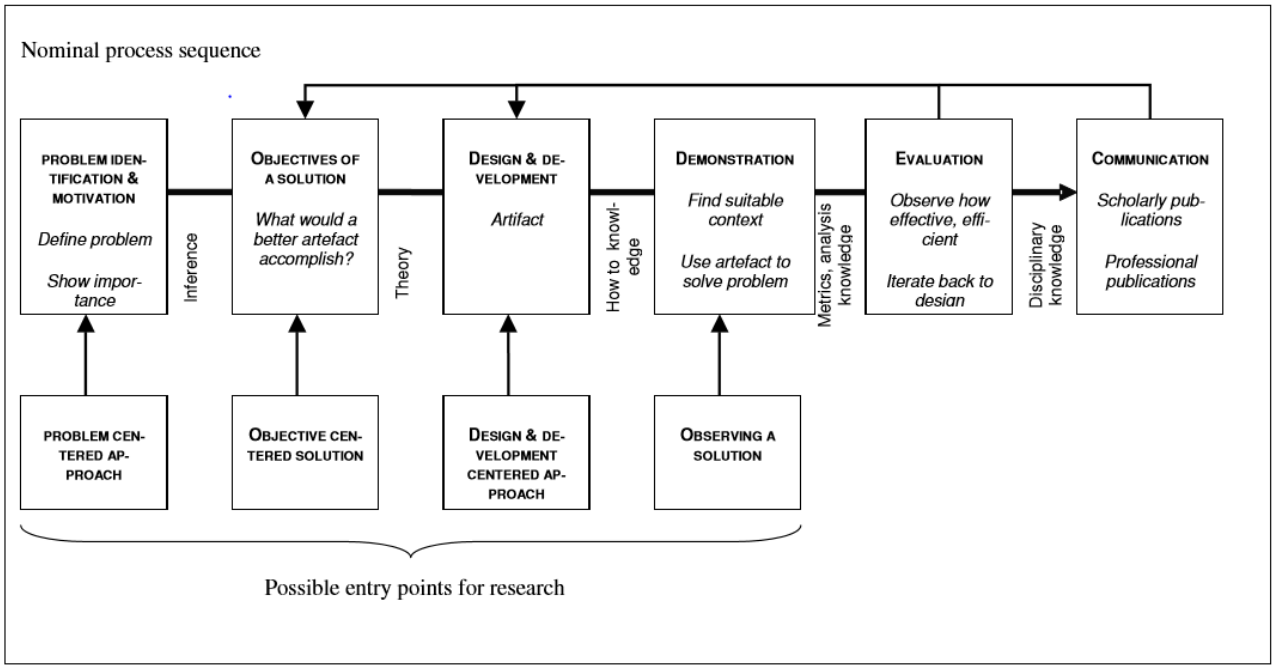


Figure 17 DSRP Model [58]

While it is widely accepted that design science research produces design artifacts, the processes by which these artifacts are created differ across studies. Peffers et al. (2006), for example, created a six-step method for conducting design science research. Problem identification and motivation, objectives of the solution, design and development, demonstration, evaluation, and communication are the steps involved [58].

On the other hand, [60] had proposed another process for design science research. It is described in the figure below.

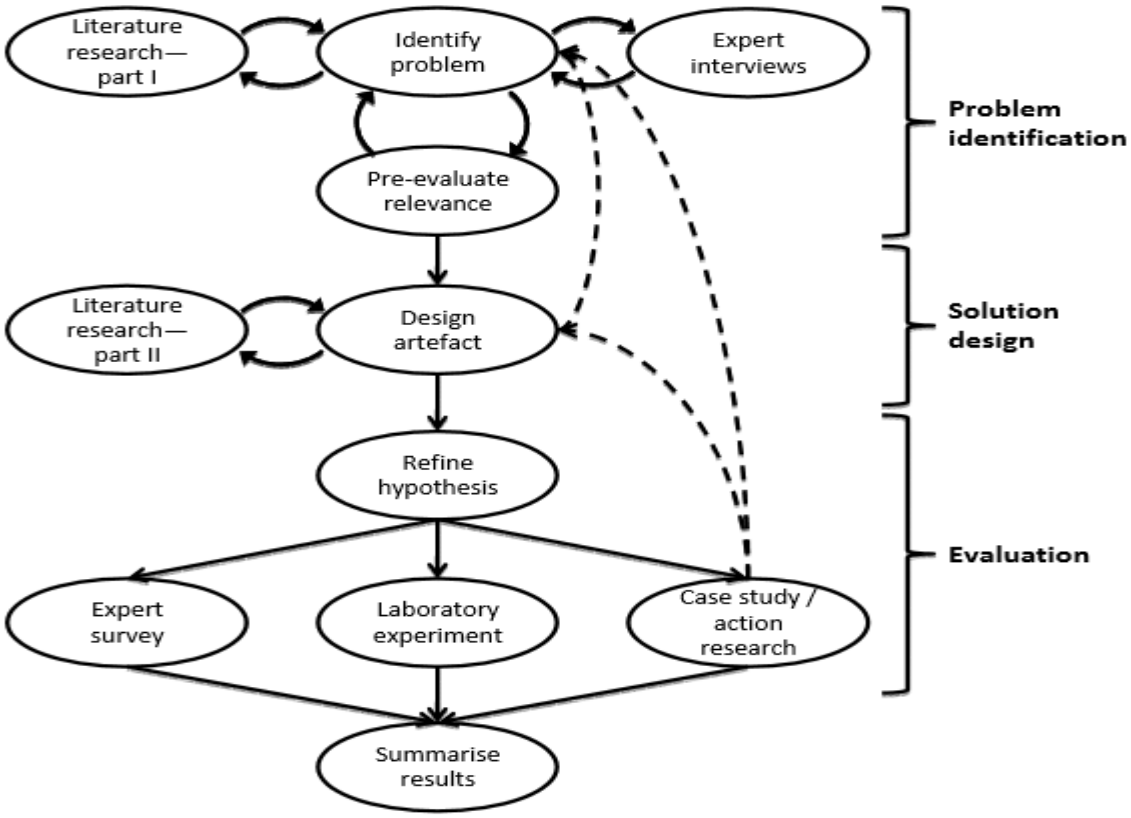


Figure 18 Proposed DSR process [60]

[61] , Also proposed alternative DSR process consisting of awareness of the problem, suggestion, development, evaluation and conclusion.

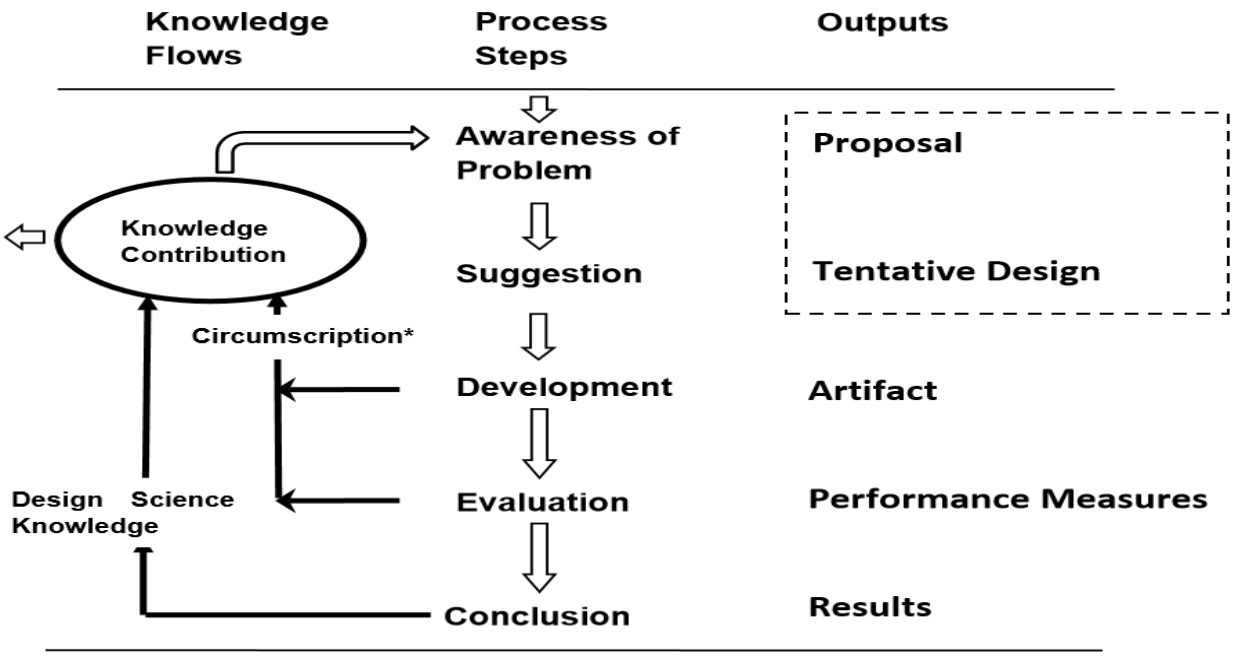


Figure 19 The DSR cycle [61]

In this research, the researchers chose to follow the methodology proposed in [61] mainly because it is the most widely used design science process model for this kind of research [59]and easily fits the problem under study. The other reason for selection is the fact that it is composed of simple and precise steps with clearly mentioned deliverables. [62] Last but not least, what makes this model different from the rest of design process models is the fact that contribution of new (and true) knowledge needs to be a key focus of design science research. The research process model shown above can be interpreted as an elaboration of both the Knowledge Using Process and the Knowledge Building Process arrows. The following are the phases of the selected DSR process model.

3.2.1. Awareness of the problem

Nowadays, organizations are overwhelmed by external data sources on the top of their internal data source called dark data. The challenges that organizations are facing is multifaceted and it is already testing their abilities to match their data processing capability against the incoming tsunami of data hence demanding a best-fit paradigm of using high-end Big data technologies and underlying Architectural studies [14].

Growing amounts of data originate from various sources that are not organized or straightforward, including data from machines or sensors and massive public and private data sources. Previously, most businesses were unable to capture or store large amounts of data. Existing processing tools are also incapable of delivering entire results in a timely manner. The adoption of new big data technologies, on the other hand, has aided in performance enhancement, product creation, and service and decision-making support. According to researchers, there are three main reasons to use big data. These include lowering hardware costs, determining the value of big data before investing substantial company resources, and lowering processing costs. [16] [17].

Despite abundance of logging mechanisms throughout the IT arena, not enough is done to utilize the contents of logs for better decision making. The effective balancing of a finite quantity of log management resources with a continuous supply of log data is a fundamental problem with log management that many organizations face. A vast number of log sources, inconsistent log content, formats, and timestamps among sources, and growing huge volumes of log data can all make log production and storage more difficult. [23].

Streaming big data analytics is the latest weapon in combating the difficulty of capturing, analyzing and managing high velocity data across industries. Since big data usually is about designing and deploying complete Stack of Technologies to manage each and every cycle of a Data life cycle Such as collection or Ingestion, Pre-processing, Analysis, Storage, Reporting and visualization, the need to review existing literature and technological trends and best practices is imminent for this research. The goal of those reviews is to design and demonstrate a log management layered

architecture of big data technologies with high Scalability, Resiliency, Performance, Accuracy and Fault Tolerance rates.

The designed architecture aims at collecting various standard log formats from remote systems in real time into a centralized system which is then piped into a series of processes such as data pre-processing(parsing, enriching, formatting etc.), storage, indexing and Visualization. In order to do that a thorough analysis of existing architectural and technological trends are reviewed in great detail.

As part of the problem awareness process step, the researchers had identified some core problems regarding difficulties of log management and monitoring to subsequently propose and demonstrate a solution architectural design of complete big-data technology stack tools. In order to understand the problem in hand, the researchers had prepared a research proposal and literature review documentations. The proposal formulates the problem in hand whereas the literature review document asses conceptual and design trends, related works and other relevant topics regarding the research. For further details, one can refer Chapter 1 and 2 of this Thesis report document.

3.2.2 Suggestion

The process of suggestion comes right after the development of a proposal based on an awareness of a problem. Suggestion is a creative procedure that requires imagining new features based on a novel combination of existing or new and existing elements [61].

Big Data is an area that focuses on the problems that available data poses, with the goal of producing actionable results based on the data. The number of technical options covering the Big Data area has exploded, inspired by the variety of inputs and the multiplicity of possible outcomes. Following the trajectory of developed technologies as they solve various Big Data problems is one way to explore this vacuum. Four distinct trajectories have developed, each of which can be recognized as a trend: (1) batch processing, (2) query processing, (3) low-latency query processing, (4) Continuous processing. These mentioned trends are mere categories of technologies embodied by tools with similar purpose. This paper focuses and further investigates technology trends that are primarily used for continuous processing.

Unlike Batch processing applications, for real-time applications there is a need to process unbound Streams of data and convert it into desired form. Projects such as Apache Storm, Apache S4, Apache Spark and Apache Samza, or managed platform solutions such as AWS Kinesis, address this challenge by providing programmable interfaces for operating on countless tuples or events. Applications like those allow one to define operations on tuples that transform incoming data as

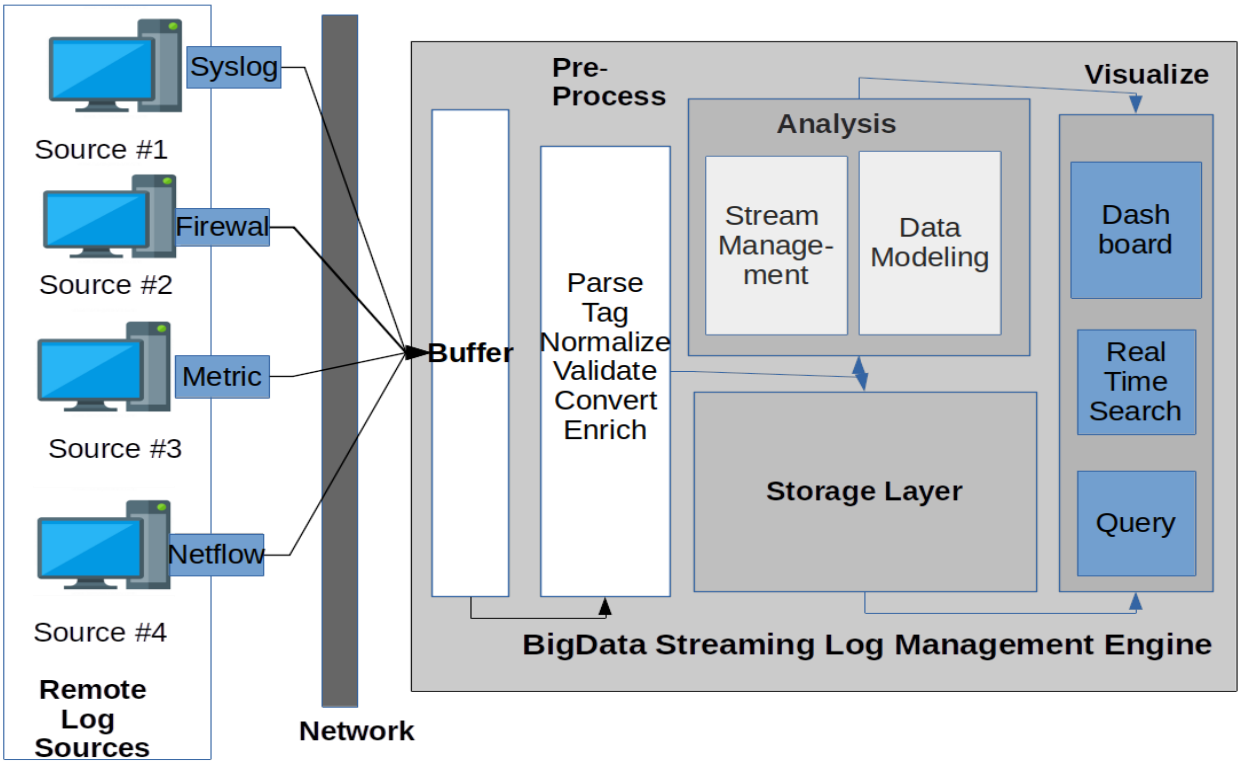


Figure 20 Proposed Logical Architecture of system

As needed (e.g., transform raw sensor data into a precursor for emergency situations or stock market data into trading trends). Each transformation can be composed into a topology, which becomes an execution workflow capable of addressing arbitrarily complex transformations [63].

The purpose of the suggestion phase in this research was to make that log management is achievable using big data streaming technologies and consequently to show the steps and procedures that are necessary. The suggestion phase is composed of the following steps and procedures.

The first step of data life cycle management is data Acquisition. Data can be produced from a variety of sources, such as data acquisition points, data entry points, signal capture and processing, computers, and sensors installed in industries, and so on [64].

This process comprises gathering raw data of various types and converting and modifying it in order to organize it. Cleaning data in real time cuts down on computation time and memory usage. This level of data quality is required since it allows for the overall optimization of the data processing circuit, which can be highly costly, especially in the context of big data [65].

Logs come in a variety of types and formats, and the absence of a common log format is a big drawback for log monitoring systems. Syslog, HTTP, and XML-like formats are some of the most well-known formats. After defining its log management requirements and goals, an organization can

prioritize certain requirements and goals based on the organization's perceived risk mitigation and the estimated time and resources required to execute log management functions.

Among handful of Log types and formats the researchers tend to experiment with 4 types of log formats. The first one is syslog. It is a standard system log writing formats for Unix based systems and one of the most abundant system logging mechanisms. System level logs are usually used to monitor and log events such as applications accessed, open and close activities, transactions made on and other system defined specific actions. System logs, such as Windows Event Logs or Linux system logs, are an important resource for computer system management. These logs hold messages produced from various sources in the computer system during its day-to-day operation. The messages could be informational or they can indicate a serious or trivial problem in the system [66].

Firewall Log is the second data source that is selected by the researchers. Firewall is one of the most abundant network devices which has a critical role for security and overall well-being of a network in any given organization. Firewall logs hold detailed data on how the firewall deals with traffic types. The main reason to monitor firewall logs is to see whether firewall rules work well or to debug them if they do not work properly. A network segment firewall located in the office premises of the researchers is used for this specific purpose.

Metric log is the third type of log the researchers considered. Nowadays, Servers such as Web servers, Database Servers, ERP hosting servers and Cloud Servers are backbones of day-to-day activities in most organizations. The failure or malfunction of those systems, even for short period of time, could cost the Organization substantial amount of financial and other kinds of damage. There for, monitoring the well-being of those systems is critical. Metric logs contain information on how the system is utilizing Disk space, Primary Memory, Network bandwidth, Processing Power, user activities, system Load and many more. A Private cloud server located in office premises of the researchers is used for this specific purpose.

Netflow Data, on the other hand, is used to analyze and monitor the flow of network in a network device such as routers, bridges, brouters and even system servers. Analysis of Netflow logs yields to determine things such as the source and destination of traffic, class of service, and the causes of congestion. A Network segment router located in office premises of the researchers is used for this specific purpose.

The above mentioned log files contents will be redirected to the Streaming log analytics engine in real-time. An agent program will be installed on each of the data sources systems and will be

configured to listen for new log entries in the target log file. When a new entry to the log file is detected, the agent will copy the new entry and transmit them to the unified log analytics engine through network.

The next step is to Buffer incoming Log traffic using suitable "Publish-subscribe" type buffering Technology. When Logs arrive to the analytics engine through the network, a buffering Technology will handle the incoming data until it is consumed by the analysis engine. The main reason to use buffering tool instead of directly piping data to the analysis engine is that, sometimes enormous logs might arrive at the same time and overwhelm the engine. When that happens the engine cannot handle the influx of data and some data might get lost. The buffering Technology will prevent this from happening by providing the engine with adequate supply of data and queue the rest until the engine requests more.

The pre-processing step will then parse the logs according to their formats, Normalize inconsistencies, Validate and Tag, Enrich with Host and GeoIP data and convert to a suitable data representation format such as XML and JSON.

The next step is to analyze the data with the goal of discovering useful information, informing conclusions, and supporting decision-making process. This phase is divided into two sub processes. The first is stream management. As mentioned in the chapter two, there are different ways to manage streaming data. Since Apache spark is the technology used for analysis phase, Micro-Batch stream management is going to be used. The main tasks needed could be initiate stream from the buffering tool, connect to the buffering tool, read data from the buffering tool, create discretized stream (D-stream) from the data and push the data to further processing. The second process is data modeling. It is the process of applying various statistical and descriptive analyses on the streamed data. Tasks such as counting, reduction, regression, correlation and other descriptive statistics and exploratory data analysis techniques will be applied. The output of this phase will be directed into two different paths. Depending on the nature of the analysis, it could be directed to the indexing phase or it could be stored into the storage framework.

The next step is to store the data for future references. Some analysis such as correlations and window operations need historical data to compare and correlate new arriving data with existing ones. A technology with low read-write latency and high throughput is a mandatory for this purpose. Nosql databases such as Hbase, MongoDB, Couchbase and Redis could be a viable option.

On the other hand, Hadoop ecosystem based data warehousing tools such as Apache hive is also frequently used by many researchers.

The indexing step will pull the analyzed data from the analysis engine and provides a distributed, multitenant-capable full-text search engine for schema-free JSON documents. The visualization step heavily relies on the efficiency of the indexing subsystem. Lucene is the standard search library used by most indexing systems nowadays. Search engines such as Elastic search and Solr use the lucene library to index and search text document entries in real-time.

Eventually the indexed documents will be ready for visualization and reporting. An interactive dashboard with bar charts, pie charts, GeoIP maps, hit maps, Meters and tabular representations will provide the user with both real-time and query based information.
A detailed explanation to each step is provided in chapter four of this thesis document.

3.2.3. Development

According to [61] the Tentative Design from the suggestion step will be further developed and implemented in this development phase. There are many forms of artifacts that can be developed that range from design theories to concepts, models, processes, or instantiations. And also the nature of the artifact to be developed usually determines the specific technique to be used for developing these artifacts.
The suggested streaming big data analytics system which is presented in section 3.2.2 above is now seen from the implementation perspective. The integration of selected big data framework stacks in each layer of the design is the main consideration in this phase. The main big data technologies that are used in the development of this system are Beats data shippers, Apache Kafka, Apache Spark, Elastic search and Kibana for collection, buffering management, process, index and visualize the logs in real-time. The development process can be summarized as follows.

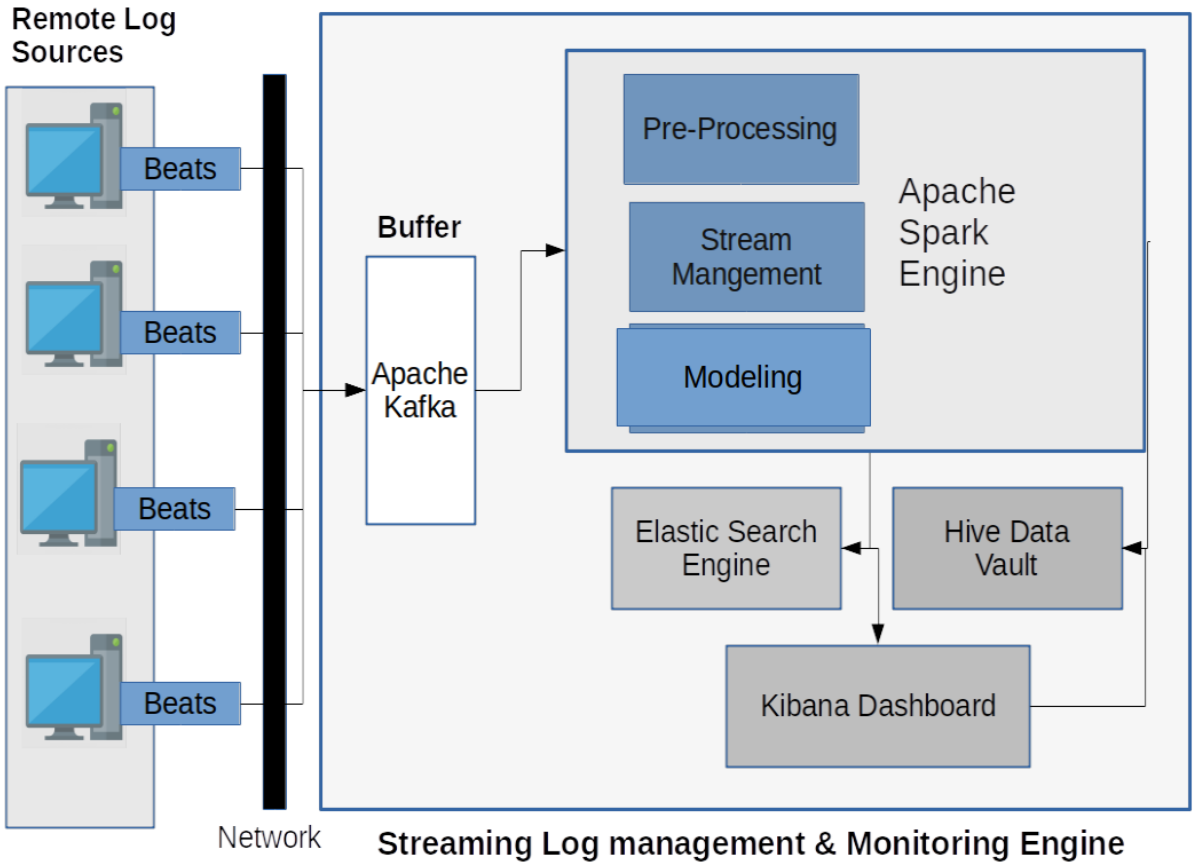


Figure 21 Proposed Technological architecture

As starting point to conduct the development of streaming log management & monitoring System, the following step by step approach is performed. First, a Cluster of 4 nodes will be prepared with specification that will handle huge amount of stream data and internal workings of all frameworks that are explained above. Beats data shipper library is installed and configured on all data sources that are explained in section 3.2.2.Every time a new log is written into the target data source, the respective beats data shipper will copy the contents of the log and send them to the Central Log management engine.

The central Stream log management engine will entertain integrated data pipeline consisting of Apache Kafka, Spark Streaming, Elastic Search Indexing and Kibana Visualization Tool. Apache Kafka will be used as a reliable and fault tolerant interface to upcoming remote logs so as Spark streaming system work with absolute confidence that data is not lost due to delays occurred for whatever reason. Apache spark will do all the Analysis on incoming data such as parsing, Tagging, Converting, Normalizing and Modeling of data. Then it will push the Analyzed data to respective

Elastic Search Indexes. Finally, Kibana Dashboards will visualize and query data stored in Elastic search. Important logs are also dumped to Apache hive data store occasionally.

A more elaborate explanation of the development process and integration of the technology stacks are presented in chapter 4 of this document.

3.2.4. Evaluation

It is always been not easy to evaluate a full cycle big data analytics platform because it consists of numerous Frameworks and Technologies integrated to form a system. But Research works have used different metric to evaluate the quality and efficiency of a big data Platform.

The general system can be described in factors such as Performance, Scalability, Fault tolerance and Efficiency where as individual components can be evaluated using specific metrics that depends on the purpose of the component. For example Beats data shippers can be evaluated by Time consumed to forward log to central system, memory usage, number of bytes read/ sec and number of bytes written/sec. Whereas Apache Kibana tend to incline to metrics such as refresh rate, resolution and color support, adaptability and memory consumption. Apache Spark also can be evaluated using memory consumption, Bytes processed/ sec, elapsed System and User time/ stream and write efficiency to the indexing system.

Evaluation of the system against similar systems in various research works is the other evaluation mechanism the researchers have used. Chapter 4 of this document presents a more detailed explanation of the evaluation metrics.

CHAPTER FOUR

STREAMING BIG DATA FOR UNIFIED LOG MANAGEMENT AND MONITORING SYSTEM

4.1. Overview

This chapter details the methods and techniques followed throughout the design and development of a Streaming big data analytics based Log management and monitoring system, as well as the outcomes of each procedure. As discussed in Chapter three (methodology) sections 3.2 of this document, a DSR process is followed. The chapter begins with a discussion of the suggestion phase, in which the researchers' solution architecture is presented in detail. After that, the development sub-section of this chapter further elaborates the overall procedures and experimental setup followed to build the log management and monitoring model. Finally, the prototype system's overall result is discussed.

4.2. Suggestion

4.2.1. Designing the Prototype system

Big data life cycle usually follows the path starting from business understanding followed by data acquisition, data cleaning, data storage, data analytics and indexing finally to visualization and reporting. Designing an efficient layered architecture is recently a hot agenda in the data science community. Since this research deals with a complete Stack of Technologies to manage each and every cycle of a Data life cycle, a step by Step explanation of each and every layer is an absolute necessity. This section will start by explaining the general architectural aspects of the solution platform followed by discussion of each component on the life cycle in a greater depth.

The first step of data life cycle management is identifying data sources and types to be used in the process. Data can be produced from a variety of sources, such as data acquisition points, data entry points, signal capture and processing, computers, and sensors installed in industries, and so on. As presented in the architecture diagram in chapter three, the main sources of data come from remote hosts in the form of real-time logs [64].

Data Types and Formats

Big data is a type of diversified data that cannot be compelled to conform to a specific format or to conform to an organization's standards and practices. Furthermore, in a big data scenario, data has a finite amount of value that can be extracted for decision-making or intervention. To be most useful, data must be linked to other data sets in order to provide reliable insight [67].

Logs come in a variety of formats and types, and the absence of a common log format is a big challenge for log monitoring systems. Syslog, HTTP, and XML-like formats are some of the most well-known formats. After defining its log management requirements and goals, an organization can prioritize certain requirements and goals based on the organization's perceived risk mitigation and the estimated time and resources required to execute log management functions [66].

User: User level audit trails are usually used to monitor and administer all commands directly initiated by the user, all identification and authentication attempts, and files and resources accessed. Those kinds of logs are mostly very useful to monitor authorization and authentication of users across systems.
Application/Database: Application/database logs are usually used to monitor and log user activities, including data files opened and closed, certain actions defined in this procedure, and printing reports. Databases usually store sensitive and personal information and they are often backbones to Web based systems in which automated queries will make CRUD operations which open a door for various access violations and attacks of which SQL injection is the most common [66].

System: System level logs are usually used to monitor and log events such as applications accessed, open and close activities, transactions made on and other system defined specific actions. System logs, such as Windows Event Logs or Linux system logs, are an important resource for computer system management. These logs hold messages produced from various sources in the computer system during its day-to-day operation. Emitted messages may be informational, or they can indicate a problem in the system, whether trivial or more serious [66].

Network Usage: Network level audit trails generally monitor what are operating, unauthorized access attempts, and vulnerabilities. A Network Log file usually is full of useful information if carefully utilized will provide indicative information such as network usage statistics, IP‘s producing excessive traffic, detect unusual and possibly fraudulent behavior and many more [69].

A set of data shippers collectively known as Beats which are designed to ship various Data types such as File logs, Network usage, Database/Application Logs, Metric and Heartbeat information from source hosts to the Analysis engine are used to collect log data from remote hosts. Beats data shippers are small application daemons that are installed on target data source systems and configured to ship various logs to remote hosts in real-time [68].

File beat, metric beat, Audit beat and packet beat are the 4 beat log shippers used in this experiment.

File beat is the most widely used beat and is used for collecting and shipping log data, as its name suggests. File beat is available as a Docker container and can be deployed on almost any operating system. It also includes internal modules for Apache, MySQL, Docker, MariaDB, Percona, Kafka, and other platforms. For these platforms, there are default configurations and Kibana pieces. As of Version 7.0.0, File beat also supports ingesting the additional log types such as audit logs, server logs, sluggish logs, and deprecation log [69].

Packet beat, on the other hand, was the first beat implemented, and it was a network packet analyzer. Packet beat records network traffic between servers and can be used to monitor application and output. Packet beat may be mounted on the monitored server or on a separate dedicated server. Packet beat monitors network traffic, decodes protocols, and keeps track of data for each transaction. DNS, HTTP, ICMP, Redis, MySQL, MongoDB, Cassandra, and several other protocols are provided by Packet beat [68].

The third one is Metric beat. It is a widely used beat that gathers and records system-level metrics for a variety of systems and platforms. Internal modules for gathering statistics from specific platforms are also supported by Metric beat. Using these modules and sub-settings known as metric sets, one can customize the frequency with which Metric beat collects metrics as well as the particular metrics to collect [68].

Audit beat can be used to track the operation of users and processes on your Linux servers. Audit beat, like other standard system auditing tools (systemd, auditd), can be used to find security flaws such as file changes, configuration changes, malicious actions, and so on [68].

Data Ingestion and Buffering

The next phase is to prepare a reliable ingestion and buffering layer for incoming high velocity data. The main task of this layer is to temporarily store incoming traffic of logs and smooth feeding it to the processing engine so that the engine is not overwhelmed by extreme tsunami of log data. Apache Kafka, Apache flume, Scribe are the most common data ingestion and buffering technologies used by the Hadoop ecosystem community.

Apache Kafka is a real-time framework that uses a distributed public-subscribed messaging framework which can manage a huge amount of data, allowing you to send messages at the end-point. LinkedIn created Apache Kafka, which is now available as an open source project through the Apache Software Foundation. It's a public-subscribed messaging system that's designed to be scalable, fast, dependable, and long-lasting.

Apache Kafka is the most developed and efficient framework for production level systems since it supports data replication and real-time querying of data. It is also flexible enough and gives more control to the data analyst or programmer. The other reason to single out Kafka is its neat integration with processing engines such as Apache Spark, Storm, Flink and Samza.

In this prototype system, Kafka is used to Store Live data in their respective topics which is sent from Remote Hosts using Beats data Shippers (i.e. Beats are the main Producers of data in this system). Apache Spark Streaming will then read the data from the Topics that it has subscribed for and hence play as a consumer for our Kafka Broker.

Data Pre-processing and Analysis

This process entails obtaining raw data of various types and performing the required conversions and modifications to organize them. Cleaning data in real time reduces calculation time and memory requirements. Data quality must be performed at this level because it allows for the overall optimization of the data processing circuit, which can be very expensive, particularly in the context of big data. A balance must be struck between the need for quick access to information and the need for high quality.

Logs have a lot of different types, e.g. sever logs, database logs, event logs etc. The contents of logs include numerical data and non-numerical data as well. For some kind of logs, like server logs, those numerical data (e.g. CPU load, memory) is sufficiently representative as features. But the meaningful non-numerical data (e.g. the indicators about the state of systems) are also interesting [75].

Since Logs comes in different formats and types, the incoming data needs to pass through a series of Data parsing, Transformation, Normalization and Enrichment processes before it is piped to the actual processing and modeling layer. A pyspark program is written in python programming language and submitted as a spark job using the “spark-submit” command. Various Regular expressions (regex) and code snippets written by various github developers are reviewed and considered to write the pre-processing module.

One of the first activities is Parsing incoming log(i.e. for example if incoming data is syslog format we need a syslog parser which will analyze the log and break to individual pieces such as time, User, Log type, Log content etc.).The next is to Tag the log so that it‘s sequence is known in case there comes a need to use windowed operation and also logs can be identified from other. Sometimes it is also a necessity to covert logs into appropriate log formats if the incoming data is different from the expected format (for example if Log came in XML format where JSON format is expected, we need an XML to JSON converter module to apply to the data).

Normalization is also a task to consider in this phase. As mentioned before, the format of our logs is not fixed. But they have some common characteristics, for example, each message in one log is started by a time stamp with a fixed format and other characteristics like the logs are written in English and composed by upper and lower case letters, digits and special characters. Inspired by Natural Language Processing, it might be helpful to do some text normalization to the raw logs before extracting features. This step can be omitted in some approaches. The following lists how our text normalization jobs.

The stream management and analysis is handled by Spark's Micro batch based Spark Streaming library whereas core Resilient Distributed Data set (RDD) abstraction handles the distribution of data and processing over multiple nodes. The Spark Streaming Library will performs task such as subscribing to Kafka topics, Initiate connection to Kafka topic, Read from Kafka Topic, Close connection to Topic. Spark core on the other will handle creation of RDDS, Manipulation of RDDs, managing RDDs and Destroying RDDS when they are no longer needed.

Apache spark then creates Data frames from the RDDs and perform required analysis and modeling techniques which are thoroughly discussed in 4.3.4. Apache Spark is also responsible to write the result data frames to elastic search index.

Indexing

Indexing task is handled by elastic search. Elastic search is a search engine built on Apache Lucene. It's a distributed, real-time full-text search and analytics engine that's open source. It uses schema-less JSON (JavaScript Object Notation) documents to store data and is accessible through a RESTful web service interface. Since it is based on the Java programming language, Elastic search can run on a variety of platforms. It allows users to quickly and efficiently explore large amounts of data.

Elastic Search is correlated with a number of primary principles. A single running instance of Elastic search is referred to as a node. Multiple nodes can be accommodated on a single physical and virtual server, depending on the capacities of their physical resources such as RAM, storage, and processing power. The term cluster, on the other hand, refers to a group of one or more nodes. Cluster offers collective indexing and search capabilities for the entire data across all nodes. The index is the third concept, which is a list of various types of documents and their properties. The definition of shards is often used by Index to improve efficiency [76].

In elastic search, a document is a set of fields organized in a specific way and specified in JSON format. Every document is assigned to a type and is stored in an index. A unique identifier known as

the UID is assigned to each text. Shard Indexes are subdivided horizontally into shards. This means that each shard has all of the document's properties but has less JSON objects than the index. Shard is an independent node that can be stored in any node due to the horizontal separation. The initial horizontal component of an index is called a primary shard, and these primary shards are then replicated into duplicate shards. Last but not least, Elastic search's Replicas feature enables users to make duplicates of their indexes and shards. Replication not only increases data availability in the event of a malfunction, but it also enhances search efficiency by running a parallel search process in these replicas [76].

Elastic search will store indexed Document entries which are usually split into shards for the purpose of Replication. Apache spark will be the main source of Production of indexed document and once documents are indexed and put in elastic cluster, they remain there for consumption by visualization tools such as Kibana.

Data Visualization, Reporting and Querying

The final step is to present the final output in various data visualization methods. An interactive, User friendly and efficient data visualization frameworks have joined the big data arena recently. Kibana is one of the dominant frameworks due to its easily customizable nature, support for querying and high refresh rates. It is an open source browser-based visualization tool that is primarily used to analyze large volumes of logs using line graphs, bar graphs, pie charts, heat maps, area maps, coordinate maps, gauge, targets, timelines, and other visualizations. The visualization makes it simple to forecast or see shifts in error patterns or other important input source events. Kibana is synchronized with Elastic search and Log stash, forming the so-called ELK stack. Kibana can access Elastic search logs and show them to the user in the form of line graphs, bar graphs, pie charts, and so on. It offers features such as Dashboards, Visualization, Dev Tools and Reporting Mechanisms, Filters and Search query, Coordinate and region maps, Timelion and Canvas features.

4.3. Development

4.3.1. Infrastructure setup

As mentioned in chapter 1, utilizing a distributed infrastructure is one of the three pillars of big data analytics. Various Distributed Infrastructure are been come into existence in the past 20 years or so. Cloud based systems, Grid computing, Cluster computing and Volunteer Computing are some of the common setups to mention some. The researchers tend to use a private cloud platform to setup a 4 node distributed computing infrastructure to deploy the designed streaming log management system. The cloud infrastructure is located in the office premises of the researchers and is not accessible to outside world due to company policies.

#	Node name	IP Address	Ram Size	Storage Size	Os installed	Flavor
1	node-1	10.0.35.239	32GB	500GB	Centos7-x86_64 1810	co_mo_small
2	node-2	10.0.35.240	80GB	500GB	Centos7-x86_64 1810	co_mo_large
3	node-3	10.0.35.241	32GB	500GB	Centos7-x86_64 1810	co_mo_small
4	node-4	10.0.35.242	80GB	500GB	Centos7-x86_64 1810	co_mo_large
Total =			**224GB**	**2TB**		

Table 7Cloud Nodes Specification

This Openstack Mirantis based cloud platform which hosts the above nodes is managed and monitored through its horizon dashboard as seen below.

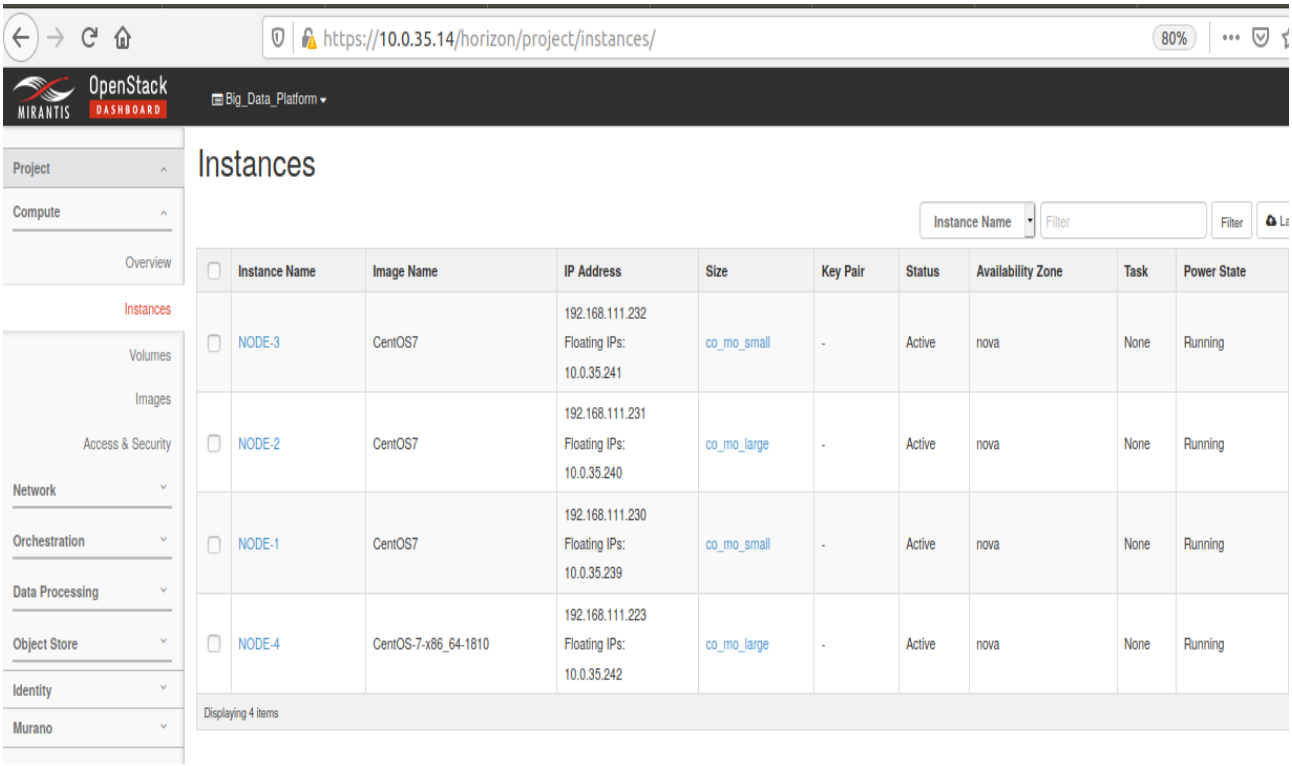


Figure 22 Horizon Open-stack Dashboard view of created nodes

All four nodes are installed with Linux operating system. Due to its stability, the researchers chose Centos 7-X86-64 operating system. Node 1 will host Elastic search Master node and Kibana Dashboard and it also serve as spark worker node. Node 2 is designed to be Spark Master node and Elastic search component node and Hive warehouse. Kafka Broker is also installed in this node. Node 3 and 4 are slave node for Apache spark and hive client systems. Kafka clients are also installed on those nodes.

#	Node name	Installed Services
1	Node-1	Elastic search master. Kibana Master
2	Node-2	Spark Master, Kafka Broker, Hive Master, Elastic Node
3	Node-3	Spark worker, Hive worker, Elastic Node, Kafka client
4	Node-4	Spark Worker, Hive Worker, Elastic Node, Kafka client

Table 8 Installed Services on each computing node

The nodes are also installed with the following software and libraries. Python2 and 3 are both installed on all systems. Although Apache version 3 is been released, the researchers chose to use Apache Spark 2.4 because of recurring Bug problems observed in version 3 series of Spark. Java Version 1.8 is also installed on all systems due to some spark and Kafka libraries that are programmed using Scala programming language require the underlying JVM to run. The four nodes are also need to communicate via a passwordless ssh. So Private and public key should be generated on one of the machines and then the public key should be securely copied to all the remaining nodes.

4.3.2 Data Acquisition

As stated in the previous section, a live data from different log types will be used for this experiment. The first setup that needs to be done is to install the beats data shippers on the target data sources. Packet beat, Metric beat, File beat and Audit beat are the four selected beat data shipper types which are used to ship netflow data, usage data, file log and audit log data types respectively. To install beats data shippers first you need to download the binary or source code of the program from the official site. The researchers used version 7.9.3 of the beat series.

For the netflow data, a Network segment gateway located in office premises of the researchers is used. This system is software defined network (SDN) based system that is used to connect the Segment of the office floor LAN to other office floor LANs. It runs on Centos7 Operating system on Huawei manufactured Server.

For syslog data, a Linux server located in the office premises of the researchers is selected and standard system log folder /var/log/ is set as the file log path for file beat to monitor and read incoming log in real-time.

For Metric beat installation, we used a Fedora based Server Computer located in office premises of the researchers. Metric logs usually contain information on how the system is utilizing Disk space, Primary Memory, Network bandwidth, Processing Power, User activities, system Load and many more.

For Audit log shipment, the above mentioned Linux server used for syslog data is re-used. Since file beat and audit beat work independently, installing them on a single system will not be a problem.

The standard procedure to install and configure beats data shippers is as follows. After a Tar ball of the binary program is downloaded to the target data source system, they will be copied to the home directory of the user. Then the tar ball will be extracted with “tar -xvf name_of_the_tarball” command which yields a folder with the contents of the corresponding beat shipper. The folder usually contains an executable program to run the beats daemon, a read me file, a “.yml” configuration file which is usually used to configure what the beats program will do. It also contains various folders with modules and libraries needed by the beats executable file. A sample configuration file for packet beat is shown below.

The main parameters to be set for packet beat are described in the table below.

#	Parameter name	Value	Description
1	Packetbeat.interfaces.device	eth0	A network interface card to use to record the logs from.
2	timeout	30s	A session time out duration
3	period	10s	Reporting period(a time duration)
4	Packetbeat.protocols	enabled=true	Enable individual protocols in this section
5	type-http	ports: [80, 8080, 8000, 5000, 8002]	Monitor these ports for http traffic and record the logs.
6	index.number_of_shards	2	How many shards to use for each data entry
7	output_kafka	hosts: ["node-2.novalocal:9092", "node-3.novalocal:9092", "node-4.novalocal:9092"]	Where to push the logs after reading
8	topic:	'%{[fields.log_topic]}'	Which topic to write log to.
9	required_acks:	2	How many acknowledgements are needed for each transfer of log.
10	compression:	gzip	What kind of compression tool to use to compress the logs while in route in network.
11	max_message_bytes:	10000000	Max number of bytes for each message

Table 9Packetbeat Configuration parameteres and values

After all the above parameters are correctly set, the next step is to start the packet beat daemon by running the packet beat executable found in the home folder of the extracted files.

Since the other three data sources use almost the same kind of configuration steps with little different parameters and values, lets skip the details of those data sources and head to configuring Kafka brokers and topics to store incoming data temporarily before Apache spark Streaming reads the contents out of the topics.

4.3.3 Data Ingestion & Buffering

Apache Kafka will ingest incoming logs sent by corresponding beats data shippers. Apache Kafka is a real-time framework that uses a distributed public-subscribed messaging framework which can manage a huge amount of data, allowing you to send messages at the end- point. LinkedIn created Apache Kafka, which is now available as an open source project through the Apache Software Foundation. It's a public-subscribed messaging system that's designed to be scalable, fast, dependable, and long-lasting. In order to configure Apache spark correctly, one need to understand the overall architecture and inner workings of the framework.

It consists of the following components

1. Topic: All Kafka messages are grouped into topics, which is a feeding mechanism through which messages are stored and written. You read a message if you want to read it, and you send a message to a particular subject if you want to send it. User applications read data from topics, while producer applications write data to them. A Kafka Topic that has been partitioned into multiple sections.
2. Producers: Messages to one or more Kafka topics are published by producers. Kafka brokers receive data from producers. Whenever a supplier sends a message to a broker, Producers can also send messages to a specific partition.
3. Consumers: It read data from brokers. Consumers subscribe to one or more topics and consume published messages by pulling data from the brokers
4. Connectors: It's in charge of getting stream data from Producers and getting it to Consumers or Stream Processors.
5. Stream processor: Stream Processors are Kafka Cluster applications that convert topic data streams into other topic data streams.
6. Broker: To maintain load balance, a Kafka cluster usually consists of several brokers. Since Kafka brokers are stateless, they depend on Zookeeper to keep track of their cluster state.
7. Zookeeper: Zookeeper is a tool for organizing and controlling Kafka brokers. Zookeeper service is primarily used to alert producers and consumers when a new broker enters the Kafka system or when a broker in the Kafka system fails [47].

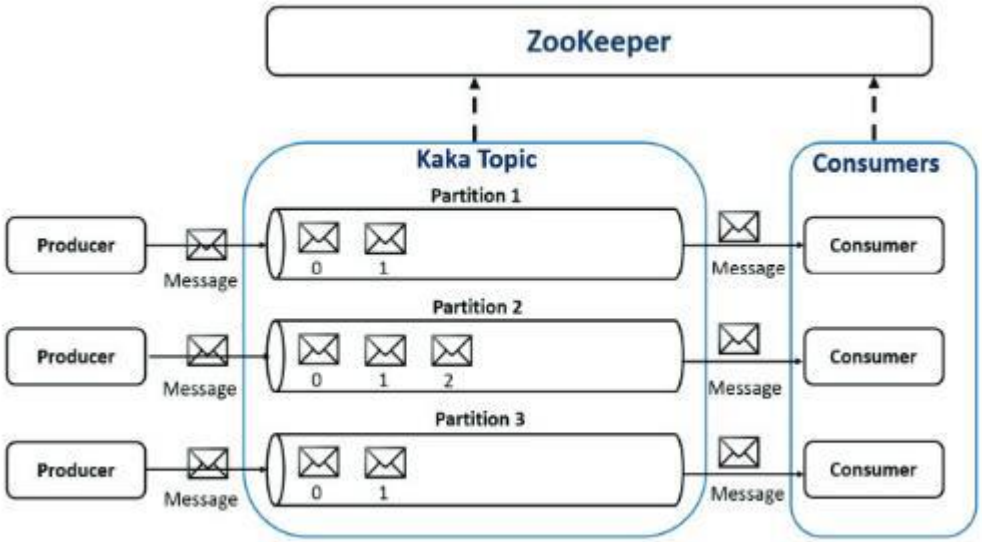


Figure 23 Apache Kafka Architecture [44]

As show in the figure above producers are agents that generate or transport the data to the Kafka broker. In the case of this experiment, Beats data shippers will act as Producers. Four different Kafka Topics will be needed to be created for Metric beat, Packet beat, Audit beat and File beat type logs. Each Topic is also managed into 4 different partitions for efficiency reasons.

Data is read from and written to the partition leader in Kafka, which is a distributed system. The cluster's leader can be on any broker. When a client (producer or consumer) starts up, it will ask for information on which broker is the partition's leader. This metadata request could come from any broker. The available endpoints for the partition's lead broker will be included in the metadata delivered. These endpoints will be used by the client to connect to the broker and read or send data as needed.

To install and setup Kafka the following steps are required. First a tar ball of the framework should be downloaded from the official download site. Apache Kafka Version 2.12–2.60.0.0 is used for this experiment. Then the tar ball is extracted using “tar -xvf tarball_name.tgz” command.
Home folder for Apache Kafka has the following contents:-

/bin	a folder consisting all executable binaries
/config	a folder consisting all configuration files used by Kafka
/libs	a folder with necessary libraries needed by Kafka to run
/site-doc	site documentation for Kafka
License	text file bearing the apache license agreement for distribution of Kafka
Notice	text file that contains the information on the owner of the software

Table 10 Apache Kafka contents

The /config folder for apache Kafka contains extensive configuration information that can be used to setup and manage a Kafka system. Some of frequently used config files in that folder include server.properties, consumer.properties, producer.properties, zookeeper.properties and log4j.properties. The first file that needs to be configured is server.properties file. It it used to set the properties for the Kafka broker.

Server.properties configuration file contains numerous variables. The following are some of the parameters that the researchers have considered setting values in this research. It is by no means the complete list of variables and their corresponding values.

#	Parameter/Variable	Value	Description
1	broker.id	0	The id of the broker. This must be set to a unique integer for each broker.
2	listeners	PLAINTEXT:/node-2.novalocal/:9092	The address the socket server listens on. it needs host name or IP address and a port where Kafka broker is going to run.
3	num.network.threads	6	The number of threads that the server uses for receiving requests from the network and sending responses to the network
4	num.io.threads	10	The number of threads that the server uses for processing requests, which may include disk I/O
5	socket.send.buffer.bytes	102400	The send buffer (SO_SNDBUF) used by the socket server
6	socket.receive.buffer.bytes	102400	The receive buffer (SO_RCVBUF) used by the socket server
7	socket.request.max.bytes	104857600	The maximum size of a request that the socket server will accept (protection against OOM)
8	log.dirs	/tmp/kafka-logs	A comma separated list of directories under which to store log files
9	num.partitions	4	The default number of log partitions per topic. More partitions allow greater parallelism for consumption, but this will also result in more files across
10	zookeeper.connect	Node-2.novalocal:2181	Address of zookeeper for Kafka to use.
11	zookeeper.connection.timeout.ms	18000	Timeout in ms for connecting to zookeeper

Table 11 Kafka Configuration Parameters and Values

There are numerous configurations setting used by the Kafka broker. Most of the parameters have default values that are usually optimal for single node configuration but when multiple nodes are used for Kafka, setting up the correct values is mandatory. For example the above server.properties file is a configuration for node-2 which is the Kafka broker node. But when we configure the Kafka clients, which are node-3 and node-4, we need client side configuration parameters to be set correctly. After setting up the configuration files, it is time to run the Kafka broker system on the server and start creating the necessary topics and getting ready for receiving logs from remote beats data shippers. But before that there are some important variables that are need to be set for zookeeper so that it can manage the Kafka broker efficiently. We need to set at least those 5 lines in the zookeeper configuration file.

tickTime=2000
dataDir=/path/to/zookeeper/data
clientPort=2181

initLimit=5
syncLimit=2

Since zookeeper manages the overall Kafka activities it needs to be started before we start the Kafka server. The syntax is usually to invoke the zookeeper-Server-start.sh script on the terminal. Once we are sure that zookeeper is running, we can start the Kafka server now. The syntax to to start is usually to run the Kafka server start script in the /bin folder with the location of the server.properties file that we configured above. bin/kafka-server-start.sh config/server.properties

The next step is to create the necessary Kafka topics. Since we had 4 different data sources, it is safe to create 4 different Kafka topics so that each can write to its own. Kafka usually provides a command line utility shell script program named kafka-topics.sh to create topics on the server.

#	command	Description
1	bin/kafka-topics.sh --create --zookeeper node-2.novalocal:2181 --replication-factor 2 --partitions 2 --topic metric-beat	Creates a Kafka topic named metric-beat using the zookeeper instance at node-2.novalocal on port 2181 with 2 replications and 2 partitions
2	bin/kafka-topics.sh --create --zookeeper node-2.novalocal:2181 --replication-factor 4 --partitions 4 --topic packet-beat	Since packet beat has the largest volume of data among the 4 log types, 4 partitions will be used for efficiency
3	bin/kafka-topics.sh --create --zookeeper node-2.novalocal:2181 --replication-factor 2 --partitions 2 --topic file-beat	Creating a Kafka topic for file beat data entries. It will have 2 partitions and 2 replications of the data.
4	bin/kafka-topics.sh --create --zookeeper node-2.novalocal:2181 --replication-factor 2 --partitions 2 --topic audit-beat	Audit beat Kafka topic with default replication and partition of 2.

Table 12 commands to create necessary Kafka topics

In the above command "--create" tells Kafka broker that the operation is to create a Kafka topic. Whereas the "--zookeeper" option tells which zookeeper instance to register the topic to so that consumers can subscribe to it later. "--replication" tells how many times a data need to be replicated on the servers. "--partitions" sets the number of partitions the topic needs. Eventually we need to set the name of the topic with "--topic" label. The name of the topic in this case is metric-beat. Once the topic has been created, you can get the notification in Kafka broker terminal window and the log for the created topic specified in "/tmp/kafka-logs/"in the config/server.properties file. Since Spark

is going to be our primary consumer for the Kafka topics created here, we need to make sure that the spark streaming session can access the nodes where the topics are created.

4.3.4 Data pre-processing & Stream Management

Data pre-processing and stream management is the most important task in streaming analysis systems. Logs have a lot of different types, such as server logs, database logs, event logs etc. The contents of logs include numerical data and non-numerical data as well. For some kind of logs, like server logs, those numerical data (for example CPU load, memory) is sufficiently representative as features. But the meaningful non-numerical data (such as the indicators about the state of systems) are also interesting.

One of the challenges of streaming big data systems is efficiently handling incoming traffic of data and processing them without any significant delay. As explained in chapter two of this document, Apache spark use a Dstream to micro batch incoming traffic using time slices. The stream management in the case of this experiment focuses on subscribing to Kafka topics, establishing connection to Kafka topic, and reading buffer from topic, closing connection, re-establishing connection if connection is lost or timed out and pushing messages to the analysis engine.

```
1  #Stream management of logs
2  #Author Muluken Sholaye
3  import pyspark
4  from pyspark import RDD
5  from pyspark import SparkContext
6  from pyspark.streaming import StreamingContext
7  from pyspark.streaming.kafka import KafkaUtils
8  import json
9  #Spark context details
10 sc = SparkContext(appName="StreamingLogManagement")
11 ssc = StreamingContext(sc,1)
12 #Creating Kafka direct stream
13 dks = KafkaUtils.createDirectStream(ssc,"Packet-beat" {"metadata.broker.list":"node2.novalocal:9092"})
14 #Starting Spark context
15 ssc.start()
16 ssc.awaitTermination()
```

Figure 24 Spark Streaming from Kafka Python code fragment

Lines 3 to 8 in the above code fragment imports important libraries that are needed to manage stream data from Kafka topics. RDDs are Apache spark's main data abstraction whereas Spark context and Streaming context are instances for spark core and streaming modules respectively.

Kafkautils is a library to interface Kafka broker from spark framework. On line 11, a streaming context instance is triggered with a 1 second streaming window. Line 13 creates a direct stream to Kafka broker at located at “node-2.novalocal:9092” specifically to a topic named “packet-beat”. Line 15 and 16 starts the streaming and continue streaming as long as a termination signal is received from the system. The complete code is submitted as a spark job to a spark scheduler using the command “spark-submit”.

One of the first activities is Parsing incoming log(i.e. for example if incoming data is syslog format we need a syslog parser which will analyze the log and break to individual pieces such as time, User, Log type, Log content etc.).The next is to Tag the log so that it‘s sequence is known in case there comes a need to use windowed operation and also logs can be identified from other. Sometimes it is also a necessity to covert logs into appropriate log formats if the incoming data is different from the expected format (for example if Log came in XML format where JSON format is expected, we need an XML to JSON converter module to apply to the data). To parse syslog logs the syslog_rfc5424_parser library is used. The following code fragment shows how the library is applied to incoming socket of log data.

```
import argparse
import json
from syslog_rfc5424_parser import SyslogMessage, ParseError
while True:
```

Figure 25Spark streaming creating Dstreams every one second

```
        try:
            message = SyslogMessage.parse(message)
            print(json.dumps(message.as_dict()))
        except ParseError as e:
            print(e, file=sys.stderr)
```

Figure 26 Code Snippet of Syslog parser python code

As shown in the figure above the syslog_rfc5424_parser library will parse the syslog document and dumps the output on a JSON file format, which is convenient data representation format for most data analysis and visualization systems.

On the other hand packet beat data set is not usually in human readable format. But luckily there is a python module that is designed to parse and convert it to a more manageable format such as dictionary data and JSON data.

```
import netflow
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("0.0.0.0", 2055))
payload, client = sock.recvfrom(4096)
p = netflow.parse_packet(payload)
assert p.header.version == 9  # NetFlow v9 packet
assert p.flows[0].PROTO == 1
```

Figure 27 Snippet code for Parsing netflow data

In the above python code fragment, a Data gram socket is initialized on port 2055 which is the port where Kafka messages are routed and it will receive a batch of 4MB of data after checking that it is version 9 net flow data. The other data types such as metric beat and audit beat usually come in standard formats so there is no need to apply any parsing and converting.

4.3.5 Data Analysis and Indexing

Once the data pre-processing and stream management is efficiently performed, the next step is to apply the actual analysis on the data and writing the output to the indexing tool. There are two possible ways the researchers used to analyze incoming data set. The first is by writing a pyspark script which takes the input data and returns the result after computing predefined analytical task. On the other hand, it is also possible to write the data directly to elastic search index and query the elastic search using the mathematical computation needed afterwards.

```
import re
from collections import Counter
import json
data = json.dumps(sc)
cnt = Counter()
ipre = re.compile(r'^(?P<ip>(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|
with open(data) as infile:
    for line in infile:
        m = ipre.match(line)
        if m is not None:
            ip = m.groupdict()['ip']
            cnt[ip] += 1
```

Figure 28 Code snippet to count frequent IP addresses from incoming log

Depending on the type of result needed, various analytical techniques were applied. It spans from a simple counting and ranking(for example to rank the top 10 packet generating hosts in a specified time frame, we need to count each host and display the output in ascending order) to more complex metric manipulation function such as calculating system load from a series of program CPU load information. Those analyses will be applied to the data RDDs in a distributed way. For example the following python code fragment shows a simple analysis to count the number of IP addresses from an incoming data.In the above code snippet “data” is a JSON file dump of incoming data piped from the spark stream management system. Regular expressions are used to filter IP addresses from the data set because different net flow data representation use different attribute names to represent IP addresses in a log. Once spark completes analyzing the data, the result will be written to the target elastic search index in the following way.

```
1  rdd = json.dump(result)
2
3  es_write_conf = {
4          "es.nodes" : "node-2.novalocal",
5          "es.port" : "9200",
6          "es.resource" : 'filebeat/apache',
7          "es.input.json": "yes",
8          "es.mapping.id": "doc_id"
9      }
10
11 rdd.saveAsNewAPIHadoopFile(
12         path='-',
13         outputFormatClass="org.elasticsearch.hadoop.mr.EsOutputFormat",
14         keyClass="org.apache.hadoop.io.NullWritable",
15         valueClass="org.elasticsearch.hadoop.mr.LinkedMapWritable",
16         conf=es_write_conf)
```

Figure 29 Writing results to ES from Spark

Line1 in the above code snippet shows the result data set being dumped to a new variable called rdd using json.dump function. On line 3 we have provided the elastic search information to be used when writing the data. It specifies where the instance is located, what port it is running on and other significant information. Then starting from line 11 the data set is been written to an elastic search index using the built in saveAsNewAPIHadoopFile function.

Elastic search is used to index and store the processed data in an efficient way so that users can query using Kibana dashboard KQL query language or by specifying search and filter criteria on the search bar. Elastic search provides distributed, real-time full-text search and analytics engine that's open source. It uses schema-less JSON (JavaScript Object Notation) documents to store data and is accessible through a RESTful web service interface. Since it is based on the Java programming language, Elastic search can run on a variety of platforms. It allows users to quickly and efficiently explore large amounts of data.

In elastic search, a document is a set of fields organized in a specific way and specified in JSON format. Every document is assigned to a type and is stored in an index. A unique identifier known as the UID is assigned to each text. Shard Indexes are subdivided horizontally into shards. This means that each shard has all of the document's properties but has less JSON objects than the index. Shard is an independent node that can be stored in any node due to the horizontal separation. The initial horizontal component of an index is called a primary shard, and these primary shards are then replicated into duplicate shards. Last but not least, Elastic search's Replicas feature enables users to make duplicates of their indexes and shards. Replication not only increases data availability in the

event of a malfunction, but it also enhances search efficiency by running a parallel search process in these replicas.

Elastic search will store indexed Document entries which are usually split into shards for the purpose of Replication. Apache spark will be the main source of Production of indexed document and once documents are indexed and put in elastic cluster, they remain there for consumption by visualization tools such as Kibana.

Elastic search is relatively easier to install and use than the other search engines such as Solr. After downloading the tar ball from the official elastic search download site, it needs to be extracted and moved into the desired location. Elastic search Version 7.9.3 is used in this experiment. The main contents of the extracted folder are /bin which consists major executable programs, /config consists all configuration files, /data contains slices of data pushed to elastic search, /libs contains libraries necessary for elastic search to function properly and /module which contains major modules of the elastic framework.

4.3.6 Data Visualization and Reporting

Data visualization and reporting is the last step in the data analytics life cycle. Once data is correctly indexed and stored in elastic search, Visualization tools can easily query using the REST API and display useful information to users in real time.

Kibana is an open source browser-based visualization tool that is primarily used to analyze large volumes of logs using line graphs, bar graphs, pie charts, heat maps, area maps, coordinate maps, gauge, targets, timelines, and other visualizations. The visualization makes it simple to forecast or see shifts in error patterns or other important input source events. It offers features such as Dashboards, Visualization, Dev Tools and Reporting Mechanisms, Filters and Search query,Coordinate and region maps, Timelion and Canvas features.

Kibana Version 7.9.3 is used in this experiment. The installation and configuration needed for Kibana is straight forward. It only involves downloading the tar ball from the official Kibana download site and extracting it. The main contents of the extracted folder include bin, config, data, node, node-modules, plugin and x-pack. /bin folder contains executable programs and scripts that used to run and hot-configure Kibana whereas /config folder contains the major configurable .yml files needed to set the necessary attributes needed. Kibana.yml is the major configuration file that is mandatory to be configured before starting the Kibana service. The following are the main attributes and their corresponding values to be set in the configuration files.

#	Attribute	Value	Description
1	server.port	5601	Kibana is served by a back end server. This setting specifies the port to use.
2	server.host	Node-1.novalocal	Specifies the address to which the Kibana server will bind.
3	server.maxPayloadBytes	4048576	The maximum payload size in bytes for incoming server requests.
4	elasticsearch.hosts:	["http://node-1.novalocal:9200"],["http://node-2.novalocal:9200"]	The URLs of the Elastic search instances to use for all your queries
5	elasticsearch.requestTimeout	50000	Time in milliseconds to wait for responses from the back end or Elastic search.
6	server.ssl.enabled:	True	Enable SSL transport layer security to encrypt the traffic between systems
7	server.ssl.certificate:	/home/centos/server.crt	The location of the SSL certificate to use.
8	server.ssl.key	/home/centos/server.key	The location of the key to be used.

Table 13 Kibana Configuration Parameters

Kibana provides an interactive and easily customizable environment to create state-of-the-art dashboards. It allows creating dashboards with contents such as but not limited to histograms, line graphs, pie charts, sunbursts, heat map, geographic plots and more. It also incorporates some example dashboards that can be used as templates to design your dashboards. Kibana can let you access all the data in a certain elastic search index using its discover tab as shown below

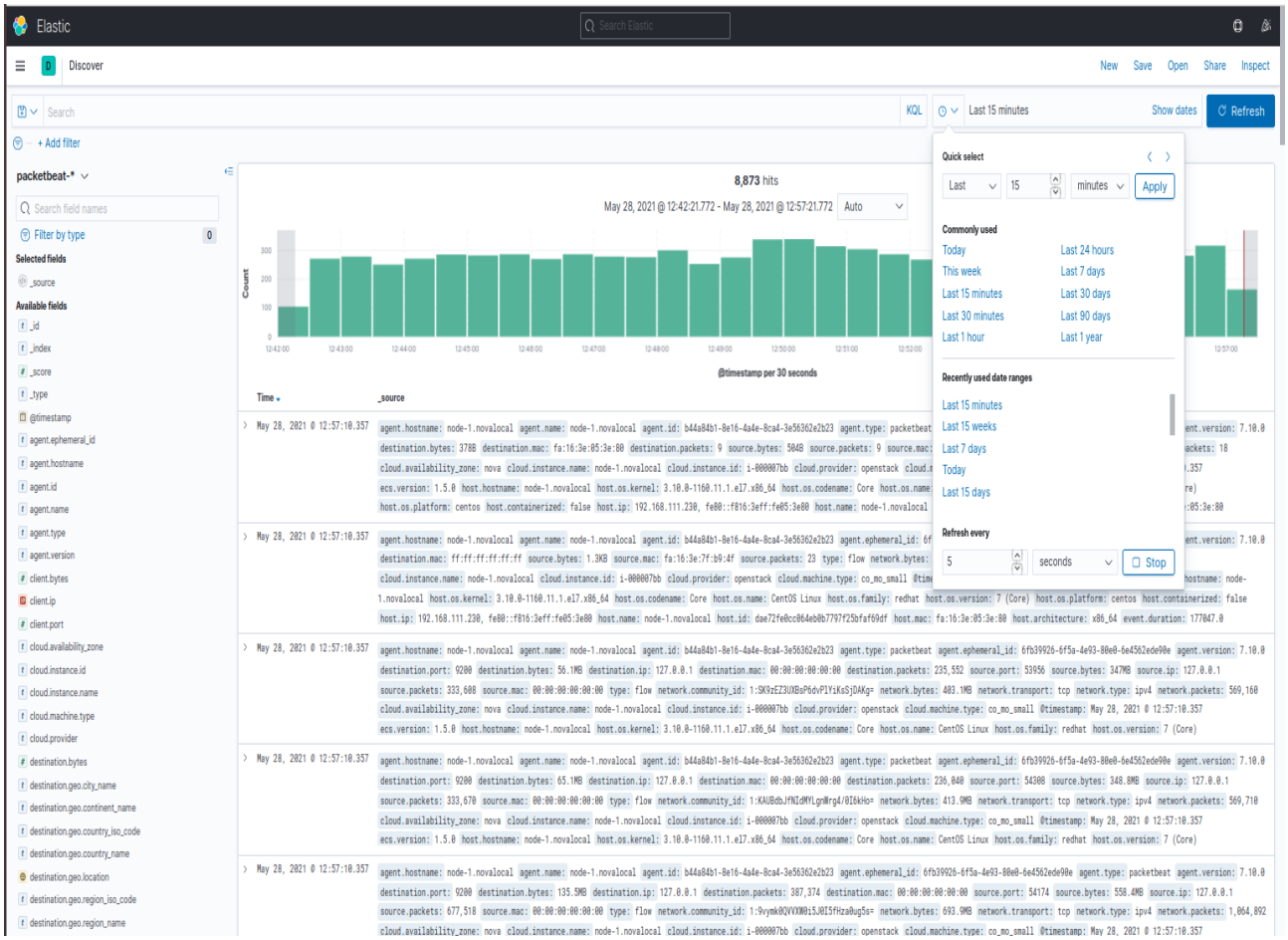


Figure 30 Packetbeat Live Raw data in Kibana Dashboard

The above screenshot shows that the system metric beat data in the past 15 minutes. Since going through all data manually is not applicable, there are some circumstances under which examining the raw data may be beneficial. In the discover tab of Kibana dashboard latest logs come up at the beginning of the screen, which makes it easier to live view recent data very easily if you are looking for some kind of investigative information. It is also possible to design dashboards by using any of histograms, bar charts, heat maps, location maps, pie charts and Gauge meters.

Figure 31Live Memory usage(left) Total Memory Usage(right)

The figure on the left shows how much of memory is used and how much is left during the past 5 seconds. Unusual system load may indicate that there is some kind of problem in the system. It is also possible to stream the metric information as it happens by setting the duration variable to "now" instead of 24 hours.

CPU usage is the other metric information that needs a close look. For most systems, a CPU load exhausted by unusual load can malfunction or crash the whole system. There are also some know denial of service attacks that try to overload the CPU by running a series of instructions so that denying the availability of the system to the designated use.

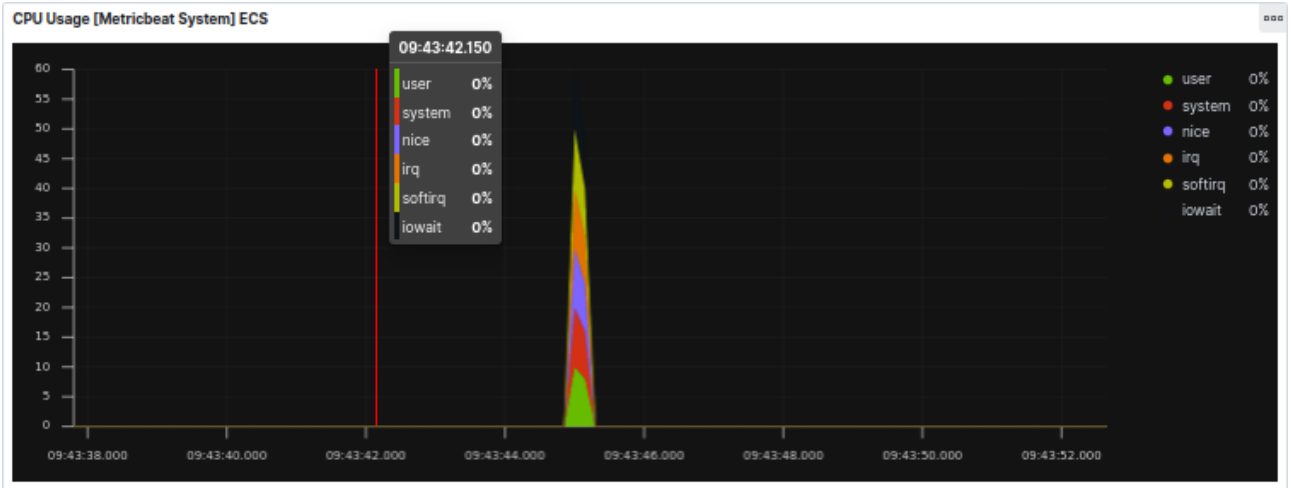


Figure 32 CPU load Visualization Panel in Kibana

The above sample CPU load screen shot shows what percentage of the CPU is occupied by system, nice, irq, softirq and iowait processes. It is snapshot information of the 2 seconds between 09:43:44:00 and 09:43:46:00. After that the window will slide left to show the next 2 seconds of information and the pattern continues as long as the connection to the system is alive.

Audit beat can also provide useful information such as what kind of system events are happening in the target system and user activities as well as login and logout times of each user.

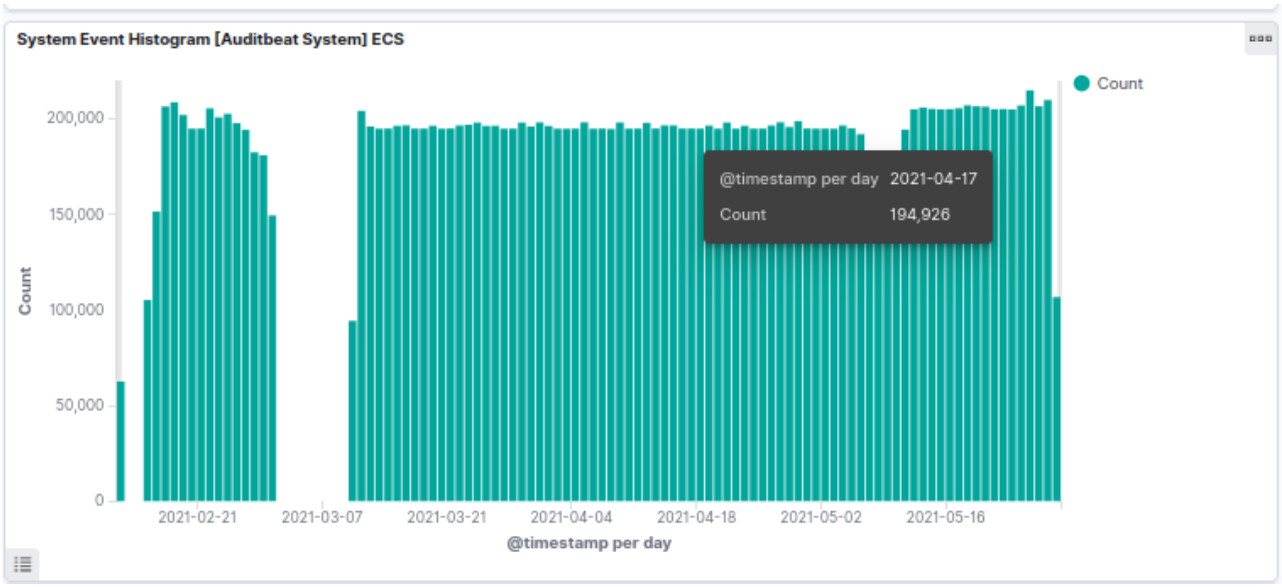


Figure 33 Weekly system Event Histogram

From the screenshot above, it can be noticed that there is a significant drop in the number of counts of events on 2021-03-07. A further drill down investigation will provide the reason why did the drop in system events happened on that specific day.

The other useful information that can be displayed about a system is a number of activities by each user in a specific duration of time. Usually system administrators are aware of magnitude of processes and programs that are running on high valued system assets in organizations such as web servers, data base servers, ERP system servers etc. A sudden spike or unusual change in the process window can be a tip off that something is happening that needs investigation.

DNS servers are also important assets to an enterprise. A DNS system that is performing below par can cost an enterprise a lot. DNS servers can be exposed to attack such as Denial of service; R2L attacks any other countless types of attacks. Monitoring the average number of DNS requests and response time by each request in a specified time interval can tell the health of the DNS server. The following screenshot shows the number of DNS requests and number of seconds needed for each

request to be processed. The time duration in this case is set to "7 days" but it can be set to any other duration such as 7weeks, 1 day, 5 minutes, 5 seconds or 4 months.

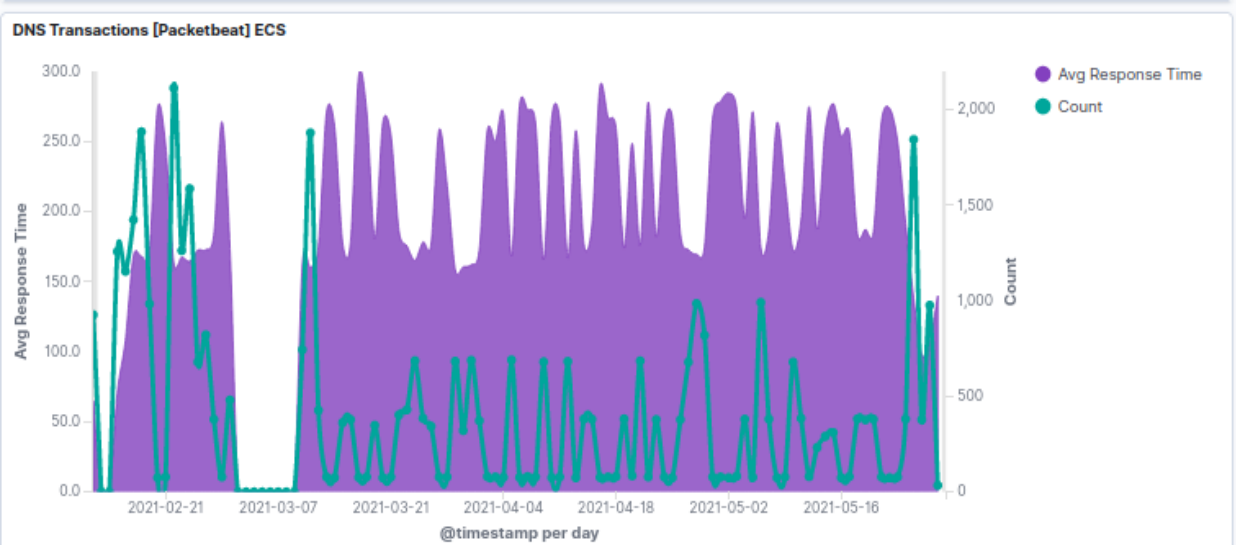


Figure 34 DNS requests per day and average Response time

In the above sample chart, it can be noticed that the week has a stable response time throughout the week but the number of DNS request has spiked in the beginning and end of the week. By setting the duration to those specific days, one can perform drill down analysis and find out the reason why those request spikes has happened.

Network connection trends are also good indicators about the health of a network in an enterprise. A number of unique connection can be a tip off that an enterprise is attracting more new site visitors or it may imply to some kind of intrusion depending on the nature of use of the network.

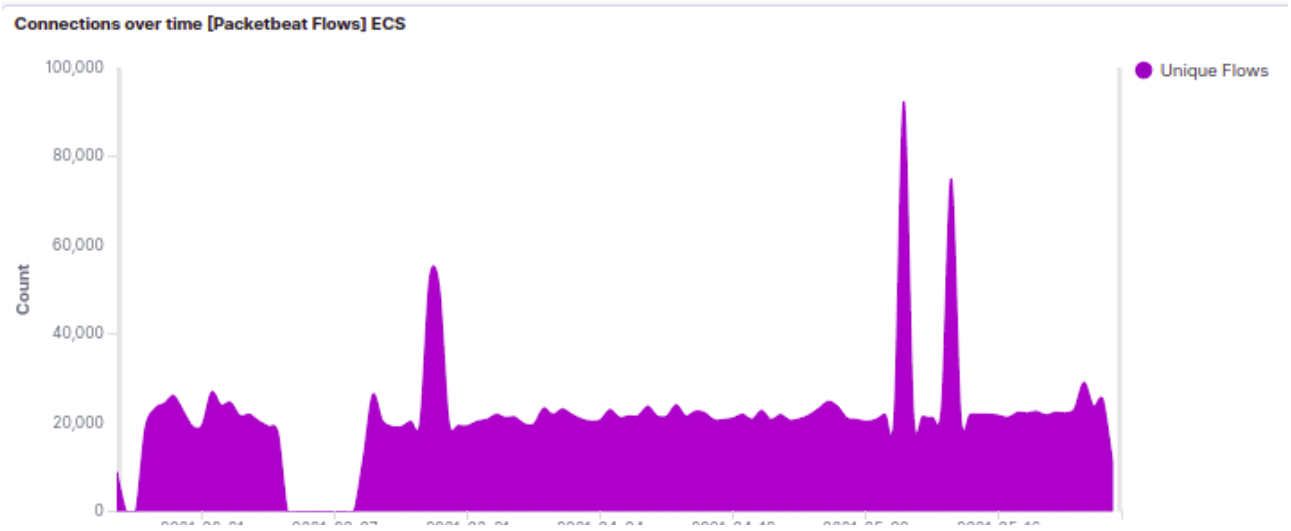


Figure 35 Number of Network Connections in 5 min 1

The above screenshot shows the number of unique connections in the past five minutes. The term unique only means that multiple connections using the same IP address are counted as 1 connection. It does not mean that the IP address is never been used to communicate with the target system.It is also vital for a system to know which IP addresses are frequently in communication with the target network gateway. Excessive connection to the system could possibly indicate a DoS attack on the network.

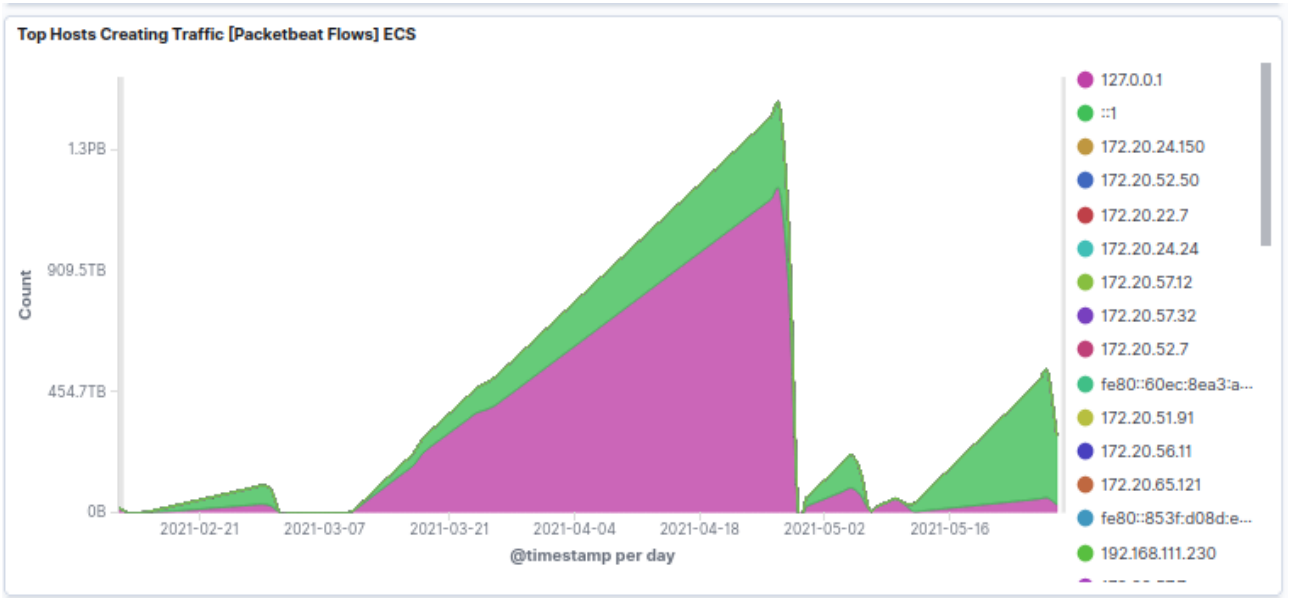


Figure 36 Top hosts creating traffic to the network

The above chart shows the top most traffic generating hosts. Enterprises usually perform seasonal internal network audits and this kind of information could be of a great use to that.

In some cases it may also be vital to track where the hosts that are connecting to our network are located in the world. By using GeoIP information, it is possible to pinpoint the country in which the connection is coming from. This functionality might be compromised by the use of certain IP blocker programs such as certain VPNs and anonymity platforms like Tor browser.

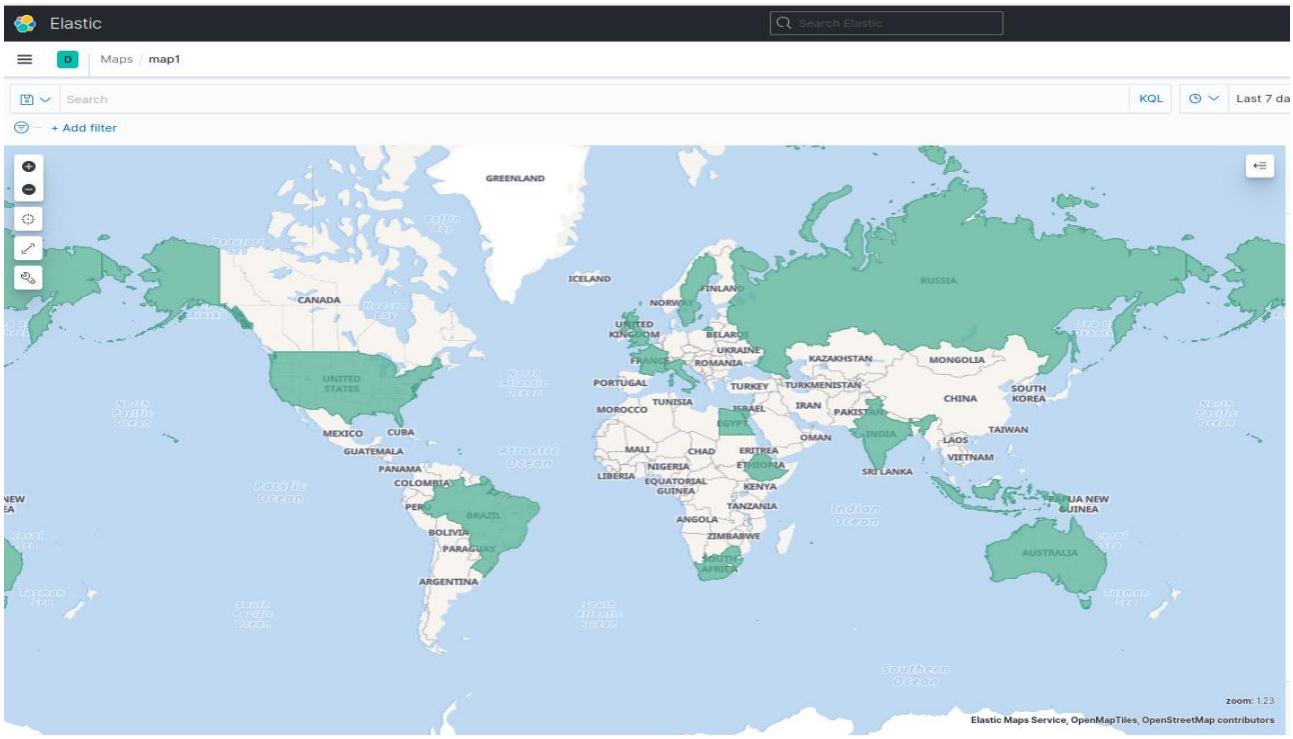


Figure 37 Geographic locations of most frequent connections to the network

For statistical purposes it is also possible to know the total number of packet connections in a specified time interval. For example the Gauge meter below shows the total number of incoming and outgoing connection in the past 24 hours. It is also possible to set the duration to a much smaller interval like 5 seconds, 1 minute or much bigger like 5 weeks, 10 months or 2 years.

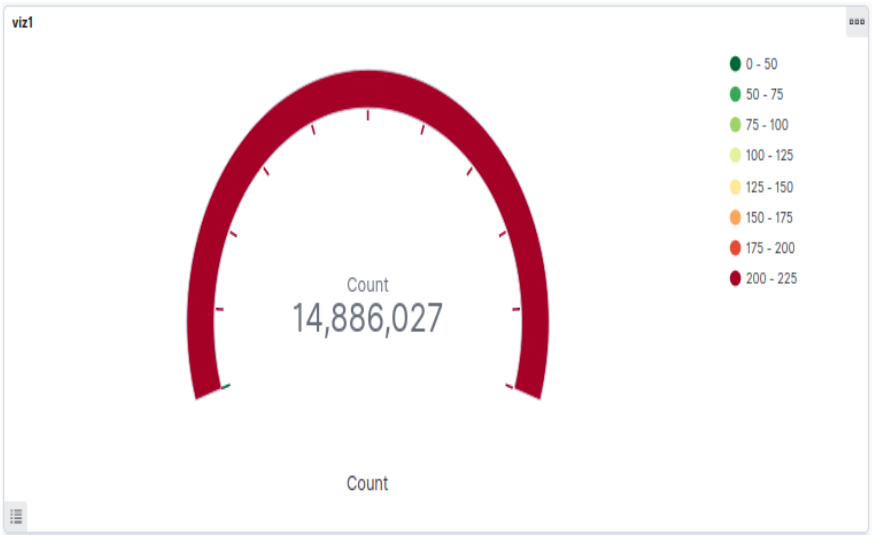
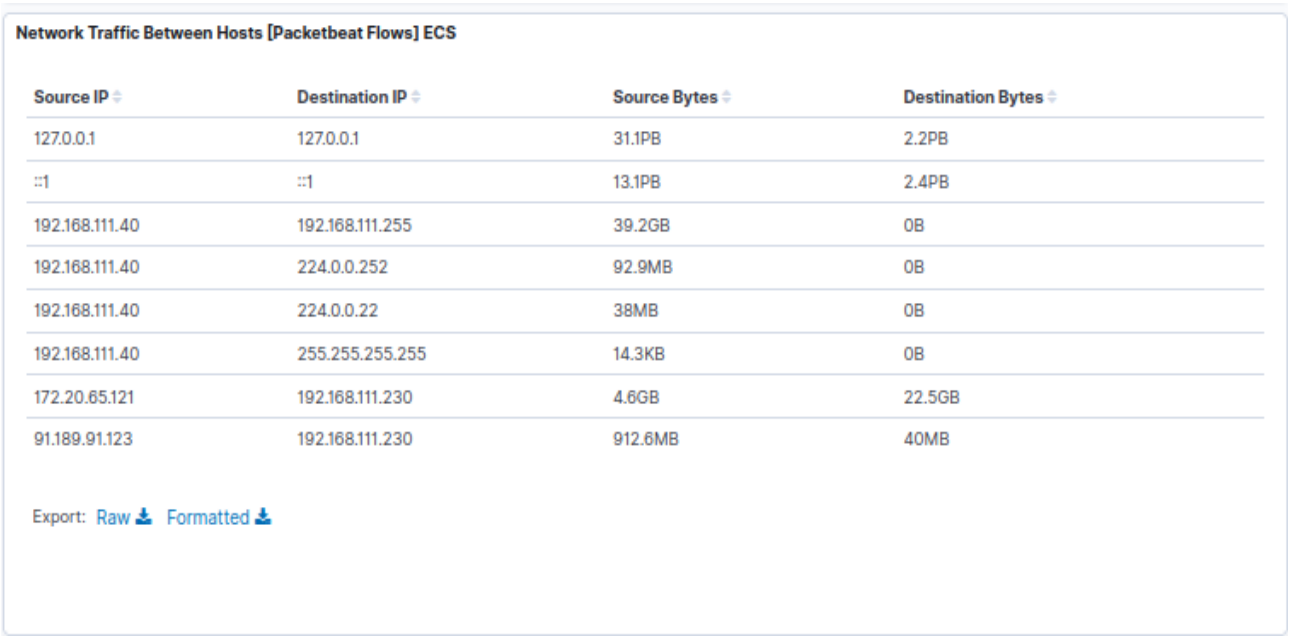


Figure 38 Total number of network connections in 24 hours

Sometimes it is also important for security operators to track the connections and data transfers each hosts in the network made within a specific time interval. Enterprises that work on classified

information such as security and intelligence agencies usually track the activities of each of their employees in real-time. Audit reports by internal or external auditors can also use this information to prepare employee or department level audit report.



Network Traffic Between Hosts [Packetbeat Flows] ECS

Source IP	Destination IP	Source Bytes	Destination Bytes
127.0.0.1	127.0.0.1	31.1PB	2.2PB
::1	::1	13.1PB	2.4PB
192.168.111.40	192.168.111.255	39.2GB	0B
192.168.111.40	224.0.0.252	92.9MB	0B
192.168.111.40	224.0.0.22	38MB	0B
192.168.111.40	255.255.255.255	14.3KB	0B
172.20.65.121	192.168.111.230	4.6GB	22.5GB
91.189.91.123	192.168.111.230	912.6MB	40MB

Export: Raw Formatted

Figure 39 Sample packet beat network traffic between hosts

Erroneous transaction are usually needs to be watched closely while monitoring an enterprise asset such as servers, cloud systems and IS systems. Too many erroneous transactions will degrade the overall productivity of an enterprise which could reflect on revenues and other Key performance indicators.

Figure 40 Erroneus transactions in network in the past 5 minutes

All the above individual visualizations can be unified into a same Kibana dashboard page so that system admins and security operation center workers can find a multi-dimensional view of the health of their enterprise's assets in one window.

4.4 Evaluation of the Model

It is always been not easy to evaluate a full cycle big data analytics platform because it consists of numerous frameworks and technologies integrated to form a system. As a matter of fact, Evaluation mechanisms for stream processing systems are a hot research area among the data science community nowadays. Different researchers have proposed possible mechanisms but they are specific to the type and nature of the system hence lacking generic and widely accepted evaluation criteria in the domain.

The researchers have used two categories of evaluation methods. The first is to use various evaluation mechanisms to measure the overall performance, fault tolerance, scalability and efficiency of the log management system. For example performance can be measured by metrics such as processing time, scheduling time, latency, total system Memory usage, Total CPU time, total time spent on I/O ,data transfer rate and Batch processing time. Scalability, on the other hand, can be measured through metrics like average time needed to horizontally scale the system and average time needed to vertically scale the system. Fault tolerance is measured through replication time and percentage of node failures that the system tolerates to continue functioning.

The second evaluation mechanism is to review research works that propose various ways to evaluate streaming analytic systems. Although there are numerous research works on streaming big

data analytic applied to various use cases such as security, physics, engineering, health, agriculture and IOT, performance is the only metrics that most of the researches used as a common metrics to evaluate their corresponding system. The specific measures used to evaluate the performance may vary from paper to paper but the overall attempt was to measure the performance of the system against the proposed performance needed for their respective use case. The following paragraphs present the main evaluation metric results.

Since a log is ingested from the remote data source targets it is vital to measure the average data collection latency of the logs. The main approach to do this is to record the time stamp when the log is been sent by the beats data shipper and the time it is written to the respective Kafka topic. Then we can subtract the time of the later from the previous to get the average read latency of the system. A random batch of 100 logs from each category is taken from 12 different days and the time latency for each is summarized in the table below.

	Data transfer latency of random logs taken at a random time in 12 days												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**AVG**
Packet beat	4.3 s	8.1s	2.5s	12.1s	5.5s	6.1s	7.2s	4.4s	5.3s	5.3s	5.3s	6.8s	6.0s
File beat	8.1s	4.5s	3.8s	6.1s	6.9s	2.1s	1.9s	0.8s	0.9s	2.5s	3.1s	8.9s	4.1s
Audit beat	4.1s	0.8s	2.1s	3.7s	2.5s	2.0s	6.3s	0.9s	0.9s	0.9s	0.6s	1.1s	2.1s
Metric beat	0.9s	1.0s	8.1s	0.5s	4.6s	0.5s	4.4s	6.7s	5.1s	2.3s	4.1s	2.2s	3.7s
Average	4.3s	3.6s	4.1s	5.6s	4.8s	2.6s	4.9s	3.2s	1.0s	2.7s	3.2s	4.7s	3.9s

Table 14 Average network transfer time of logs to the engine

From the above table, it can be seen that Audit beat logs has the least average time of transport whereas packet beat logs took the longest time to ingest the 100 logs in the batch. The average total time needed to transport 100 logs is about 3.9 seconds which roughly means that 0.039 seconds are needed to transport a single log entry from the source system to the log management engine.

Read latency is dependent on some variables that are not considered in this research. Factors such as network speed, DNS and DHCP server's efficiency, number of network hops between the source and the destination systems and network technologies used such as Infiniband can influence the network time latency.

The average processing time needed is another key indicator in the performance of a log management system. Once the log is stream ingested to Spark's system, it is possible to measure the total processing time needed by each batch of data. A batch of 200 data is used from 12 different time slots and their processing time is presented as follows.

	Random Dstreams of 200 logs taken at a random time in 12 days												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**AVG**
Packet beat	0.4 s	0.2s	0.5s	1.1s	0.5s	0.1s	0.2s	0.4s	0.3s	0.3s	0.1s	0.2s	0.3s
File beat	0.1s	0.5s	0.4s	0.1s	0.1s	0.1s	0.1s	0.2s	0.1s	0.2s	0.1s	0.4s	0.2s
Audit beat	0.4s	0.8s	0.1s	0.7s	0.5s	0.2s	0.3s	0.5s	0.9s	0.9s	0.6s	0.1s	0.5s
Metric beat	0.9s	1.0s	0.1s	0.5s	0.6s	0.5s	0.4s	0.7s	0.1s	0.3s	0.1s	0.2s	0.4s
Average	0.4s	0.6s	0.2s	0.6s	0.4s	0.2s	0.2s	0.4s	0.3s	0.4s	0.2s	0.3s	0.3s

Table 15 processing time for 200 logs of random Dstream batches of data

The processing time for the 4 data type's shows that file beat takes the minimum processing time of 0.2 seconds whereas audit beat logs took the longest average time of 0.5 seconds. The average processing time is around 0.3 seconds for the 200 logs. It means it will take around 0.0015 seconds to process a single log entry.

Scheduling delay is the third metric that is used to evaluate the log management and monitoring system. When using distributed system, scheduling jobs and resources is the major task that should be handled efficiently. Although YARN and Mesos are frequently used resource and job schedulers in the Hadoop ecosystem, Spark's built in scheduler is used in this experiment. The following table summarizes the scheduling delay for a random of 200 logs from each data source over the period of 12 days. The method used to collect is accessing Apache spark's web interface and querying elastic search index data.

	Scheduling delay time in milliseconds(ms) of 200 logs taken at a random time in 12 days												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**AVG**
Packet beat	1 ms	1.2ms	1.5ms	1.1ms	1.5ms	0.9ms	1.2ms	1.4ms	1.2s	1.4ms	1.1ms	1.2ms	1.2ms
File beat	1.2ms	1.2ms	1.7ms	1.6ms	1.2ms	1.1ms	1.5ms	1.2ms	1.3ms	1.2ms	1.6ms	1.4ms	1.4ms
Audit beat	1.3ms	1.8ms	1.7ms	1.6ms	1.2ms	1ms	1ms	1.2ms	1.9ms	1.6ms	0.9ms	1.2ms	1.3ms
Metric beat	1ms	1ms	1.1ms	1.4ms	1.4ms	1.6ms	1.9ms	1.7ms	1ms	1ms	1.2ms	1.3ms	1.3ms
Average	1.1ms	1.3ms	1.5ms	1.4ms	1.4ms	1.1ms	1.4ms	1.3ms	1.3s	1.3ms	1.2ms	1.2ms	1.3ms

Table 16 scheduling delay for Dstream of 200 logs

According to the above results, the type of log has no significant effect on the delay of the scheduler. An average of 1.3 milliseconds is the scheduler delay expected for 200 logs of data and an average 0.0065 milliseconds for each log.

It is also possible to calculate an average memory usage for a cluster of logs. The same technique of selecting 200 random logs from each data type is used just like the previous evaluation procedures and the total memory is usage information is read from the spark Web interface. One of the main reasons why Apache spark is lightning fast compared to other frameworks is its support for in-memory computing. System memory is the only memory type considered in the result. The result is presented in the table below.

	Total Memory usage of batch of 200 logs selected among 12 days of data												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**AVG**
Packet beat	2.1mb	2mb	2.5mb	2.1mb	2mb	2mb	2mb	1.9mb	2mb	2.4mb	2mb	2.2mb	2.1mb
File beat	1.9mb	1.8mb	1.9mb	1.9mb	1.8mb	1.8mb	1.8mb	1.9mb	1.8mb	1.9mb	1.6ms	1.4mb	1.8mb
Audit beat	3.1mb	1.2mb	2.9mb	2.6mb	2.2mb	2.8mb	3.1mb	2.2mb	2.7mb	2.2mb	2.2mb	2.2mb	2.4mb
Metric beat	2.1mb	2.4mb	1.9mb	2.4mb	2.6mb	2.6mb	2.9mb	2mb	2.9mb	2.1mb	2.2mb	2.1mb	2.5mb
Average	2.3mb	1.8mb	2.3mb	2.2mb	2.1mb	2.3mb	2.4mb	2mb	2.3mb	2.1mb	2mb	1.9mb	2.2mb

Table 17 Memory usage of batch of 200 logs

Packet beat logs consumed average of 2.1mb of system memory whereas file beat consumed the minimum of all at 1.8mb of space. Audit beat and metric beat consume around 2.4 and 2.5mb of space respectively. The average memory consumption per batch is around 2.2mb for the 200 logs which translate to 0.011mb of space are needed for each piece of log instance.

Distributed systems are usually in need of horizontal and vertical scaling. Horizontal scaling is the addition of a new node alongside the existing ones whereas vertical scaling is the improving the capacity of the existing nodes by extending its computing capacity. In the log management arena, vertical scaling usually require a minimal configuration change such as extending the minimum and maximum memory usage limits on each node. Horizontal scaling on the other hand requires the installation of one or more of the executor nodes on the new node and addition of the name of the name and the address of it on the configuration file of the master node. It usually requires a reboot of the affected services for vertical scaling to take effect whereas horizontal scaling only requires committing the changes to the master node. Due to the fact that horizontal scaling is performed without disrupting the whole system, it is safe to say that the log management system is easily scaled horizontally than vertically.

Different literatures also propose various evaluation criteria for streaming applications. But most focus on specific environment, setting or use case. But few also tried to formulate ways to measure effectiveness of streaming analysis systems in general. For example [9] emphasizes the need for fault tolerance in streaming analysis. Fault tolerance for distributed system is obtained through replication of data on different nodes redundantly. In case when the node containing specific data is

offline, the backup node that contains the exact copy of the data can be used. The minimum replication factor for fault tolerance functionality is 2; which means each data is replicated to one more node. Larger systems usually have a higher replication factor since the number of nodes and the data they process is usually higher.

[9] Also argue that the main challenges in streaming analytic are scalability, integration, fault-tolerance, timeliness, consistency, heterogeneity and incompleteness, load balancing, privacy issues, and accuracy. The writers also argue that streaming analytic systems should be evaluated based on those key factors. On the design stage of this experiment, the researchers have considered most of the above mentioned key factors in the process. In the literature review part of this document, a brief comparison of various frameworks is presented in detail.

[13] Revises mostly used log analysis tools in the market to carefully recommend that heterogeneity, integration and performance could be the major factors when selecting appropriate logging tools for your business.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

This chapter is organized into two subtopics; conclusion and recommendation. The conclusion part starts by summarizing some of the concepts and ideas discussed in the previous four chapters in a direction that leads to bringing the answers to the research questions that were posed at the start of the study. The next section covers the concerns and future works that that the researchers would like to point out so that they can be addressed or further investigated.

5.1 Conclusion

The economic industry in the twenty-first century, dubbed the "information era," is heavily reliant on data. How much data must be processed in order for it to be useful? According to an IDC survey, barely 0.5 percent of all data created globally is analyzed. In a world where we produce as much data every two days as we did from the beginning of time until 2003, there is a need to link data analyzed with current trends to improved business models; systematic processing and analysis of big data is the underlying element.

The complexity of the IT system in most middle and large enterprises is increasing. Data is usually dispersed among departments and branch offices and subsidiaries. Ability to collect process and visualize those scattered data needs a new paradigm of applying big data stream processing rather than using traditional techniques. With the expansion of IT infrastructure into hosted and cloud deployments, there is now not only more data to manage, but it is now spread across multiple environments. Collecting and managing a steady stream of dispersed log data can easily overwhelm an IT department.

Despite the abundance of logging mechanisms in modern systems and devices, Logs are usually not efficiently utilized in many enterprise environments. In most cases logs are either completely forgotten or only noticed when something is gone really wrong other diagnostic mechanisms known to the IT department failed to work. But logs are reach sources of information if they are been managed and analyzed in an effective way. Log analysis isn’t easy, and can be messy. As opposed to the “plug-and-pray” approach, effective log analysis takes some work. Logs come in a variety of shapes and sizes, and at times it can be difficult to extract information from them.

Big data rests on the interplay of infrastructure, technology and mythology. It requires the integration of various efficient technologies for data collection, storage, processing, visualization and reporting by utilizing distributed environments to facilitate timely and accurate decision making process by minimizing cost and computing time when there is a need to obtain useful knowledge

from current happenings in an efficient and speedy manner in order to enable organizations to quickly react to problems, or detect new trends which can help improve their performance.

Streaming analytic sub-domain of big data allows processing of massive amount of data generated at high-velocity from multiple sources with low latency in real-time. It is a new paradigm necessitated because of new sources of data generating scenarios which include ubiquity of location services, mobile devices, and sensor pervasiveness.

Lambda, kappa, Zetta and Iot Are the four mostly deployed architectural trends when designing big data streaming analytic systems. The Lambda architecture was one of the first to be proposed for Big Data processing, and it has since become the industry standard. The Kappa architecture was next, followed by many other architectures designed to address some of the limitations of the lambda architecture for use cases where the former standard failed to provide satisfactory results. IoT is the latest architecture which has gotten wide spread recognition for its suitability for designing smart systems such as self-driving cars and smart homes.
A fundamental problem with log management that occurs in many organizations is effectively balancing a limited quantity of log management resources with a continuous supply of log data. Log generation and storage can be complicated by several factors, including a high number of log sources; inconsistent log content, formats, and timestamps among sources; and increasingly large volumes of log data. Log management also involves protecting the confidentiality, integrity, and availability of logs. Another problem with log management is ensuring that security, system, and network administrators regularly perform effective analysis of log data.

Using stream processing platforms can easily remediate the above mentioned complexities of handling log data using traditional computing systems. But applying streaming analytic has its own challenges. The first challenge is selecting suitable framework that best fits the design in hand. There is a wealth of big data frameworks in the market and choosing among those is usually not an easy task. The other problem is difficulty of integrating different and independent frameworks. It is known that big data usually deals with the whole data analytic ranging from data acquisition to data visualization and reporting. In order to manage that, it requires integration of different independent tools which usually creates lots of problems. It requires an in depth study before selecting frameworks.

There are numerous frameworks that are designed to manage streaming data in an efficient way. But they usually use one of the two execution models. These are two execution models used in stream processing are stream data flow and Micro batch approaches. In Stream Data flow Approach, an

application is viewed as a data flow graph with operators and data dependencies between them (where sometimes mentioned as operator-based approach).A task encapsulates the logic of a predefined operator like filter, window, aggregate or join or even a routine with user-specified logic. The Micro-Batch Approach, on the other hand, enables processing of data streams on batch processing systems. With this approach, we can treat a streaming computation as a sequence of transformations on bounded sets by discretizing a distributed data stream into a series of batches, and then scheduling these batches sequentially in a cluster of distributed worker nodes.

The most frequently used open source stream processing frameworks include Apache Spark, Samza, Flink and Storm. Apache Spark provides in-memory computing using its native resilient distributed dataset (RDD) abstraction model. It has also an easy integration with various other tools such as Kafka, Hadoop HDFS and Flume. It can also be programmed using languages such as java, scala, python and R making it by far programming friendly tool in the list. Among other reasons, because of the above mentioned reasons Apache spark is the best open source stream processing framework in the market.

After careful literature review of various technology designs for stream log management and analysis, the researchers found out that the integration of beats data shippers, Kafka message broker, Apache Spark Engine, Elastic search indexing and Kibana visualization can be used to design fault tolerant, easily scalable, efficient and lightning fast real time log management and monitoring platform.

5.3 Recommendations

Stream processing is in its infancy stage. There is a steady progress by researchers nowadays and it is predicted that it will continue to develop in the coming years. The following are some of the research areas in stream big data analytic that interested researchers can further investigate.

1. The application of streaming big data analytic for web servers click stream is in the agenda in recent years. Since websites are becoming integral organizational assets, analyzing website click stream and usage mining in real time will be very interesting.
2. Evaluation of stream processing platforms is usually difficult. Since abundance of social media and IoT are forcing stream processing to be a trend, further research works that propose better ways of evaluating stream processing systems are expected.

Bibliography

[1] Brar Yaqoob, Ibrahim Abakar, Targio Hasem, and Abdullah Gali, "Big data : From begenning to Future," *International Journal of information Management*, 2016.

[2] Nikoleta tantalaki, Stavros souravlas, and Malos Rouzmeliotis, "A review on Bigdata realtime stream processing and its scheduling techniques," *University of Macedonia Quarterly*, p. 38, march 2019.

[3] Kevin Taylor-Sakyi, "Big Data: Understanding Bigdata," 2016.

[4] K, Sam M., "Bigdata Concepts and Architecture," 2020.

[5] Hasib E. M El-Desouky and A. I. Labib, "An Imbalanced bigdata Classification Framework Using Whale Optimization and Deep Neural net".

[6] Gayantri Kapil, Alka Agarwal, and Raees Khan, "A study of bigdata characterstics," 2016.

[7] Danah Boyd and Kate Crawford, "CRITICAL QUESTIONS FOR BIGDATA," 2012.

[8] Muhammed Jibril, "Social Media Analytics Driven CounterTerrirism Tool to improve Intelligence Gathering Towards Combating Terrorism in Nigeria," 2017.

[9] Kolajo Taiwo, Daramola Olawunde, and Adebiyi Ayodele, "Bigdata Stream Analytics: A systematic Literature Review," 2019.

[10] Sahoo P.K and Chottray R.K, "Syslog A promising Solution to Log management," 2019.

[11] Sosnowski Janusz, Gawkowski Piotr, and Cabaj Kryzysztof, "EVENT AND PERFORMANCE LOGS IN SYSTEM MANAGEMENT AND EVALUATION," 2011.

[12] Howard Barringer, Groce Alex, Havelund Klaus, and Smith Margareth, "Formal Analysis of log files," 2010.

[13] Dileepa Jayathilake, "Towards Structured Log analysis," 2012.

[14] Kodali Paith, "Bigdata Analytics with splunk," 2013.

[15] Thosar Sandesh et al., "Survey on Log analysis and Management," vol. 3, no. 6, 2015.

[16] Fareed Khan et al., "Bigdata From Beginning To Future," 2016.

[17] McAfee Andrew and Erik Brynojolfsson, "Bigdata : The management Revolution," 2012.

[18] Souppaya Murugiah and Kent Karen, "Guide To Security Log managment: Recommendations of The National Standards and Technology".

[19] Taylor-Sakyi, Kevin;, "BigData : Understanding Bigdata," *Engineering and Applied Science*, 2016.

[20] Kaiser Giri and Towseef Lone, "Bigdata: overview and challenges," *International JOurnal of Advanced Research in Computer science & Software Engineering*, vol. 4, no. 6, june 2014.

[21] Rong Liu, Feng Li, Qicheng Li, and Lijun May, "BigData Architecture for IT incident management," *IEEE interanational conference on Service Operations and Logistics, and INformatics(SOLI)*, October 2014.

[22] Moody Amakobe, "A Comparison Between Apache Samza And Storm," *Colorado Technical University Press LTd*, February 2016.

[23] Karen Kent and Murugiah Souppaya, "Guide to security Log management," *NIST-Special Publication 800-92*, september 2019.

[24] A Chuvakin, K Shmidt, and C Philips, "Logging and Log management: The Authoritative Guide to Understanding the concepts surrounding Logging and Log management," *Journal of Information Security*, December 2012.

[25] Steve Anson, "Event Log Analysis," in *Applied Incident Response*. USA, 2020, pp. 199-234.

[26] Prasanta K Sahoo, R K Chottrey, Gunammani Gena, and S Pattniak, "Syslog A Promising Solution to Log Management," *International Journal of Advanced Research in Computer*

Science, May 2019.

[27] Laxman Natak and Arvind Kwalnakar, "Efficient Network Management using SNMP," *Journal of Network and systems Management*, February 2006.

[28] Pranita Bavaskar, Onkar Kemker, Aditya Sinha, and Dr M, "A SURVEY ON : "LOG ANALYSIS WITH ELK STACK TOOL"," vol. 6, 2020.

[29] Erik D Knapp and Joel Thomas, *Industrial Network Security: Second Edition*, 2nd ed., 2015.

[30] Vankatesh Naganathan, "Comparative Analysis of Big Data: Bigdata Analytics Challenges and Trends," *International Research Journal of Engineering And Technlogy(IRJET)*, vol. 05, no. 05, May 2018.

[31] Hiba J Hadi, Ammar H Shnain, Sarah Hadishaheed, and Azizahbt H Ahmad, "Bigdata and Five V's Characterstics," *International Journal of Advances in Electronics and Computer Science* , vol. 2, no. 01, January 2015.

[32] Kevin Edward, "A brief Introduction on Bigdata 5Vs Characterstics and Hadoop Technology," *International Conference on Intelligent Computing, Communication and Convergence*, vol. ICCC-2015, December 2015.

[33] Kaiser J Giri and Towseef Lone, "Bigdata Analytics : Overview and Challenges," *International Journal of Advanced Research in Computer Science and Software Engineering*, June 2014.

[34] Godson Koffie Kalipe and Rajat Kumar Behera, "Bigdata Architectures: A detailed and Applicaiton Oritented View," *Institute of Industrial Technology Journal* , October 2019.

[35] Mariam Kiran and Peter Murphy, "Lambda Architecture for Cost Effective for batch and speed processing," *Energy Sciences Network(ESN)*, October 2015.

[36] Yuvraj Kumar, "Lambda Architecture - Realtime Data Processing," *Quantum Computing and Artificial Intelligence*, January 2020.

[37] Soumaya Oumacer, Mohammed Talhaoui, and soufianne Ardichir, "A new Architecture for realtime data Stream Processing ," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, 2017.

[38] Bhumi Nakuva and Tushar Champaneria, "Study of Various Internet of Things Platforms," *International Journal of computer science and Engineering Survey*, vol. 6, no. 6, pp. 71-84, December 2015.

[39] Baoan Li and Jianjun YU, "Research And Application on The smart Home based on Component Technologies and Internet of Things," *Procedia Engineering Journal*, vol. 15, pp. 2087-2092, 2011.

[40] AbdulGafar Shoro and Tariq Rahim Soomro, "Bigdata Analytics: Apache Spark Perspective," *Global Journal of Computer Science and Technology*, vol. 15, no. 1, 2015.

[41] Xinyao Zhang, "The Analysis of Parallelism of Apache Storm," Concordia University, Montreal, Canada, Thesis Work 2016.

[42] Shadi Noghabi, "Samza: A stateful Scalable Stream Processing at Linkedin," *VLDB endowment*, August 2017.

[43] Jonas Traub and Volker Marki, "Apache Flink in Current Research," *Journal of Information Technology*, vol. 58, no. 4, January 2016.

[44] Wissem Inoubli, Sabeur Aridhi, and Haithem Mezni, "Comparative Study on Streaming Frameworks for Bigdata Analytics," *Latin America Data Science Workshop Procedings*, August 2018.

[45] Kabilan GnanavaroThayan and Just Van Stam, "Realtime Threat Detection Through Network Analysis," , 2019.

[46] Eugenio,Koga, Ivo Almeida, Marcio Santana, Patricia Guimaraes, Luciana Sugarwara, and Tero Eklin, "Exploratory Study of the ELK stack for meterological observation system data analysis," 2017.

[47] Michael Mitra and Davvid Sy, "The rise of Elastic Stack," 2016.

[48] Eva Kostrekova and Helena Binova, "Security Information and Event Management," *International Journal for Cyber Security* , vol. 4, no. 2, February 2015.

[49] Natalia Miloslavkaya, "Analysis of SIEM systems and Their Usage in Security Operations and Security Intelligence Centers," *Advances in Intelligent systems and Computing*, August 2018.

[50] Martin Petracca. (2021, August) www.wazuh.com. [Online]. http://www.wazuh.com

[51] Joshila Grace, Maheswari V, and Nagamalai Dhinaharan, "Analysis of Web logs and Web User In web mining," 2011.

[52] Sonali Agarwal and Bakshi Prasad, "High Speed data Analysis of Web generated log streams," 2015.

[53] Soyeon Park, Joon Lee, and Hee Bae, "End User Searching: A web analysis of NAVER, a Korean Web search engine," 2005.

[54] Madhury Mohandas and Dhanya Pm, "An Exploratory Survey of Hadoop Log Analysis Tools," vol. 75, 2013.

[55] Jing Xia et al., "An Online Visualization System for Streaming Log data of Computing Clusters," vol. 18, 2013.

[56] Xiaojian Liu, Yi Zhu, and Shengwei Ji, "Web log Analysis on Genealogy System," 2020.

[57] Kalle Piirainen and Robert Briggs, "Design Theory in Practice-Making Design Science Research more transparent," *Advances: MechelininKatu* , 2011.

[58] K Peffers, T Tuunanen, T Gangler, G Rossi, and W Hui, "The design Science Reseach Process: A model For producing and presenting information systems research," *Conference on design science research in information systems and technology*, pp. 85-106, Feb 2006.

[59] Matti Rossi, "Design Science Research- The road Travelled and the road that lies Ahead; An introduction to a special issue," *Journal of Database Management*, vol. 24, no. 3, pp. 1-8, 2013.

[60] Offerman Philip and Olga Levina, "Outline of Design Science Research Process," *Conference Proceddings of the 4th international conference on Design science research in information systems*, May 2009.

[61] Vijay Vaishnavi and B Kuechler, "Design Science Research in Information Systems," *JOurnal of Association for Information systems*, 2004.

[62] Richard BaskerVille, Abayomi Baiyere, Shirley Gregor, and Alan Hayner, "Design Science Research Contributions: Finding a Balance between artifact and theory," *Journal of associations for information systems*, May 2018.

[63] Tomislav Lipic, Karolj Skala, and Enis Afgan, "Deciphering Bigdata Stacks: An overview of Bigdata Tools," *JOurnal of Informatics and Computing*, 2017.

[64] Kumar Rahul and Rohitash Kumar Banyal, "Data Life Cycle Management in Bigdata Analytics," *International Conference on smart Sustainable intelligent computing and applications*, vol. 173, pp. 364-371, january 2020.

[65] Mohammed ElHaras and Nissrine Sousei, "Data lifecycle: From Bigata to Smart Data," *International IEEE congress on Information Science and Technology*, vol. 5, october 2018.

[66] Sivan Sabato, Elad Yom-Tov, and Aviad Tsherniak, "Analyzing System Logs: A new View of What is Important," *Haifa University Press, Technology Edition*, 2017.

[67] Temesgen Desalegn, "Bigdata Processing and Visualization in the context of unstructured data set," *Addis Ababa University Postgraduate Thesis*, June 2016.

[68] Logz.io. (2020, April) Logzio Tutorials. [Online]. https://logz.io/blog/beats-tutorial/

[69] James Hamilton, Brad Schofiled, Manuel Gonzalez Berges, and Jean-Charles Tournier, "SCADA Statistics Monitoring using The Elastic Stack(Elasticsearch, Logstash and kibana),"

CERN, Geneva, Switherland, Report 2019.

[70] Jang-Jaccard Julian and Nepal Surya, "A survey of Emerging threats in cybersecurity".

[71] Andriy Miranskyy, AbdelWahab Hamou-Lhadj, Enzo Cialini, and Alf Larsson, "Operational Log analytics for bigdata systems: Challenges and Solutions," vol. 1, no. 1, 2016.

[72] He Pinjia, Zhu Jieming, He Shilin, and Li Jian, "Loghub: Towards Automated Log Parsing for Large-Scale Log Data Analysis," 2017.

[73] Suozhu Wang, "A Method of E-governement Website Services Quality Evaluation Based on Web log Analysis," 2013.

[74] (2020, may) Logging and Log management Life cycle. [Online]. https://awontis.com/2018/12/12/the-10-steps-of-the-log-management-lifecycle-infographic/

[75] Kiranjit Pattniak, bahbani Shankar, and Prishad Mishara, "Introduction to Bigdata Analytics," in *Techniques and Environments for bigdata analysis studies in bigdata*. Geneva, Switzerland: Springer International Publishing Ltd, 2016, ch. 1.

[76] (2017, July) KDENuggets. [Online]. https://www.kdnuggets.com/2017/07/4-types-data-analytics.html

[77] (2020, March) Apache Spark. [Online]. org.apache.com/Spark

[78] Confluence. (2021, March) Confluence Apache Wiki Page. [Online]. https://cwiki.apache.org/confluence/display/METRON/metron-architecture

[79] S Kamalakkannan and S prasanna, "Web Usage Mining: Users Behavior in Web Page Based on Web log Data," *International Journal of Emerging TEchnologies in Engineering Research(IJETER)*, vol. 4, no. 7, july 2016.

[80] Bhole Rahul Hiraman, Chapte Viresh M, and Karve Abhijeet C, "A study of Apache Kafka in bigdata stream processing," *International conference on information, communication,engineering and Technology*, August 2018.

[81] Abdul Ghaffar Shoro and Tariq Rahim Soomro, "Bigdata Analytics: Apache Spark Perspective," *Global Journal of Computer Science and Technology: Software and Data Engineering*, vol. 15, no. 1, 2015.

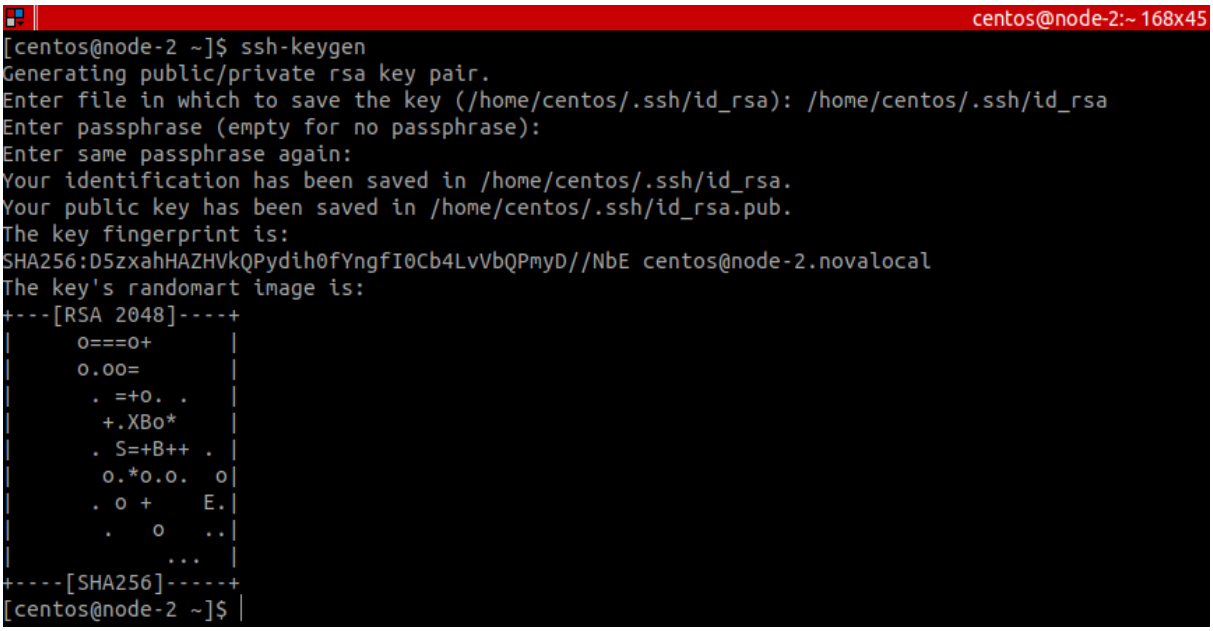
[82] Weixi Li, "Automatic Log Analysis using Machine Learning Awsome Automatic Log analysis Version 2.0," Uppsala University, Uppsala, Msc Thesis 2013.

[83] Tutorialspoint. (2020, April) Tutorialspoint. [Online]. https://www.tutorialspoint.com/elasticsearch/elasticsearch_tutorial.pdf

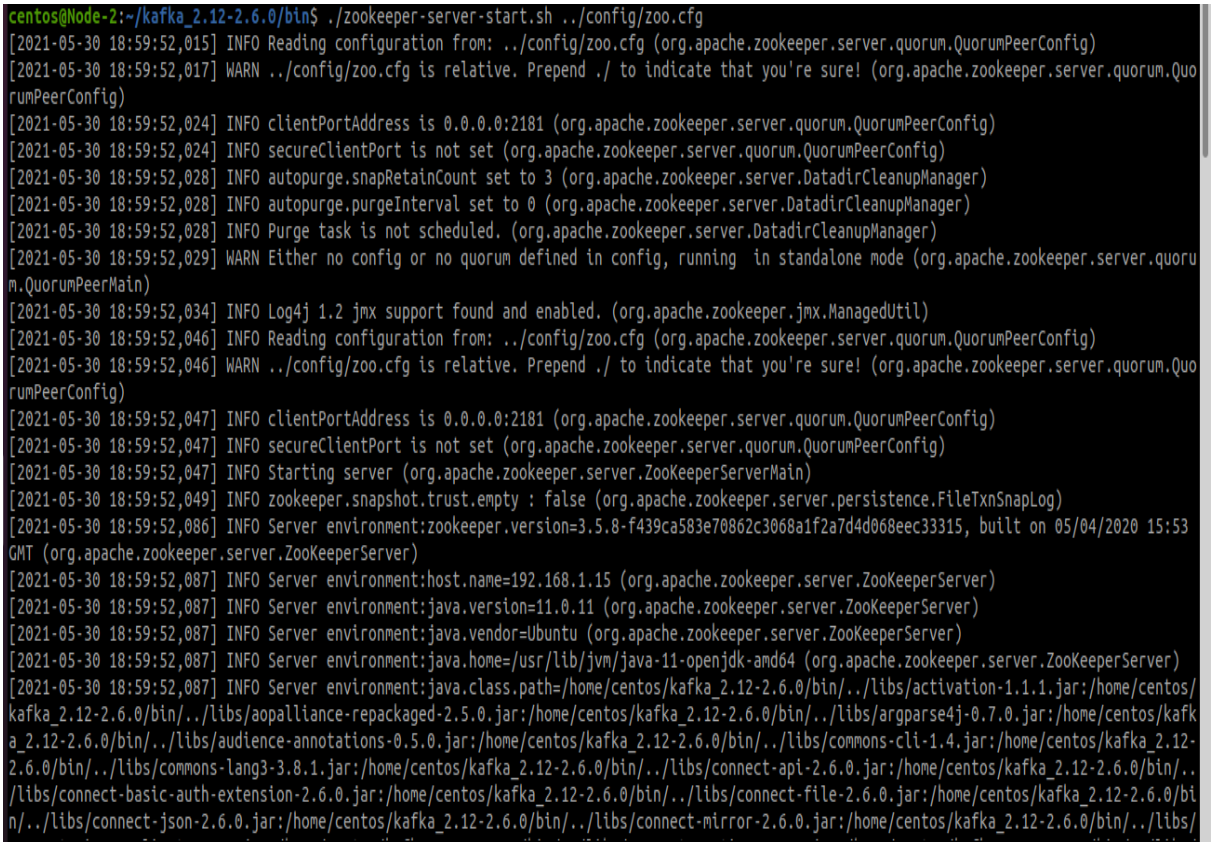
[84] Tutorialspoint. (2020, April) Tutorialspoint. [Online]. https://www.tutorialspoint.com/kibana/kibana_tutorial.pdf

Appendix

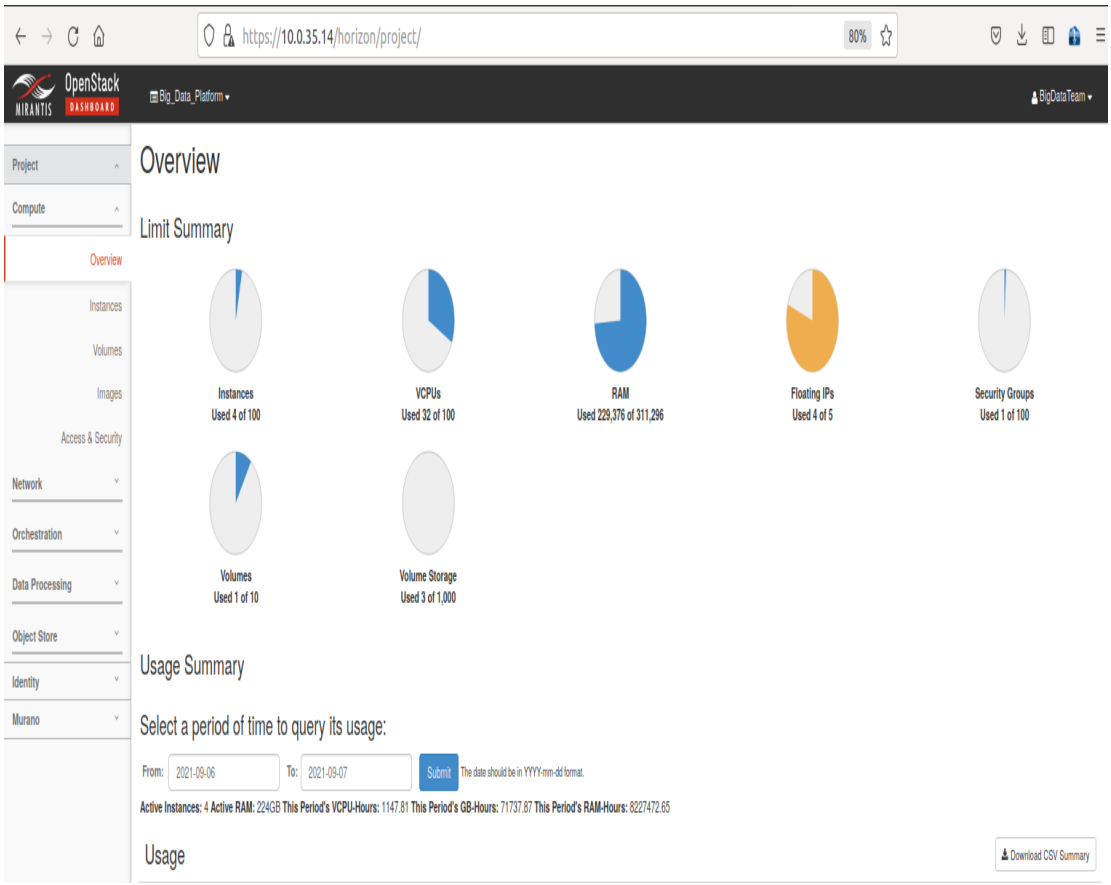
1. Generating SSL keys for passwordless connection



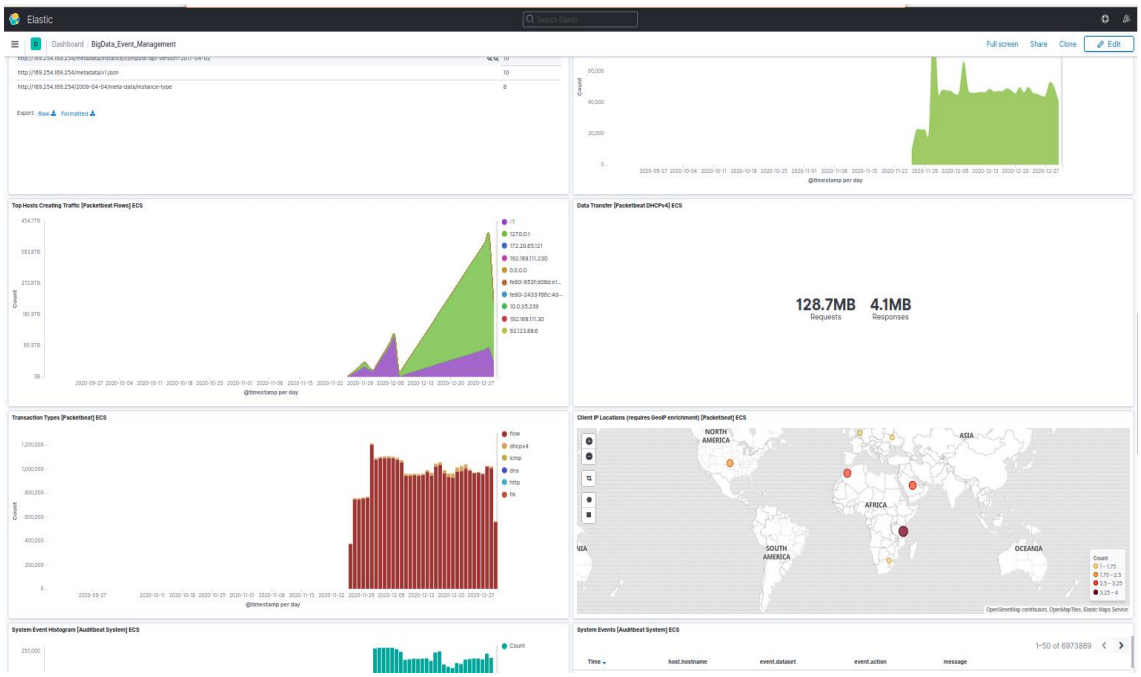
2. Zookeeper startup screen



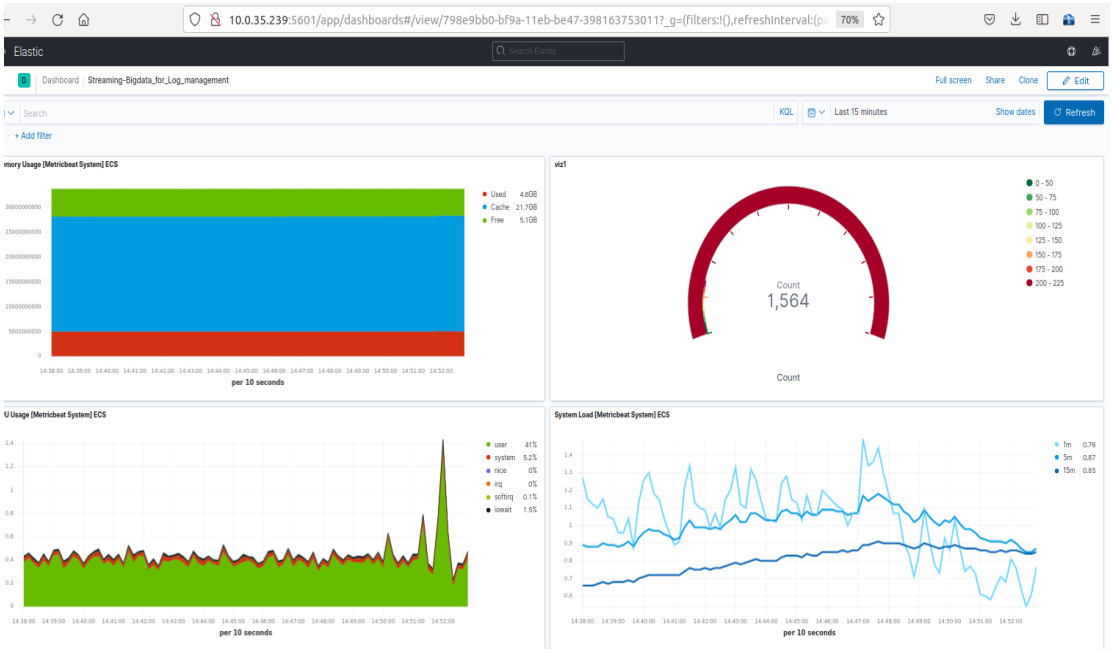
3. Openstack cloud nodes statistics summary page



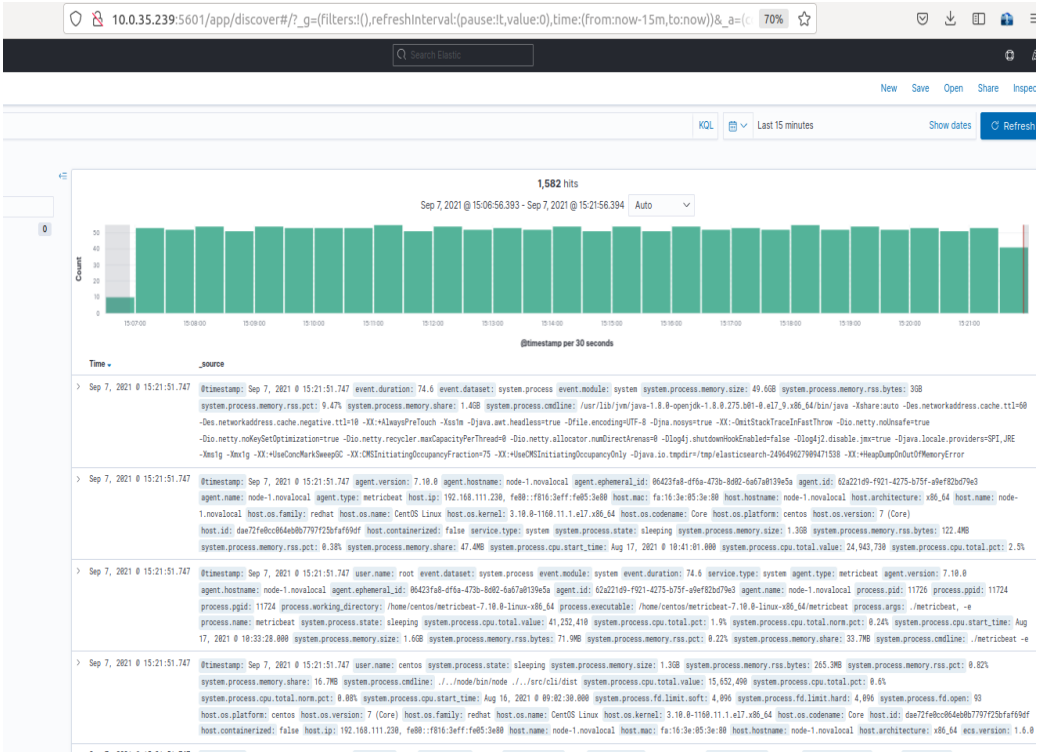
4. Kibana Visualization(main Window)



5. Kibana dashboard front view



5.Kibana Discover Tab to view Raw Metricbeat data



6.Kibana Discover tab single entry packet beat geoip Information