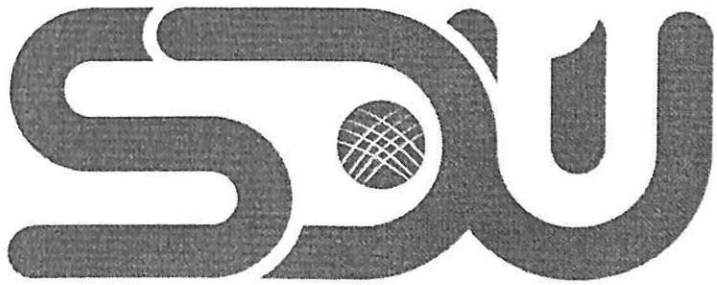
Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University



Yerkebulan Mukhamedkali

Log analysis for cyber attack detection and classification using Apache Spark and Python

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computer Science
(degree code: 7M06102)
Department of Computer Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Phd, Assis.Prof. Meiram Murzabulatov**

Kaskelen, 2022

Suleyman Demirel University

Faculty of Engineering and Natural Sciences

Department of Computer Science

Dean of Faculty

Associate Professor, PhD Zhamanov A.

«20» 06 2022

Topic of the thesis:

Log analysis for cyber attack detection and classification using Apache Spark and Python

---

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science", SDU, 2020-2022

Head of Department Assoc.Prof. PhD Cemil Turan

Academic Supervisor Assis.Prof. PhD Meiram Murzabulatov

Master's student Yerkebulan Mukhamedkali

Kaskelen, 2022

Abstract

The IT infrastructure of lots of companies have increased from several to hundreds and thousands of servers over the past decades, in turn the digital footprints have risen. Digital footprints consist of access logs that include information about specific events, for instance, IP address of the user, timestamp when the event occurred and hostile activity that can affect the network. IT companies mostly used the architecture of Apache Hadoop that is based on MapReduce to analyze the access log files of the companies. These access log files contain confidential information which is related to security of servers, as for it these companies are seeking for the architecture which can analyze the access logs in real time. In order to transcend these limitations of Apache Hadoop, in this paper are given efficient architectures such as streaming based Apache Spark which can analyze both batch-based and real time data.

Аңдатпа

Соңғы онжылдықтарда көптеген компаниялардың it-инфрақұрылымы бірнеше жүздеген және мыңдаған серверлерге дейін өсті, бұл өз кезегінде сандық қатысуды арттырды. Сандық іздер белгілі бір оқиғалар туралы ақпаратты қамтитын кіру журналдарынан тұрады, мысалы, пайдаланушының IP мекенжайы, оқиға болған кездегі уақыт белгісі және желіге әсер етуі мүмкін дұшпандық әрекет. IT компаниялары негізінен MapReduce негізіндегі Apache Hadoop архитектурасын компаниялардың кіру журналдарының файлдарын талдау үшін қолданды. Бұл кіру журналдарының файлдарында серверлердің қауіпсіздігіне қатысты құпия ақпарат бар, өйткені бұл компаниялар нақты уақыт режимінде кіру журналдарын талдай алатын архитектураны іздейді. Осы Apache Hadoop шектеулерін жеңу үшін, бұл мақалада пакеттік деректерді де, нақты уақыттағы деректерді де талдай алатын Apache Spark сияқты тиімді архитектуралар келтірілген.

Аннотация

За последние десятилетия ИТ-инфраструктура многих компаний увеличилась с нескольких до сотен и тысяч серверов, что, в свою очередь, привело к увеличению цифрового присутствия. Цифровые следы состоят из журналов доступа, которые содержат информацию о конкретных событиях, например, IP-адрес пользователя, временную метку, когда произошло событие, и враждебную активность, которая может повлиять на сеть. ИТ-компании в основном использовали архитектуру Apache Hadoop, основанную на MapReduce, для анализа файлов журналов доступа компаний. Эти файлы журналов доступа содержат конфиденциальную информацию, которая связана с безопасностью серверов, поскольку для этого эти компании ищут архитектуру, которая может анализировать журналы доступа в режиме реального времени. Чтобы преодолеть эти ограничения Apache Hadoop, в этой статье приведены эффективные архитектуры, такие как Apache Spark на основе потоковой передачи, которые могут анализировать как пакетные данные, так и данные в реальном времени.

Acknowledgements

I would like to say thank you to my supervisors Abdinurova Nazgul and Orynbekova Kamila for support throughout my Master's degree.Also I would like to thank my university for this opportunity.Thanks to the Ministry of Education for giving me the opportunity to study within the state grant.

Dedication

This dissertation is dedicated to my parents and friends whose support gave me motivation to not give up.

Contents

Nomenclature

API Application Programming Interface

DAG Direct Acyclic Graph

DDoS Distributed Denial-of-Service

DPC Data Processing Centers

DStream discretized stream

HDFS Hadoop Distributed File System

IP Internet Protocol

RDD Resilient Distributed Datasets

SQL Structured Query Language

URL Uniform Resource Location

List of Figures

List of Tables

1. Introduction

Since the development of the Internet, humanity has stopped using real information data that is transmitted in a classical way, that is, printed or written on paper, which has increased the number of users, which, consequently, has increased the amount of data transmitted between users. Because of this, the number of both intentional and unintentional cyber attacks has increased dramatically.

With the rapid development of information technology, users' expectations of the product are increasing. The increased expectations of users generate further development of information technologies. As a result, products inevitably become more complex, striving to satisfy the emerging desires and requirements of users. This means that the requirements for the corresponding software systems are increasing and becoming more complex. Developers are faced with the need to find and apply solutions that will work reliably in conditions of high load, large data flows, while allowing them to effectively develop new functionality and support existing ones.

One of the answers to the question of the device of such systems has become a microservice architecture, when a single monolithic application is divided into a large number of smaller parts that can be developed by independent and isolated teams, as well as independently deployed and, consequently, scaled. However, another question that has to be answered is the question of who owns the data and how to manage it. Disconnecting related services using an event stream turns out to be a fairly effective solution that shows itself well in practice in highly loaded systems. And such a decision is quite natural. Users interact in some way with the system by clicking on buttons, or, say, by clicking on links. All this is an example of events occurring in real life, and it turns out that these and similar events are well transferred to the concept of events in the context of software systems.

There are millions and millions of different sites and servers on the Internet that record all the actions and requests of users that remain digital fingerprints forever. These fingerprints consist entirely of Internet usage data in general and are recorded as access logs on various web servers. These access logs store a lot of important data, such as the user's Internet Protocol (IP) address, time and date, how many actions were performed, how much time it took, and in general general information about these actions, etc. These logs help to track the number of users who visited the site and the amount of time they spent on the website. These activity logs also help identify possible cyber security threats and anomalies in order to use this data to protect users and their privacy. This is all an advantage as it all increases the level of customer confidence.

Every event and logs should be analyzed, it is not just a part of a good security system of every company's infrastructure in IT, it is also a crucial part of standards in companies' security compliance.[5]

1.1 Problem statement

With the development of technology and the use of the Internet and Internet applications has led to an increase of internet users, which in turn has led to an increase in the amount of data processed on web servers. Corporations/companies spend a sufficient amount of their budget on quarterly data analysis, as the size of this data grows year after year and is considered a category of big data. Classic frameworks such as Apache Spark and Hadoop provide a good opportunity to process large amounts of data independently using the MapReduce paradigm.

Although Hadoop processes data efficiently, it cannot process them in real time, it does not reflect the flow of information, that is, it works with ready-made data. The main problem is that while data or information about the event is being collected, their relevance and usefulness are lost.

There are also frameworks like Apache Spark that have the ability to process data in a stream as efficiently as static data as soon as the data is available. Real-time data analysis reduces the time for the return itself, since the data is processed as it arrives and does not wait until the data is fully available.

There are many open source enterprise frameworks, such as Apache Spark, which can process both batch data and in real time, Hadoop, which analyzes only

batch data. But, unfortunately, there is no single platform that is capable of processing both types of data. The aim of the dissertation is to analyze logs using Apache Spark and Python, and compare the results with Hadoop, also paying attention to their viability/ability to process batch and streaming data.

1.2 Why are logs important?

Software applications, network devices, internet, operating systems and other hardware systems generate logs. These logs store information on files, disks or other applications.

These logs are important for hardware systems as logs can tell what event is happening and they can tell the actions of the system. All of the actions and processes that are happening and running on our systems create logs. The main issue is that these processes and files arise in different formats and different systems. The data from logs should be in meaningful and centralized form in order to be understood by humans and be decrypted in easy ways by systems of machine learning. The logs data are collected by different sources and they are connected to suitable patterns and trends.

All of the analysts should be aware and sure that log data stores all of the necessary information and that information can be explained by their context. This context can be different in every system. For instance, some of the systems acn use "significant", and other ones can use "alerts". When the formats of logs are written in the same template then analysis is simplified and easy to read, and there are less chances to be confused while reading these logs. It is better to store the logs in one centralized location because it will be hard to collect them then from different places. For instance, if there is an error and if the logs are placed in a shuffled way you will have to check every one of them. Even if you have an excellent tool, you can spend a lot of time trying to find out what is wrong. After the logs are cleaned, aggregated, normalized - all the anomalies can be easily identified with the help of analysis. [1]

1.3 Research questions

This dissertation tries to find answers to the following questions:

1. How to detect cyber attacks with access logs?

2. How to analyze logs using Apache Spark and Python?

3. What is the difference in analyzing data with Apache Spark and with Hadoop?

1.4 Objectives

1. To explain how to detect cyber attacks with the help of access logs

2. To identify the difference between analyzing the data using batch-based and streaming-based tools

2. Literature review

Software is deployed on multiple servers or on cloud servers (now companies use a combination of these two servers) to distribute the load so that when one server falls, every one does not fall in a row. This is done to increase availability and speed up delivery. These servers are located on several data processing centers (DPC) . These servers record all user actions and store them in the form of files called logs[12].

The data of these logs are usually recorded in different formats since their sources are located in different systems. Some of the sources include the server access log, proxy server log, browser log, or any other source of data from users that come from different web servers. Even if the log sources are in different formats, the basic information contained in the log file is common to most of them. According to Verma and Bhatia [14], it mainly consists of various components such as the IP address, the date and time when the access was made, the method by which the data was requested, the Uniform Resource Location (URL) of the available page, etc. In order to extract certain data from a file, it is necessary to perform certain actions such as clearing the data and reducing the amount of data.

The speed of creating log files varies between a few terabytes to a petabyte per day. Between the types of logs that are recorded in the web servers of different applications, access logs have information that relates to the navigation behavior of the user. To determine viewing patterns through user navigation behavior, Mavridis and Karatsa [9] implemented the MapReduce algorithm. In addition to navigational behavior, MapReduce is also used to identify possible system threats and problems based on access logs.

The Hadoop Distributed File System (HDFS) is considered to be the basis of the Hadoop platform, which increases the speed of data input/output, but not

performance. The data first of all goes to the HDFS itself and then only Hadoop in order to do/run any tasks or calculations. This slows down the performance of the Hadoop itself in analyzing data flows. A certain amount of time will be required to run a particular calculation in MapReduce and this time may take from two minutes to several hours until all calculations are completed. The main part of the application can analyze the data immediately as they arrive in real time. These applications should grasp on the fly and process data immediately despite the fact that the previous data has not been processed to the end, in short, these applications should not slow down. One such example may be the detection of anomalies or problems of incoming data in real time.

Shahrivari's [13] work describes the great advantages of Spark over the Hadoop platform regarding the speed of the hard disk and the delay period. Spark is superior to Hadoop in providing an in-memory computing environment that uses distributed main memory to process any kind of data in real time. While Spark computes data over 10 GB/s, Hadoop computes at a speed of 200 mb/s. Spark allows nanosecond delays when computing in memory and millisecond delays when working in local disks for computing.

Wingerath, Wolfram, etc. [15] noted that Spark already uses the streaming data processing paradigm, while Hadoop uses batch data processing, to increase speed and performance. Next, Spark divides the data stream into micro packages, which in turn turn into Resilient Distributed Datasets (RDD) . In the stream, words and data come in order and are processed synchronously, while in RDD everything is done asynchronously. Thanks to this paradigm, Spark increases the speed of data processing on a streaming basis than Hadoop itself on the basis of batch data processing. Spark Application Programming Interface (API) provides the possibility of RDDs processing through actions and transformation.[15]

With the ever-increasing scale and complexity of modern systems, the volume of journals is growing rapidly. Therefore, the traditional way of analyzing logs, which relies heavily on manual verification, has become a time-consuming and error-prone task. To solve this problem, many attempts have been made recently to automate log analysis using data mining techniques. However, raw log messages are usually not structured, as developers are allowed to write log messages in a free form for convenience and flexibility. To apply automatic extraction of unstructured logs, the first step is to perform a syntactic analysis, whereby un-

structured raw messages can be converted into a sequence of structured events.

3. Concepts

3.1 DDoS

Distributed Denial of Service (DDoS) attacks is a hacker attack aimed at causing problems with opening a website. Denial of service attacks are mainly used to disable public sites. Usually these are the sites of large government agencies, banking and commercial structures. If the attack is not hidden, it is easy to identify it by a number of signs:

- Unavailability of the entire site.
- Unavailability of the site partially (certain pages).
- Incorrect page response.
- Page loading is too long.
- Incorrect operation of site functions, objects on the page.

Shadow DDoS attacks are determined by qualitative and quantitative analysis of incoming traffic, which is organized on the network side.

3.2 Log Files

A log file is a document with a sequential record of all events occurring on a website or remote server. The term comes from the English Log - a log with notches, which in the Middle Ages was used as a kind of event log[2].

Since absolutely everything that happens is recorded in the log, the study of the log allows you to use the accumulated data for various purposes important for the promotion and maintenance of sites.

In case of technical problems, unavailability of the site, virus infection, hacker attacks and distributed denial-of-service (DDoS), the resource administrator can find out the reason according to the information recorded in the log, which will

facilitate and accelerate the elimination of undesirable phenomena.

In general, there are four main types of logs, these are Access Log files, Error Log files, Agent Log files and Referrer Log Files, but mainly in this paper the researcher will use only one type of it - Access Log Files.

Access Log Files is the total list of all requests processed by the server.

3.2.1 Access Log Files

Access log files are types of log files that store the information about visits to certain websites. It also contains the information about the IP address of the user, date and time when the website was visited, the device that user is connected from, etc.

Log Format "DateTime, Ip Address, User-Agent"

1) DateTime format - dd/mm/yyyy hh/mm/ss - this is the information about the time and date when the user visited the website

2) Ip address requested from - the user's IP address when the request was made

3) User Agent - Information about client device requested from - the user's device information, whether it is laptop, computer or personal device

15-11-2022 00:00:00, 10.45.76.198, Mozilla/5.0 (iPhone; CPU iPhone OS 10_2 like Mac OS X) AppleWebKit/602.3.12 (KHTML, like Gecko) Version/10.0 Mobile/14C92 Safari/602.1 - there is a sample of access logs that were analyzed during the experiment. As it is shown all the information that was described above is given and clear enough to read and understand.

3.3 Big Data

Today, one of the actively developing areas in the field of information technology is Big Data technology. In recent years, Big Data has been a recognized sign of economic and technological development[11].

Research by the consulting company Gartner predicts that Big Data technology will have a significant impact on information technology in manufacturing, healthcare, trade, public administration and other industries that use a large flow of information. In Kazakhstan for the period up to 2030 the research of Big Data

technology for the management of industrial and social processes, along with the processing of unstructured data, is recognized as a priority in the field of information technology.

If we give a brief definition, then Big Data is data that does not fit into the computer's RAM.

In fact, this definition means that the property of "being big" is not an independent property of the data, but depends on the characteristics of the system used to process them.

For example, it is difficult for an ordinary person to remember exactly what temperature was in our city every day for the past month. Thus, three dozen values may well be an example of Big Data. However, the person confidently reports "the past month was cold." This message carries information about the processed data: according to the interlocutor, the average temperature over the past month was lower than usual this month for several decades. Another example could be data about objects that theoretically carry important information, but have such a size that it is almost impossible not only to process or save this data, but even to collect it. Consider, for example, a data set containing the coordinates and velocities of molecules in an air column over the territory of an airport.

There is also metadata describing at what point the measurement was carried out and what kind of molecule it is. Such a data set carries information about weather conditions over the airport, including temperature, pressure, humidity, cloud cover, special weather conditions – a passing tornado or falling hail. On the other hand, for correct processing, the data for all molecules must be sufficiently complete and representative for statistical processing. As a result of such a thought experiment, we understand that in order to work effectively with big data, we need a data model that allows us to form methods of working with data.

The concept of "Big Data" in the field of information technology, which appeared recently, was introduced by Clifford Lynch. Here are the most common definitions. Big Data is:

- a group of technologies and methods for productive processing of dynamically growing amounts of data in information technology;

- a series of approaches, tools and processing methods structured and unstructured data of huge volumes and considerable variety for obtaining human-perceived results, effective in conditions of continuous growth, distribution across

numerous nodes of the computer network.

To work with Big Data, complex systems are used in which several components or layers can be distinguished. There are usually four levels of components of such systems: reception, collection, analysis of data and presentation of results. This division is largely conditional, since, on the one hand, each component in turn can be divided into sub-components, and on the other, some component functions can be redistributed depending on the task being solved and the software used, for example, data storage is allocated to a separate layer.

To work with Big Data, system developers create data models that are meaningfully related to the real world. The development of adequate data models is a complex analytical task performed by system architects and analysts. The data model allows you to create a mathematical model of the interactions of real-world objects and includes a description of the data structure, data manipulation methods and aspects of maintaining data integrity. Description of the development of data models is not the task of this manual.

3.3.1 Five Vs of Big Data

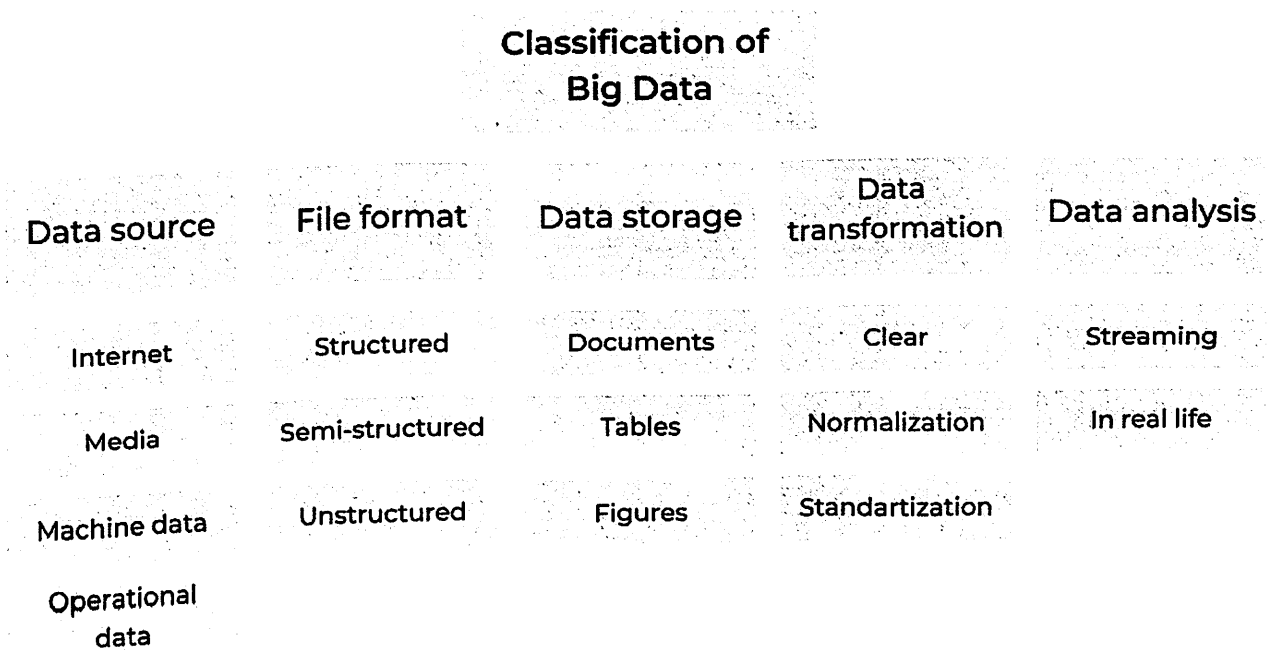


Figure 3.1: Classification of Big Data

Big data is classified depending on the type of data source (Internet or media sources, generated machine data), by file format (structured, semi-structured,

unstructured), by data storage method, etc. For clarity, Figure 3.1 shows the general classification of Big Data.

3.3.2 The main features of Big Data technology.

For the first time in 2001, the signs of the "Three V" were identified by a leading analyst of Gartner, a friend of Lani, namely diversity, volume, speed, which are the main characteristics of the technology[11]:

Variety - helps to effectively analyze data;

Volume - (the most important of the properties of the technology), i.e. the amount of generated data determines the value and potential of the research

Velocity - i.e. at what speed the data is generated and processed. In accordance with the requirements of Zikopoulos suggested adding 2 more features - veracity and value, thus obtaining "5V":

Value is a technology attribute that describes the economic effect that the technology provides to users.

Veracity is the quality of the collected data, which can vary significantly. The exact analysis depends on the reliability of the source data.

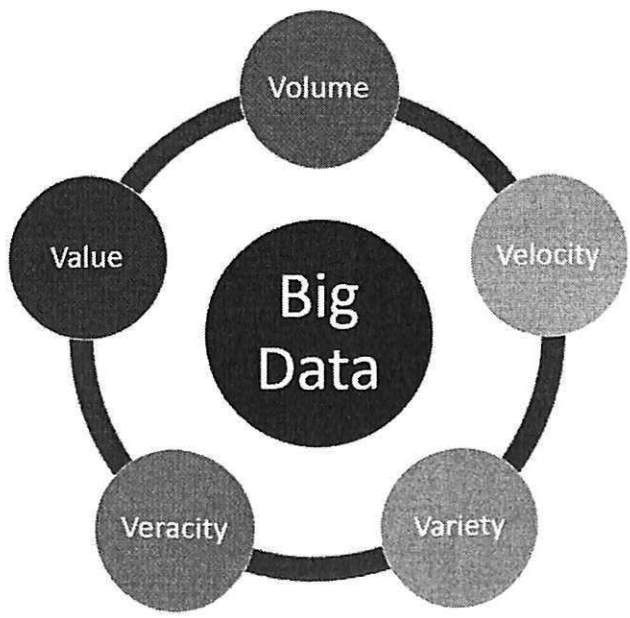


Figure 3.2: 5 V's of Big Data
Note. Hadoop Architecture. Retrieved April 16, 2022, from https://medium.com/analytics-vidhya/the-5-vs-of-big-data-2758bfcc51d

3.4 Apache Hadoop

Apache Hadoop is a package of utilities, libraries and frameworks, it is used to build systems that work with Big Data. It stores and processes data for uploading to other services. Hadoop has open source code written in Java. This means that users can work with it and modify it for free[16].

Hadoop is divided into clusters — a group of servers (nodes) that are used as a single resource. This makes it easier and faster to collect and process data. Division allows you to perform many elementary tasks on different cluster servers and produce the final result. This is necessary primarily for overloaded sites, for example Facebook.

There are several projects inside Hadoop that have turned into separate startups: Cloudera, MapReduce and Hortonworks. These projects are distributions or an installation package of a program that processes big data.
But Hadoop basically consists of two components: a distributed file system known as HDFS, and a MapReduce software model for processing this data.
Pros of "Hadoop":

- The big number of data can be stored and processed on Hadoop

As the technology and social media is developing, the number of data is increasing too. It is one of the important factor

- Its power of computing

The process of computing data in a huge amount on Hadoop is very quick. It depends on the number of nodes, more nodes = more power of computing

- Fault-tolerance

Hadoop has excellent fault-tolerance. In case the one node is getting non-functional, all the tasks go to the other node

- Cost

As it is open-source tool, it is free to use and install, and of course to store big data

- Horizontal scalability

It is easy for Hadoop to scale up just by adding extra nodes. The deep administration is not required.

- Encapsulation of implementation complexity.

"Hadoop" hides many complexities of distributed and multithreaded systems.

Frees the developer from worrying about system-level problems, such as waiting for data, organizing data transfer, data distribution, code delivery. Allows the developer to focus on application development and business logic implementation.

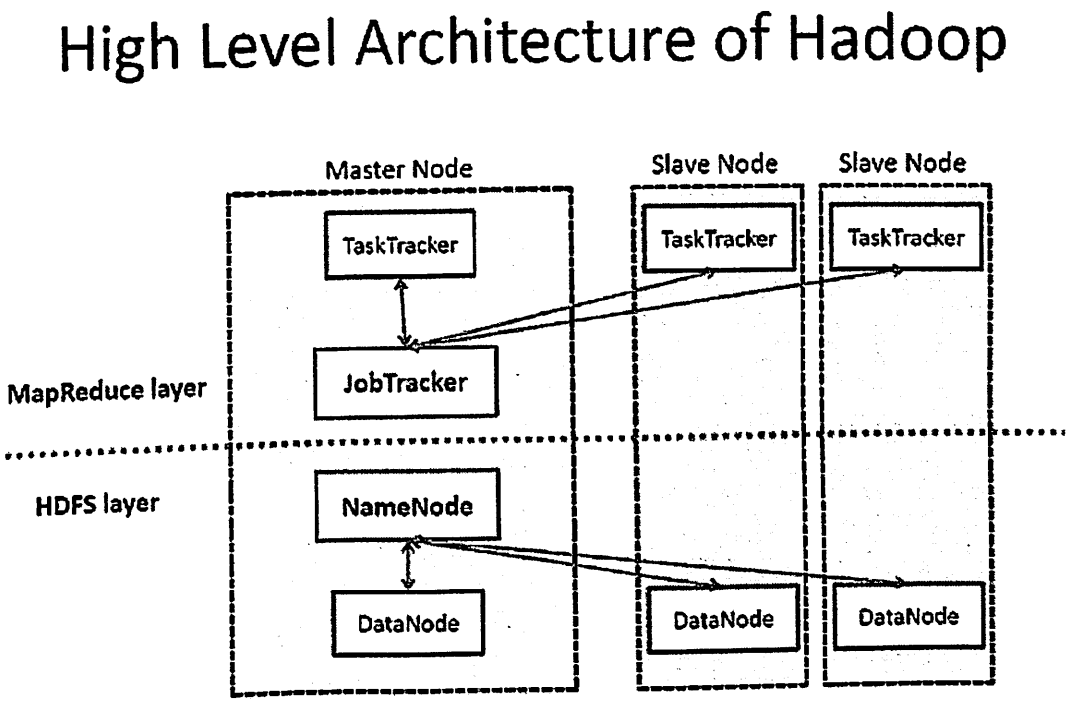


Figure 3.3: Hadoop Architecture
Note. Hadoop Architecture. Retrieved March 13, 2022, from https://www.projectpro.io/article/hadoop-architecture-explained-what-it-is-and-why-it-matters/317

3.4.1 Hadoop Distributed File System (HDFS)

Hadoop HDFS (Hadoop Distributed File System) is a hierarchical system for storing large files with streaming access. This means that HDFS makes it easy to find and duplicate data.

HDFS consists of a NameNode and a DataNode - a control node and a data server. NameNode is responsible for opening and closing files and manages access to directories and blogs of files. DataNode is a standard server where data is stored. It is responsible for writing and reading data and executes NameNode commands. A separate component is the client (user), which is granted access to the file system.[7]

HDFS Properties:

- Storage of very large files.
- Streaming data access: Following the concept of single write and multiple read will give maximum efficiency of data processing on HDFS.
- HDFS, like Hadoop in general, does not require expensive equipment with high reliability, since the system is designed to work on clusters consisting of standard machines and taking into account the high probability of failures of individual nodesHigh bandwidth in HDFS is preferable to access speed.

One of the main concepts of HDFS is the blockiness of stored data files (the standard block size is 64 MB). Blocks of a single file can be stored distributed, which is also used for replication, and block abstraction instead of file abstraction simplifies the storage subsystem (fixed size makes it easy to calculate the number of blocks that a disk can hold) and metadata organization. Data security in HDFS is achieved by replicating NameNode node metadata across multiple file systems and using Secondary NameNode to create checkpoints. High availability is provided by using two NameNode nodes in the "active/standby" configuration.To do this, both NameNodes must use a common high availability data store to store the change log.

DataNode nodes, in turn, should send reports to both NameNodes. The transition from the active NameNode node to the backup node is controlled by the so-called fault management controller.

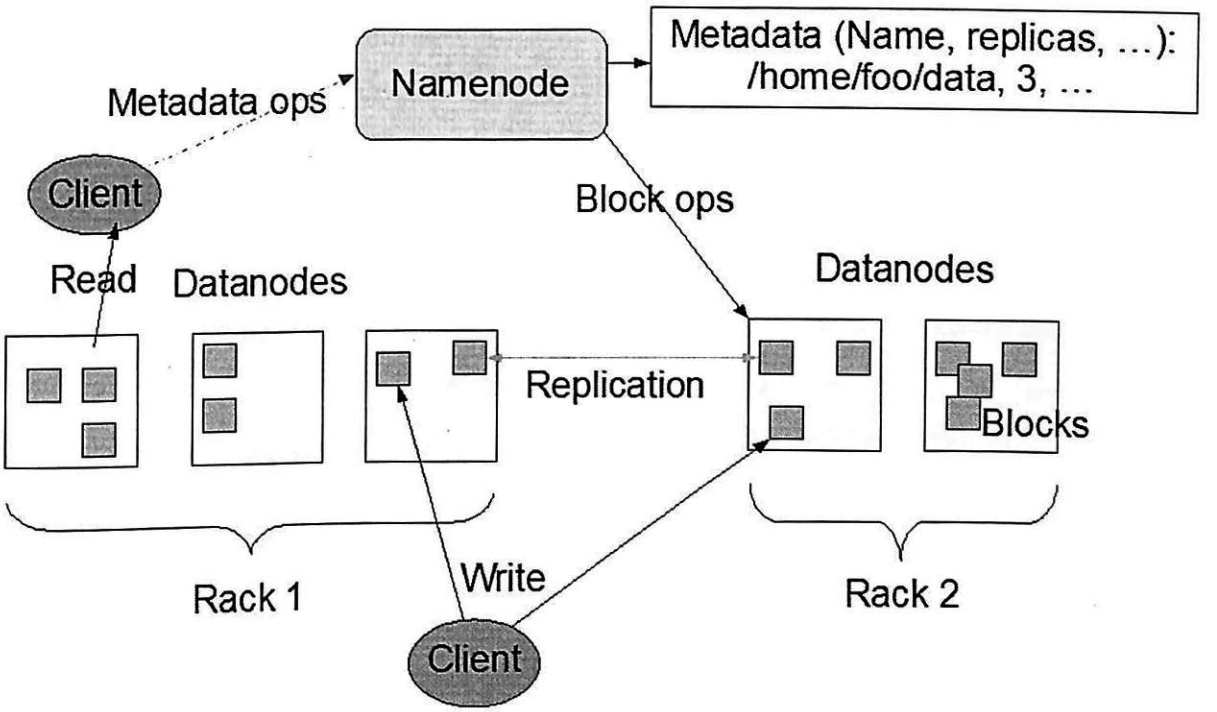


Figure 3.4: Architecture of HDFS
ote. Architecture of HDFS. Retrieved March 14, 2022, from https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

3.4.2 MapReduce

MapReduce is a programming model that was first used by Google to index its search operations. MapReduce is built on the principle of "master-subordinates". The main server in the system is the JobTracker server, which distributes tasks to subordinate nodes of the cluster and monitors their execution. The Map function groups, sorts, and filters multiple datasets. Reduce aggregated data to get the desired result. [4]

To better understand how MapReduce works, the example is given. Let's say you have a text and you want to count the frequency of each word. You can easily do this by storing each word and its frequency in the dictionary and looking up all the words in speech. However, imagine if you have a text for the entire Wikipedia (say a billion words) and we want to do the same.

Map takes a word and "emits" a <key-value> pair. In this case, key = word and value = 1 (we have only one instance of the word). For example, the phrase

"Big Data Analysis" will emit ("Big", 1), ("Data", 1) and ("Analysis", 1). Map is called for each word - only one billion of them. Now we have one billion key-value pairs. What are we doing about it?

That's where Reduce comes in. All key-value pairs with the same key are combined and fed into Reduce. Let's take "hello", for example, and say that you are given a certain text with a plural of this word. Then we have 500,000 key pairs that are passed in one Reduce call. Now all we have to do is iterate through 500,000 ("hello", 1) pairs and sum up all the values. It's much easier than going through a billion. Then Reduce "emits" another key-value pair; this time it is ("hello", 500,000), or a word, and the total sum of the values associated with that word. Repeat the Reduce call for every other word, and we have a frequency for every word.

Having distributed this huge amount of work across many computers, if any of these computers fails for any reason, we can simply restart a certain segment that he was working on.

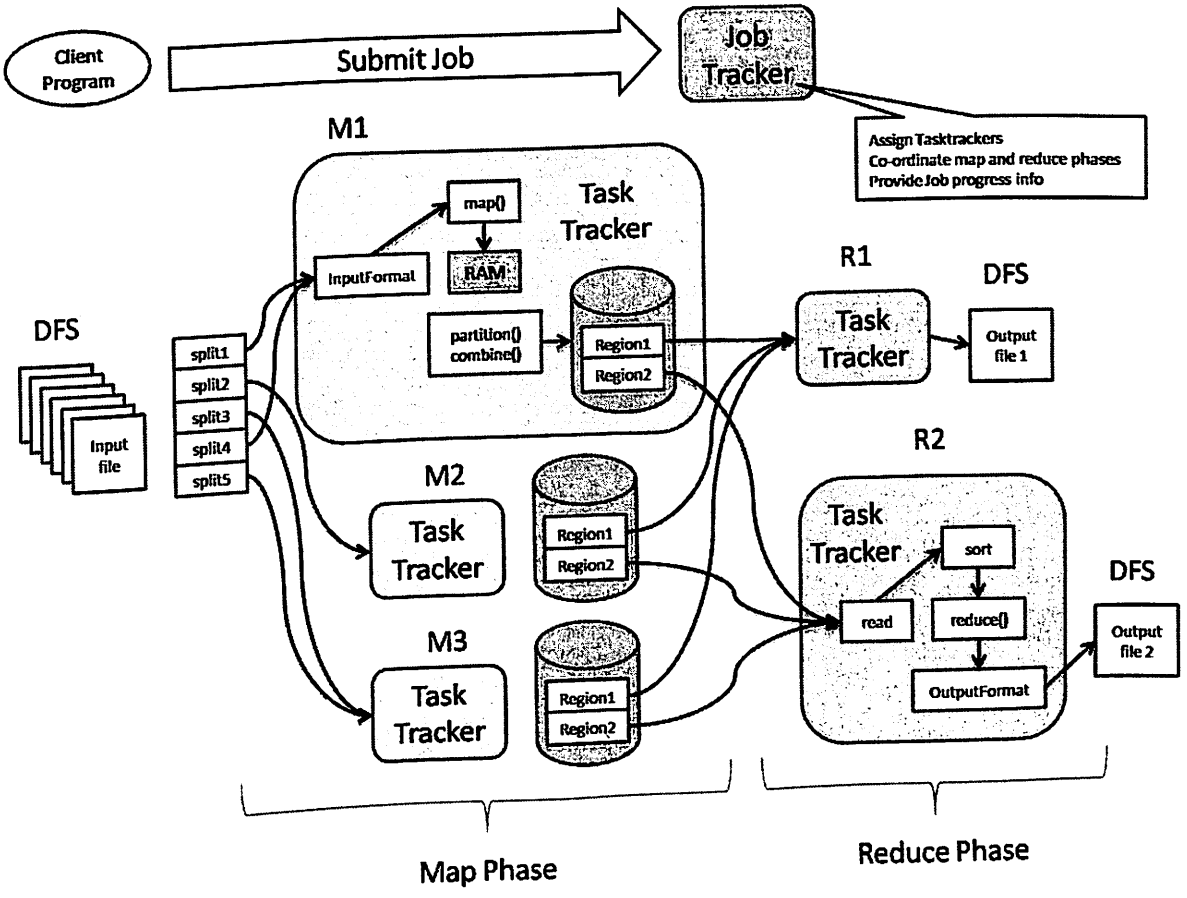


Figure 3.5: MapReduce Framework

Note. MapReduce Framework. Retrieved March 14, 2022, from http://a4academics.com/images/hadoop/Hadoop-Mapreduce-Architecture.png

3.5 Apache Spark

Apache Spark or simply Spark is a framework (software that combines ready-made components of a large software project) that is used for parallel processing of unstructured or weakly structured data.

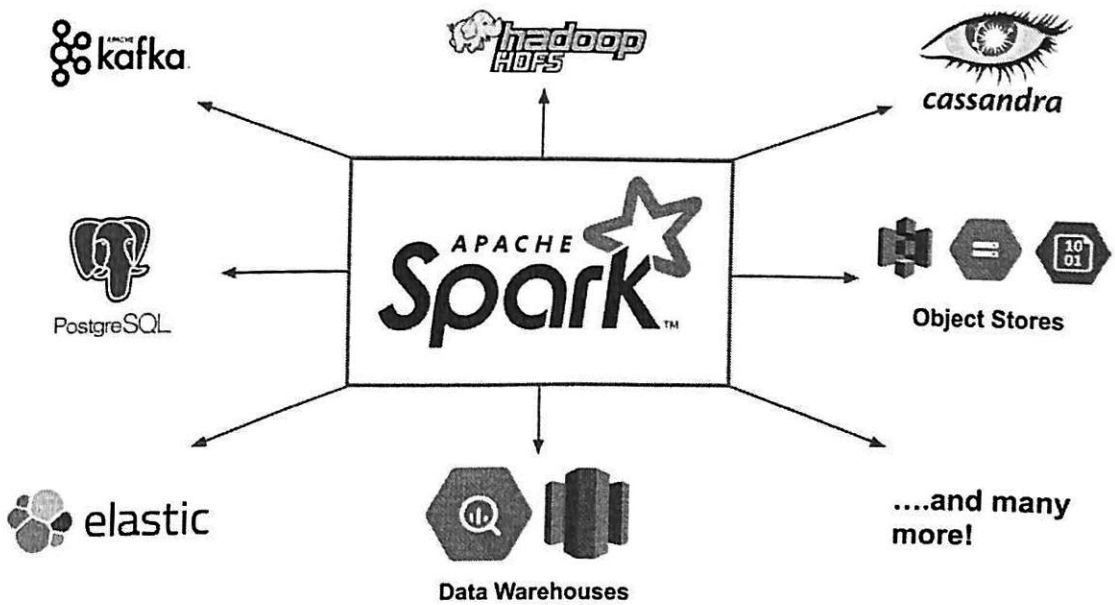


Figure 3.6: Introduction to Spark
Note.Introduction to Spark. Retrieved April 13, 2022, from https://www.datamechanics.co/apache-spark

Apache Spark initially was as a part of Apache Hadoop system, but now Spark is a separate tool for analyzing big data. There is a claim that Apache Spark is much faster than Apache Hadoop in big data analysis, 10 times faster in processing on disk and 100 times faster in processing in memory[3].

The framework was created in order to cover as wide a range of workloads as possible, which previously required the creation of separate distributed systems, including batch processing applications, cyclic algorithms, interactive queries and streaming processing. By supporting all these types of tasks with a single mechanism, Spark simplifies and reduces the cost of combining different types of processing that often need to be performed in a single data processing pipeline. In addition, it reduces the maintenance burden by maintaining separate tools.

For example, if you need to process data on the annual sales of one store, then one computer and Python code will be enough for a programmer to make a calculation. But if data is processed from thousands of stores from several

countries, and they arrive in real time, contain omissions, repetitions, errors, then it is worth using the power of several computers and Spark. A group of computers simultaneously processing data is called a cluster, so Spark is also called a framework for cluster computing.

Spark supports Scala, Java, Python, R and Structured Query Language (SQL) programming languages[10]. Apache Spark works with the help of RDD, in the next chapter there is more detailed information about RDD and the process of work. At first, only the first two were popular, since the framework was written in Scala, and part of the code was later written in Java. With the growth of the Python community, this language has also become more actively used, although updates and new features are primarily available to Scala developers. The R language is most rarely used to work with Spark.

3.5.1 Spark Components

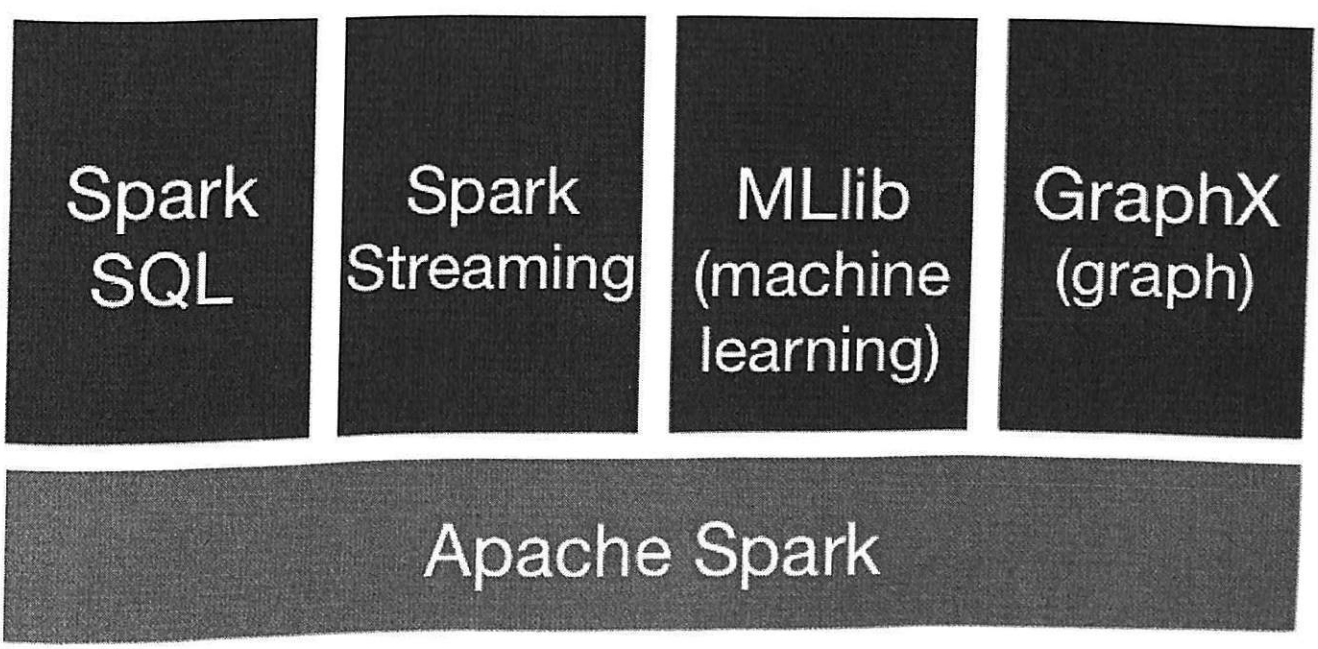


Figure 3.7: Components of Spark
Note. Components of Spark. Retrieved April 13, 2022, from https://medium.com/expedia-group-tech/start-your-journey-with-apache-spark-part-1-3575b20ee088

The Spark structure includes a data processing core and a set of extensions:

Spark Core or core is just an engine. It is responsible for data storage, memory management, distribution and task tracking in the cluster.

MLlib - a set of libraries for machine learning;

The SQL component that is responsible for querying data;

GraphX is a module for working with graphs (abstract representations of connections between a set of objects);

Streaming is a tool for processing Big Data streams in real time.

3.5.2 Resilient Distributed Datasets (RDD)

Resilient Distributed Datasets (RDD) is a simple, immutable, distributed collection of objects in the Apache Spark framework. RDS is a distributed data set that is divided into many parts, processed by various nodes in the cluster.

RDD has the following properties[17]:

Immutability and partitioning – RDD consists of a set of records that are divided into sections. A partition is a unit of concurrency in the RDD. Each partition is a logical subdivision of data, which is immutable and stored on a separate node in the cluster.

The use of common operations (coarse-grained operations) that are able to manipulate all data at the same time (for example, filter or grouping).

Fault tolerance: All transformations on the RDD sets are performed in a distributed environment with replication support (copying data between nodes), and each transformation is registered by each individual node in the cluster. Therefore, if one node fails, the data can be restored using any other working node.

Lazy calculations: Apache Spark performs the necessary transformations on the RDD only once at the time of their creation. This significantly reduces the total execution time of all operations and speeds up work on data.

Persistence: users can choose a convenient format for storing RDD (for example, in memory or on disk in a file).

3.6 Spark Streaming

Spark Streaming is a Spark component for processing streaming data. Examples of sources of such data can be log files populated by active web servers, or message queues sent by users of web services. Spark Streaming has an API for managing data flows, which closely corresponds to the RDD model supported by the Spark Core component, which facilitates the study of the project itself and various data

processing applications stored in memory, on disk or arriving in real time. The Spark Streaming component application interface was developed with the aim of providing the same reliability, throughput and scalability as Spark Core.

Spark Streaming is built using a micro-packet architecture, when the main data stream consists of a sequence of small data packets. At the same time, new packages are created regularly at a specified time interval. At the beginning of each time interval, a new packet is created, which includes any data received during this interval. At the end of the interval, the packet stops increasing. The size of the interval is determined by such a parameter as the batch interval. Usually this interval is selected in the range from 500 milliseconds to several seconds. Each packet forms a collection of RDD and is processed by a Spark job, which creates another set of RDD at this point in time. The results of processing the packet are then transmitted to external systems for analysis and further processing[8].

As a software abstraction, Spark Streaming uses a discretized stream (DStream) , which is a sequence of RDD sets, where each individual set of RDD corresponds to one time interval. Such streams can be created from external data sources or by applying various transformations to other streams of the same type. For each data source, the Spark Streaming module automatically starts receivers, which are used to collect data from sources and save them to RDD sets.

Spark Streaming supports a mechanism for copying data at checkpoints, which is able to save the state of the program in the file system, for example, HDFS in Apache Hadoop or Amazon S3. Such copying is usually performed every 5-10 packets. Thanks to this, Spark Streaming provides the ability to restore data in the event of a failure - calculations will continue from the last checkpoint.

Taking into account the architecture features described above, we list the main advantages of Spark Streaming for a developer of Big Data applications:

• copying data at control points limits the amount of calculations in case of failure;

• fault tolerance of working nodes due to simultaneous copying of data from external sources to them;

• fault tolerance of data receivers – in case of failure of one of the receivers, Spark Streaming will launch receivers on other nodes of the cluster;

• strictly one-time message delivery semantics (exactly-once) - if the working node fails at the time of processing, the final result of the conversion will not

Figure 3.8: Spark Streaming Input/output Data Sources
Note. Spark Streaming Input/output Data Sources. Retrieved March 16, 2022, from https://spark.apache.org/docs/latest/streaming-programming-guide.html

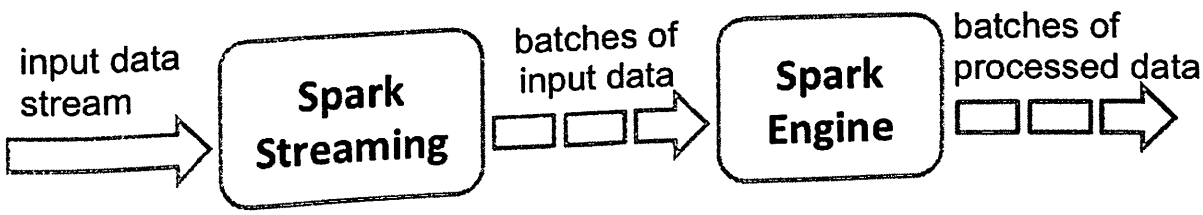


Figure 3.9: Spark Streaming Flow
Note. Spark Streaming Flow. Retrieved March 16, 2022, from https://spark.apache.org/docs/latest/streaming-programming-guide.html

change.

3.7 Apache Kafka

Kafka Apache is a distributed real-time messaging system between server applications. Due to its high throughput, scalability and reliability, it is used in companies working with large amounts of data. It is written in Java and Scala languages.Kafka Apache is an effective tool for organizing the work of server projects of any level. Due to its flexibility, scalability and fault tolerance, it is used in various areas of the IT industry, from video streaming services to Big Data analytics[6].

Apache Kafka is a message passing system operating on the publisher—subscriber model and designed to solve problems that arise during client-server interactions

in architectures with a large number of connections and dependencies. Kafka is designed internally in such a way that it allows you to store messages for as long as you want, which makes it possible to re-reproduce messages, the use cases of which will be described later.

Initially, Kafka began to be developed by LinkedIn when they directly encountered the problems described in the first chapter. At the beginning, LinkedIn had a monolithic application written in Java, which was subsequently divided into about a thousand separate services, which led to complex dependencies that led to instability, problems with the versions of the services and libraries used.

To solve these problems, it was decided to implement an asynchronous approach to the interaction of services, which allowed the transmission of trillions of messages per day within the company's services. To implement this approach, a system was developed, called Apache Kafka.

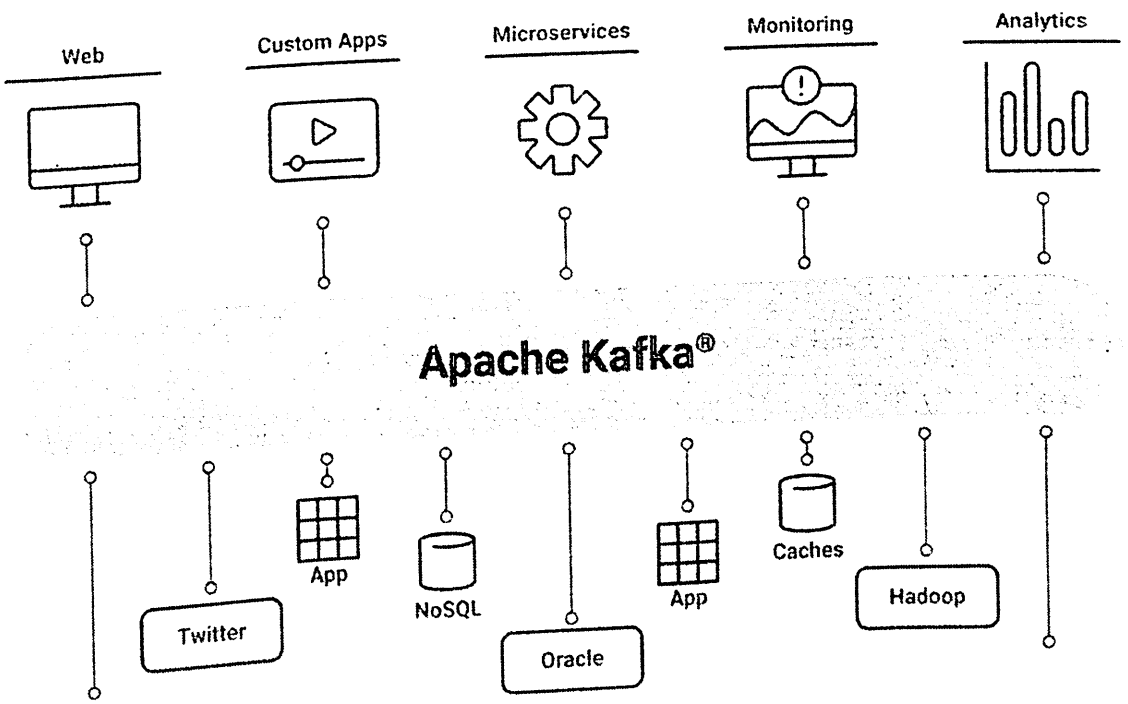


Figure 3.10: Apache Kafka
Note. Apache Kafka. Retrieved March 23, 2022, from https://docs.confluent.io/5.5.1/kafka/introduction.html

For communication of microservices. Kafka is the link between the individual functional modules of a large system. For example, it can be used to sign a microservice to other components for regular updates.

Data streaming. The high throughput of the system allows you to maintain continuous flows of information. Due to competent routing, Kafka not only re-

liably transmits data, but also allows you to perform various operations with them.

Event logging. Kafka stores data in a strictly organized structure in which it is always possible to track when an event occurred. The information is stored for a specified period of time, which can be used to unload the database or slow-running logging systems.

Briefly, the architecture of the message system can be described as follows:

distribution - individual nodes of the system are located on several hardware platforms (clusters). This provides it with high fault tolerance;

scalability - the system can be expanded by simply adding new nodes (message brokers).

Messages in Kafka are stored in topics. Publishers write messages to the appropriate topic, and subscribers subscribe to topics of interest to them, and begin to receive new messages from these topics. Rough analogues for a theme are a database table or folders in the file system. Each topic consists of message logs. Messages can only be written to the end of the log. Usually one topic consists of several magazines. And it is the logs that are the means of scaling.

The fundamental data structure in Kafka is a message log with sections, which can be repeated from the very beginning or from a certain point. The log is a data structure that contains elements called messages; messages can only be written to the end of the log. When a client wants to read messages from Kafka, he looks at what position was the last message he read, then sequentially reads new messages in the order in which they are located in the log.

Such an approach with storing data in the form of logs has interesting consequences. Both read and write operations are sequential operations. This makes pre-sampling, caching easier, and message bundling occurs in a completely natural way. All this increases the overall performance of the system at the output.

Also, the bundling and sequence of operations make it possible to store data for a long time. Many traditional message brokers are built using hash tables or B-trees, for which storing data for long periods is not the strongest side, for the reason that these data structures index data, and maintaining indexes for a large number of messages is time-consuming.

Among other consequences of the logs approach is that the accumulation of new messages without regular reading has no significant impact on system perfor-

mance, unlike traditional message brokers and queues. And in the end, all these factors make Kafka suitable for event storage.

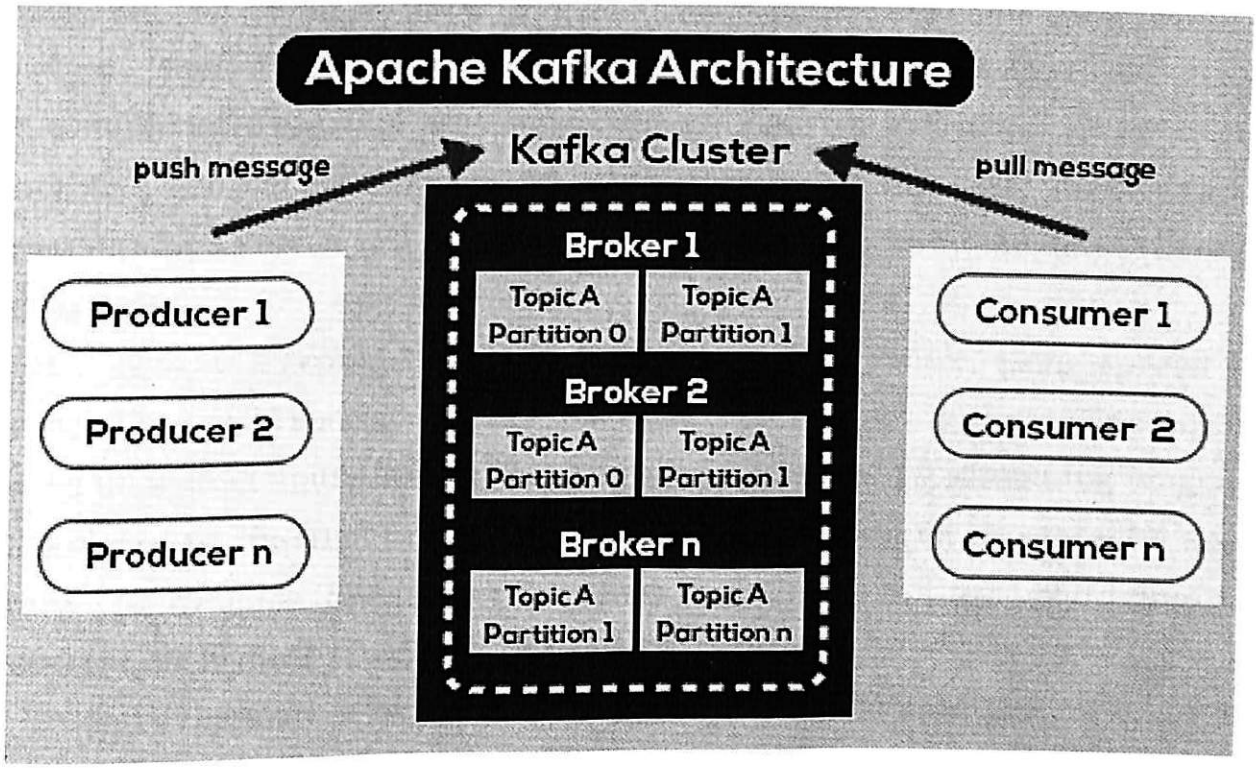


Figure 3.11: Apache Kafka Components and Architecture
Note. Apache Kafka. Retrieved March 23, 2022, from https://www.projectpro.io/article/apache-kafka-architecture-/442

3.7.1 Kafka Components

In the Kafka Apache architecture, the key concepts are:

- producer - an application or process that generates and sends data (publishes a message);
- consumer - an application or process that receives a message generated by the producer;
- a message - is a data package required to perform an operation (for example, authorization, purchase or subscription);
- broker - node (dispatcher) for transmitting messages from the producer process to the consumer application;
- topic - is a virtual message storage (log of records) of the same or similar content, from which the consumer application extracts the information it needs.

3.8 Python

A program can be represented as a set of sequential commands (algorithm) for an object (performer) that must execute them to achieve a certain goal. So you can conditionally program a person if you give him an instruction "how to cook pancakes", and he will begin to execute it clearly. In this case, the instruction (program) for a person will be written in the so-called natural language (Russian, English, etc.).

It is usually accepted to program computers, not people, using special languages. The use of special languages is caused by the fact that machines are not able to understand our (human) languages. A variety of programming languages are designed to "instruct" machines, which are characterized by syntactic unambiguity (for example, certain words cannot be swapped in them) and limitations (a strictly defined set of words and symbols).

To date, there are a huge number of different and similar programming languages. The reason for this phenomenon becomes clear if you imagine the number and variety of tasks that are currently being solved with the help of computer technology. Different tools (i.e. programming languages) are required to solve different tasks. Many programmers have tried in the past and are trying now to come up with their own programming language that has certain advantages. Although the vast majority of programmers currently spend a huge amount of time studying the already existing arsenal of tools. All the existing variety of languages can be conditionally classified according to different criteria. For example, by the type of tasks to be solved (system or application languages, languages for web development, etc.).

For this work the author decided to use the programming language called Python.

The Python programming language was created around 1991 by the Dutchman Guido van Rossum.

Python got its name from the name of the TV series, not the reptile. After Rossum developed the language, he posted it on the Internet, where a whole community of programmers joined in improving it.

After Rossum developed the language, he posted it on the Internet, where a whole community of programmers joined in improving it.

Python is being actively improved at the present time. New versions of it are often released. Official website http://python.org.

Python is a language with dynamic type control, in which names can represent values of various types during program execution. And indeed, the names used in the program are only labels for various quantities and objects. The assignment operator simply creates a relationship between the name and the value. This is one of the differences between this language, for example, from C, in which names are represented by objects with a constant size and placement in memory, where the results are located.

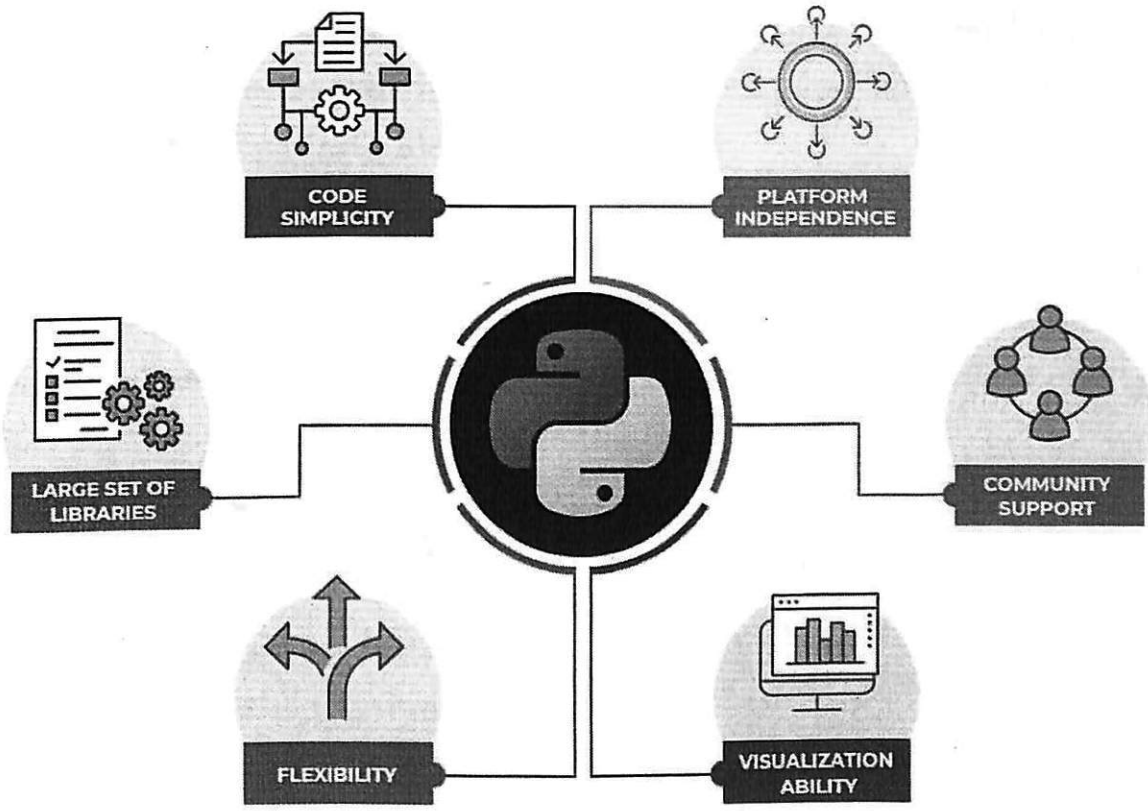


Figure 3.12: Introduction to Python
Note. How Python To Be A First Choice Of World. Retrieved May 3, 2022, from https://news4hackers.com/how-python-to-be-a-first-choice-of-world/

3.8.1 Features of Python

Python is an interpreted programming language: the source code is partially converted into machine code during execution by a special interpreter program.

Python is characterized by a clear syntax. It is quite easy to read the code in

this programming language, because there are few auxiliary elements in it, and the rules of the language force programmers to indent. It is clear that a well-designed text with a small number of distracting elements is easier to read and understand.

Python is a full–fledged, one might say universal, programming language.

It supports object-oriented programming (in fact, it was developed as an object-oriented language).

Basically, the interpreter executes commands line by line: you write a line, press Enter, the interpreter executes it, you observe the result.

This is very convenient when a person is just learning programming or testing some small part of the code. After all, if you work in a compiled language, you would first have to write code in the source programming language, then compile and then run the executable file for execution.

Despite the convenience of the interactive mode of operation when writing Python programs, it is usually necessary to save the source code for later use. In this case, files are prepared, which are then passed to the interpreter for execution. In relation to interpreted programming languages, the source code is often called a script. Python code files usually have the extension py.

3.8.2 Downloading Packages for Python

There are several ways for installing packages for Python.

Starting from installing from the source.

Being able to install the packages from the source is a good skill to learn, other ways are easier to use, but we will look at them later in this chapter. However there are some packages that are necessary to use this way for installing. To install the certain package from the source first of all we need to use command below:

```
nrg@Machine:~$ python3 setup.py install
```

Figure 3.13: Installing package from the source

If Python is not located in the path of your system, you will receive an error message indicating that the command was not found, or the application is un-

known. You can also invoke this command by applying the entire path to Python. For Linux users it looks like this:

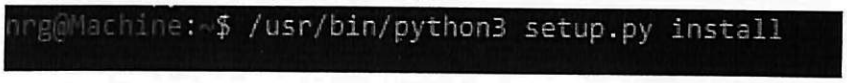


Figure 3.14: Entire path to Python

This method is especially comfortable for those who have different versions of Python and they need different versions of packages for each of them. All you have to do is to apply and write the entire path to Python and install the package.

Some packages contain C code, for example, the main C files, which must be compiled under the package for correct installation. In the case of Linux, you should have a C/C++ compiler installed, so you can install the package without any problem.

Installing packages using PIP (Package Installer for Python)

The pip program came out with Python 3.4. If you have an older version of Python, you need to install pip manually. The installation of pip is slightly different from what we discussed earlier. You need to log into PyPI, but instead of downloading the package and running its script setup.py , you will be offered to download one script called get-pip.py . After that, you needed to execute it as follows:

```
python get-pip.py
```

In the new versions of Python there is no need to install pip manually. Now it comes along with Python itself.

For this work the author installed the packages of pyspark in the way that is given below:

Figure 3.15: Installing pyspark for Python

And the author also installed the packge of kafka-python:

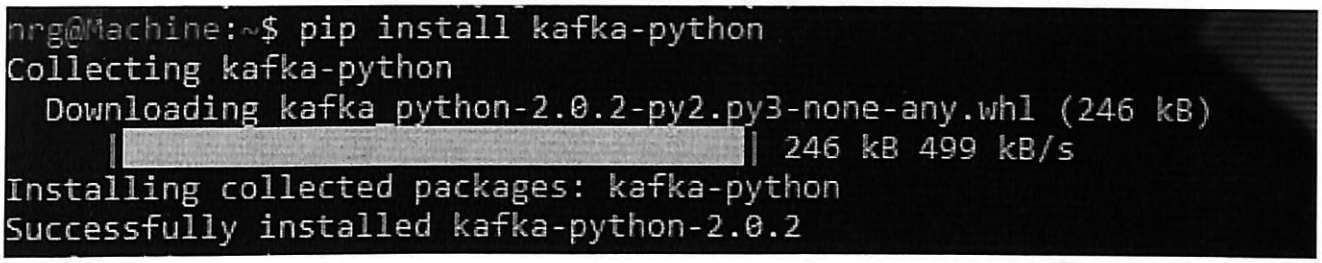


Figure 3.16: Installing kafka-python for Python

These two packages are necessary for Python Producer. Without them the Python Producer will not do its job.

4. Design and implementation

4.1 The process of log analysis

For the experiments the author used the ready access logs data from the internal website. Before analyzing them in every tool data needed to be cleaned up. In the beginning the logs were in SQL query format. It was hard to read and the author made it easier to read for comfortable analysis purposes by getting rid of SQL keywords.

4.1.1 The process of analysis in Apache Hadoop and Apache Spark

After the access logs data was cleaned up it was sent to HDFS to be read from the MapReduce job and analyzed. After all these actions you get analyzed results where there will be information about the amount of requests made from each IP address in definite date and time.

4.1.2 The process of analysis in Apache Spark Streaming

Same as it was in Apache Hadoop and Apache Spark the data cleans up, but instead of HDFS it will be sent to Apache Kafka by Python Producer. Again we get from that all the information about the number of actions made from every IP address in certain data and time.

To evaluate the performance of batch-based data processing and streams I conducted the analysis of clickstreams. The experiment was conducted in order to demonstrate the capabilities of analyzing streams and batch-based data using Hadoop and Spark. Since the possibilities go beyond accessibility, these experiments cannot be called a complete comparative analysis of these two technologies.

During the analysis of clickstreams, the access logs data is analyzed to retrieve the data set, for instance, the time when the server was visited and the IP address. The experiment was aimed to compare Hadoop and Spark and the time that was taken for analyzing the data. The number of data was approximately 8 million records. Firstly a log file is uploaded to HDFS for convenience and accessibility for the readability no matter from Hadoop or Spark.

As an extension to the architecture of batch-based data processing, the streaming architecture which is based on Spark streaming analyzes the data in real-time. While analyzing the data using streaming-based Spark we also calculate and accumulate the client's unique IP address and the timestamp of the moment when the event has occurred. Table 4.1 shows the tools and their versions that were used in these experiments.

Software	Java	Python	Apache Hadoop	Apache Spark	Apache Kafka	Zookeeper
Version	1.8	3.8.10	3.3.1	3.2.1	7.1.1	3.6.3

Table 4.1: Versions of Software

The implementation of this design was made in a virtual environment that is set up on ubuntu 20.04 LTS installed on Windows subsystem for Linux hosted on windows 11 which consists of two cores 16 GB RAM and 200 GB Disk capacity. In such a kind of environment tools Kafka, Spark, and Hadoop are installed in a single node cluster in a pseudo-distributed cluster. That environment is used to compare MapReduce in Hadoop, Spark Streaming, and Spark batch-based applications.

4.2 Batch Based Architecture Using Spark

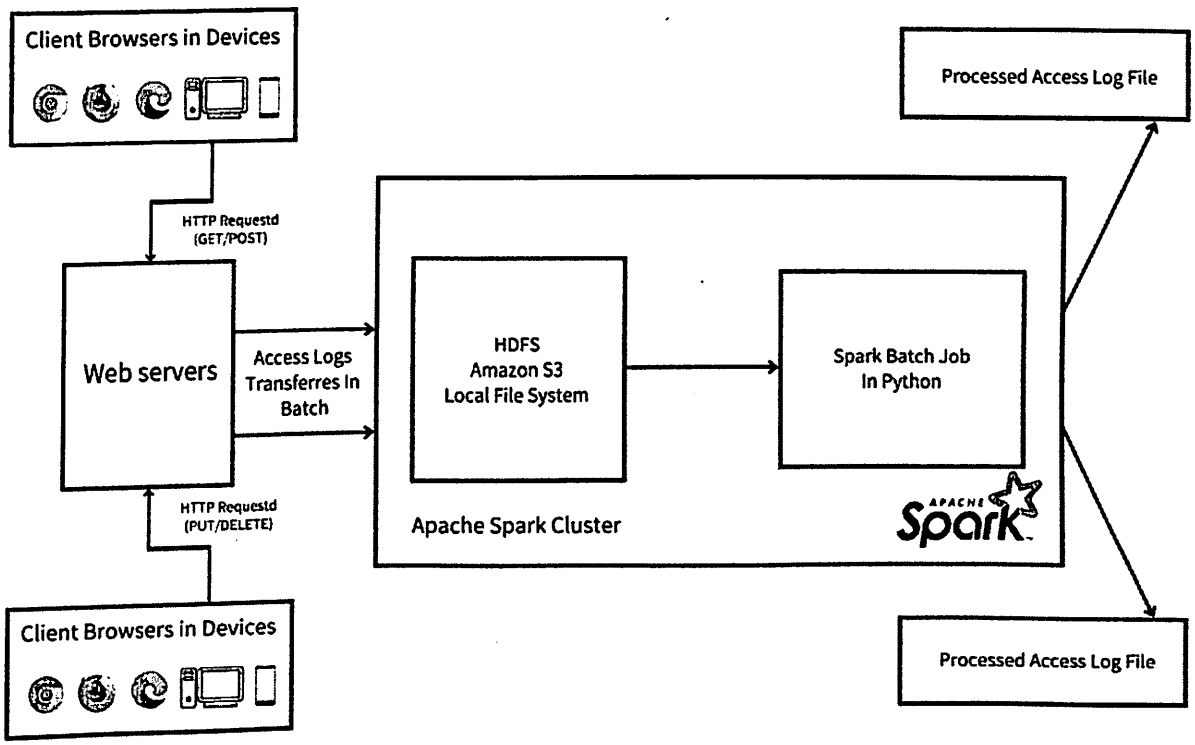


Figure 4.1: Spark Streaming Flow

While using this architecture all the access logs are collected and kept on web servers and analyzed using batch based Apache Spark. Access logs are collected on web servers when they are accessed from clients such as Edge, Chrome or Firefox, no matter if it is a personal handheld device or desktop. First, access logs are aggregated and sent to the Apache Spark storage cluster with the help of SFTP and then they are analyzed with Spark bath-based application. Access logs can be read by Apache Spark jobs through storage types such as HDFS, disk, Amazon S3 or Ceph.

Command which is shown below is used to run a script which starts Apache Spark job for analyzing 8 million lines of access logs and potentially can be modified for other types of logs.

```
spark-submit --executor-memory 4G --driver-memory 4G batchSpark.py
```

4.3 Batch Based Architecture Using Hadoop MapReduce Model

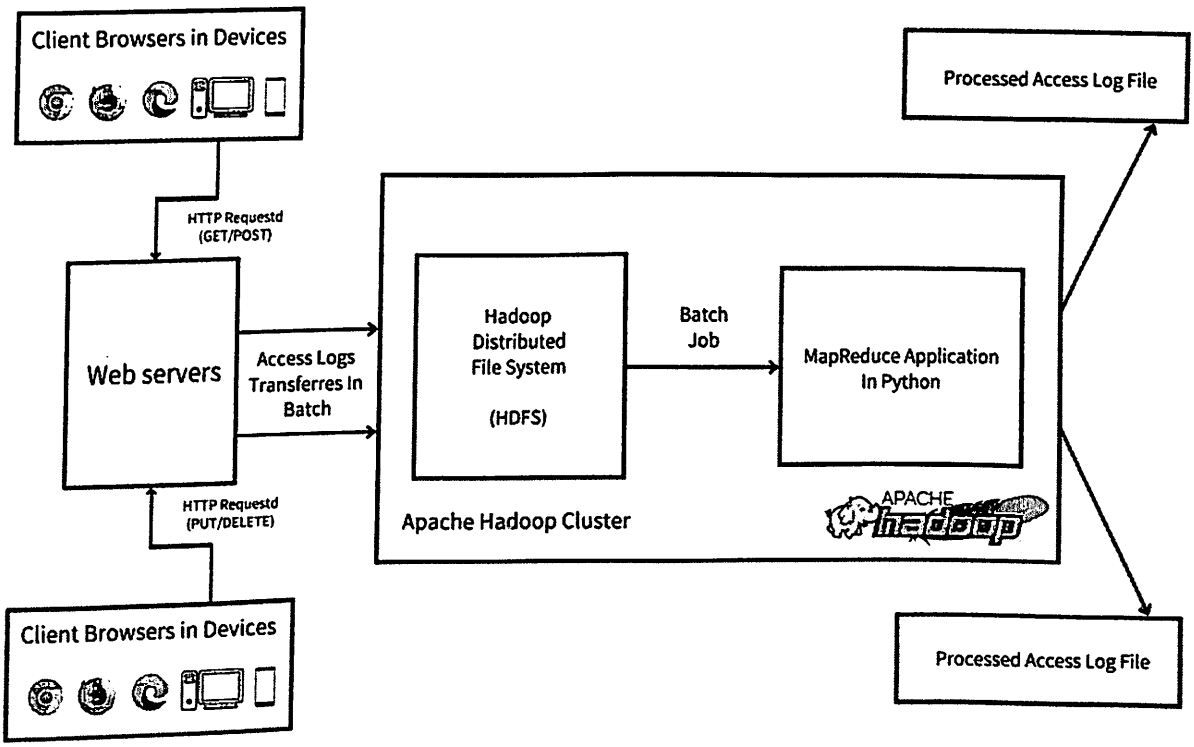


Figure 4.2: Architecture of analyzing through batch-based Hadoop

In the architecture of MapReduce which is based on Hadoop, as batch-based Spark, access logs are collected and analyzed, but in contrast the access logs can be analyzed in offline mode. Access logs are collected on web servers when they are accessed from clients such as Edge, Chrome or Firefox, no matter if it is a personal handheld device or desktop. First, access logs are aggregated and sent to the Hadoop storage cluster with the help of SFTP and then they are moved to HDFS which is using the Hadoop file system command. After copying the data to HDFS it is analyzed in batch mode via MapReduce job. Command which is shown below is used to run a script which starts a MapReduce job for analyzing 8 million lines of access logs and potentially can be modified for other types of logs.

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
```

```
3.3.1.jar -input access.log -output $HOME/hadoop/streaming/wordcount
-mapper "python3 mapper.py" -reducer "python3 reducer.py"
```

4.4 Streaming Based Architecture Using Spark

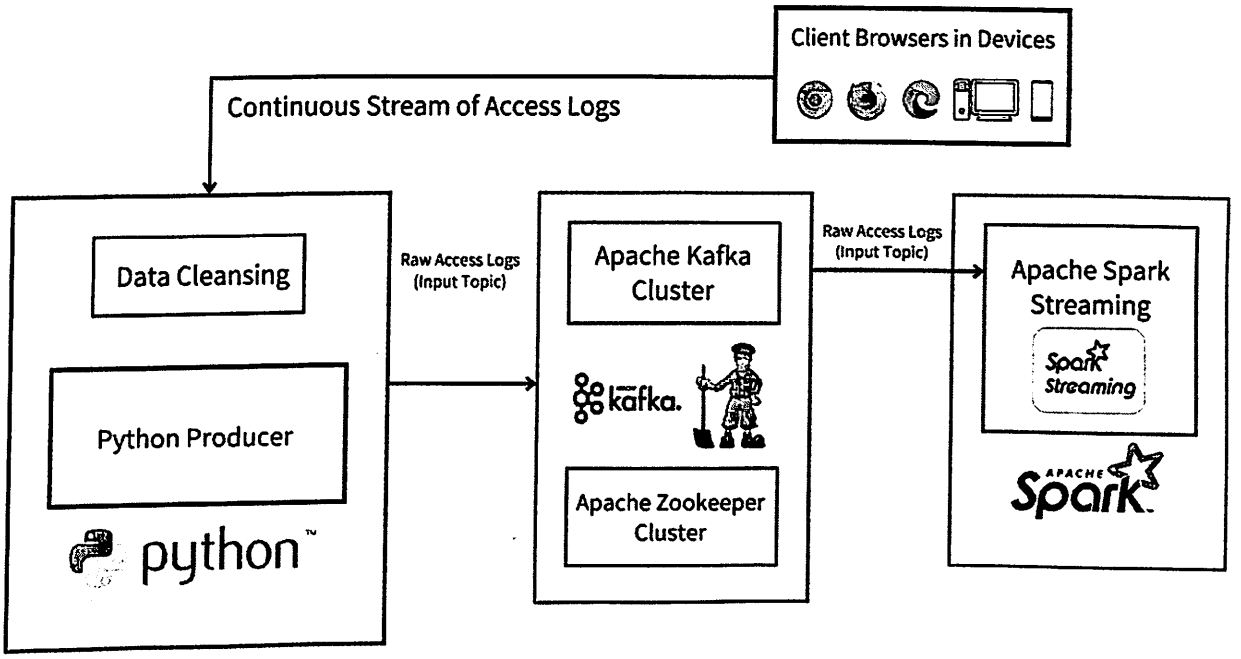


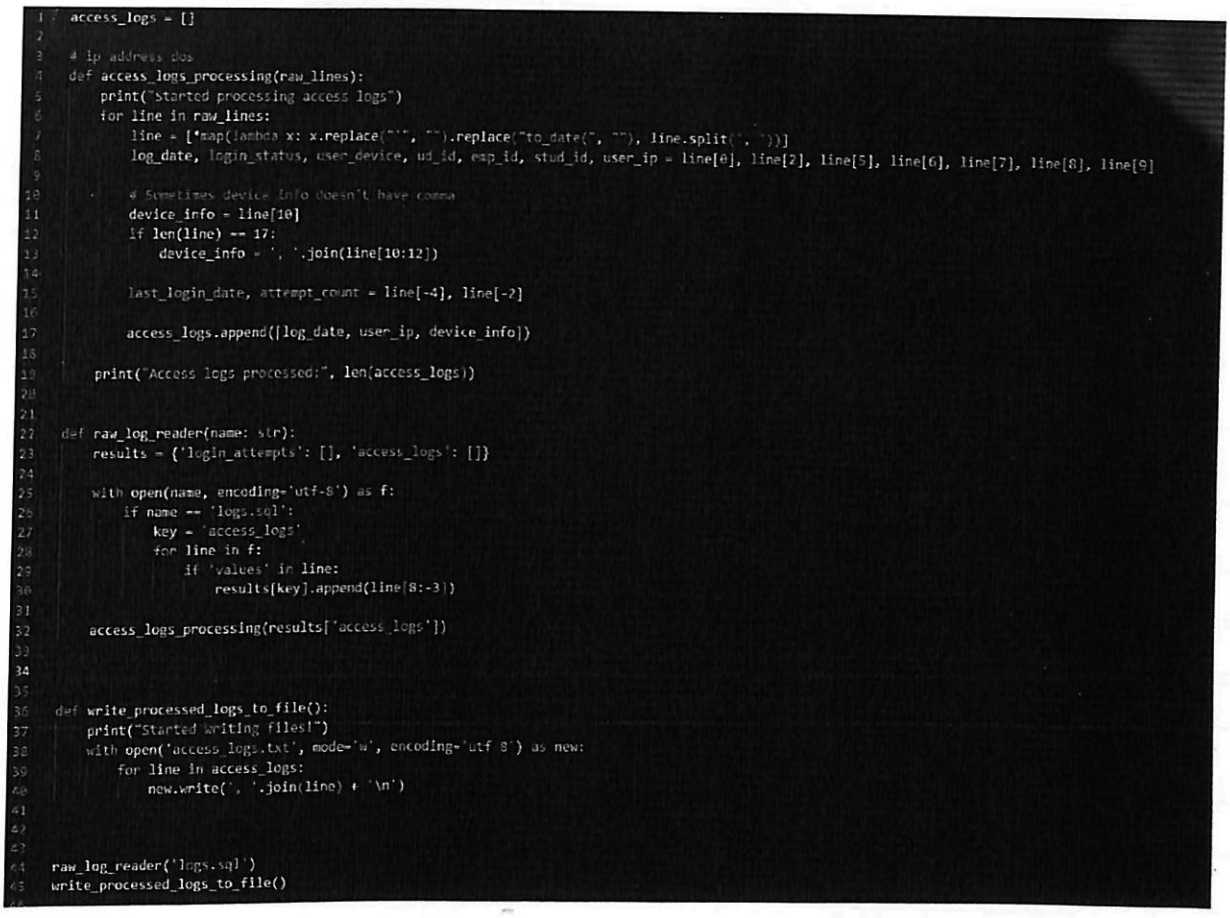
Figure 4.3: Spark Streaming Flow

In this given streaming based architecture the continuous access logs are collected through clients' devices and go into Python. Python firstly cleans up the data and the clean data goes to Python Producer. After Python Producer the data goes to specific topics of Apache Kafka. Then Apache Streaming consumes the data, analyzes it and prints out to the console.

The following command helps to clean up the data:

```
python3 cleanup.py
```

Here is the contents of cleanup.py file:



```
access_logs = []

# ip address dos
def access_logs_processing(raw_lines):
    print("started processing access logs")
    for line in raw_lines:
        line = [*map(lambda x: x.replace("'", "").replace("to_date(", ""), line.split(', '))]
        log_date, login_status, user_device, ud_id, esp_id, stud_id, user_ip = line[0], line[2], line[5], line[6], line[7], line[8], line[9]

        # Sometimes device info doesn't have comma
        device_info = line[10]
        if len(line) == 17:
            device_info = ', '.join(line[10:12])

        last_login_date, attempt_count = line[-4], line[-2]

        access_logs.append([log_date, user_ip, device_info])

    print("Access logs processed:", len(access_logs))


def raw_log_reader(name: str):
    results = {'login_attempts': [], 'access_logs': []}

    with open(name, encoding='utf-8') as f:
        if name == 'logs.sql':
            key = 'access_logs'
            for line in f:
                if 'values' in line:
                    results[key].append(line[8:-3])

    access_logs_processing(results['access_logs'])


def write_processed_logs_to_file():
    print("Started writing files!")
    with open('access_logs.txt', mode='w', encoding='utf-8') as new:
        for line in access_logs:
            new.write(', '.join(line) + '\n')


raw_log_reader('logs.sql')
write_processed_logs_to_file()
```

Figure 4.4: cleanup.py

Spark cluster is run in a streaming based Spark application and reads all the raw data from the specific topic of Kafka. And then this generates a direct stream from the topic which will read the data continuously. RDD will be created for every record that goes as stream every two second as presented in the given code.

Continuous stream logs in turns transform into micro matches in the interval of two seconds that can be processed by Spark Job.

5. Comparison and testing

The paper measures the time that has been sent to process the log files in batch based Hadoop, batch based Spark and streaming based Spark. Before this time is measured there is a step when the data is cleaned up with the help of Python script. In the experiment there were 8 million access logs that were analyzed and the time that has been sent to it are recorded.

While analyzing and getting the results the author has come to the conclusion that the analysis process in Apache Spark is approximately 82% faster than Apache Hadoop, which in turn proves the efficiency of Spark and claims that Spark is much faster than Hadoop. But within the comparison between the batch-based Spark and streaming based Spark, the Apache spark which is streaming based is faster for approximately 7.5%.

Hadoop	Spark	Streaming
1m50s	0m20s	0m18.5s

Table 5.1: Properties of the dataset environment

All the access logs were analyzed three times using batch-based Apache Hadoop, Apache Spark and streaming based Apache Streaming. In the below there is a command that starts to run on Hadoop to analyze these 8 million access logs.

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
3.3.1.jar -input access.log -output $HOME/hadoop/streaming/wordcount
-mapper "python3 mapper.py" -reducer "python3 reducer.py"
```

Command in the below run the script of batch based Apache Sark to analyze 8 million access logs.

```
spark-submit --executor-memory 2G --driver-memory 2G batchSpark.py
```

Average time that is sent to analyze the access logs between batch based Apache Hadoop, batch based Apache Spark and streaming based Apache Spark are shown in the Fig. 5.1

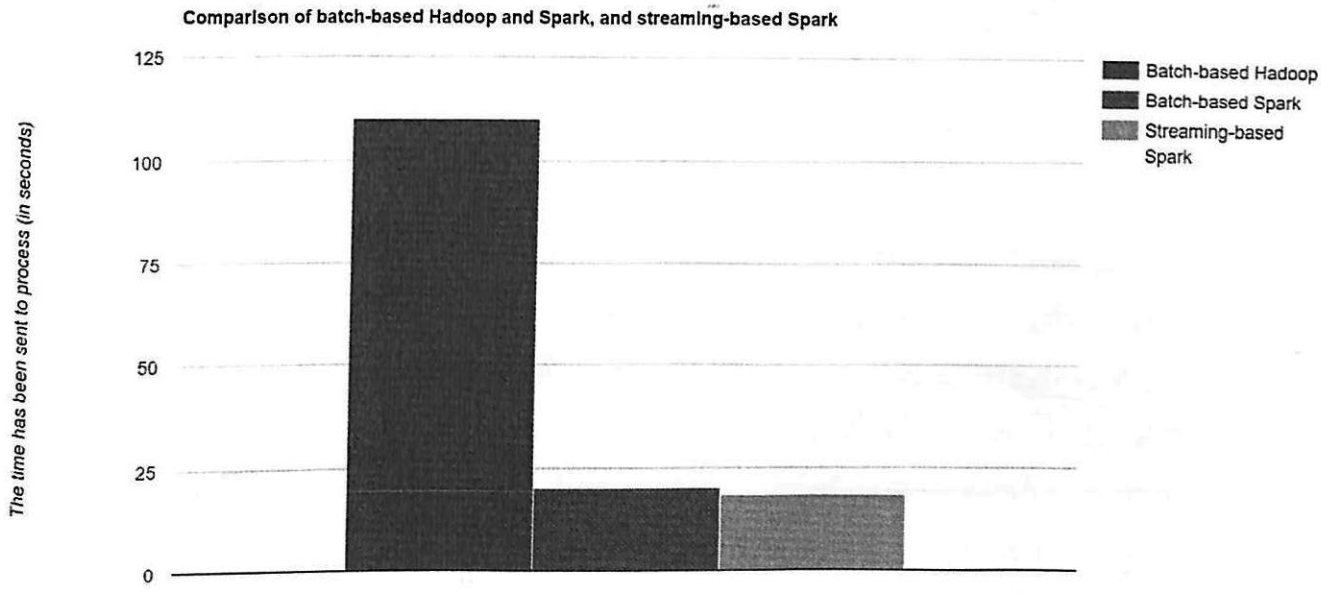


Figure 5.1: Comparison of batch-based Hadoop and Spark, and streaming-based Spark

6. Conclusion

After conducting these experiments for analyzing all the 8 million access logs in three different architectures are shown and observed in Fig.5.1. According to this figure it can be concluded that the data analyzed in batch based and streaming based Apache Spark are quite less than the Hadoop which is based on MapReduce Model. That is all because the data on the MapReduce model is analyzed in several steps, these steps cannot be done at the same time, it means that they are done one by one, and all these sequential steps depend on HDFS to store all the steps.

MapReduce, which was created relatively to Apache Hadoop, was actually created as a batch-based processing that can take from a few hours and days to analyze the data. It all can be explained with the reasons that MapReduce initially was not created as a fast tool. The Mapper process of MapReduce model analyzes the input data in key and value pairs, but the mapper can analyze only one pair every time. All the data in this model firstly are recorded to local disk and the ending of the phase "map", and are recorded to HDFS at the ending of the phase "reduce".

The MapReduce model of the Apache Hadoop contains several phases as reducer, mapper, input splits, shuffle and sort phases. Shuffle and sort phase is an input and output task in which the intermediate results from mapper stage are sorted and shuffled according to their key. This phase is hard and time consuming because it has to transfer the data firstly from networks to nodes of data. After this stage is done the reducer step starts to combine the data.

Spark is way faster than the MapReduce model as it works on RDD, and the data is stored without input and output. Also it does not have reduce and map phases, it means that data is read from the disk while processing is started and recorded to disk while the persistence of results is unnecessary. The data in Spark

RDD API can analyze multiple key and value pairs with the help of partitions.

While the HDFS, while working and analyzing data, scatters them across servers and Spark uses Direct Acyclic Graph (DAGs) instead, that is, with the help of this, data is not replicated and the whole process of work goes in the DAGs themselves. DAGs are convenient because if some error has occurred and the work has stopped, then you can start from this moment without losing data. The RDD in Spark can do both batch based and streaming based jobs that is a big challenge for batch based Apache Hadoop platform.

References

[1] O. Awotipe. "Log analysis in cyber threat detection". In: (2020).

[2] H. Barringer et al. "Formal analysis of log files." In: *Journal of aerospace computing, information, and communication* 7.11 (2010), pp. 365–390.

[3] S. M. Hernandez. "Near real time fraud detection with Apache Spark". In: *(Doctoral dissertation, Universitat Politècnica de Catalunya. Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, 2015 (Enginyeria de Telecomunicació))* (2015).

[4] Dean J. and Ghemawat S. "MapReduce: simplified data processing on large clusters." In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[5] K. A. Kent and M. Souppaya. "Guide to Computer Security Log Management". In: (2006).

[6] J. Kreps, N. Narkhede, and J. Rao. "Kafka: A distributed messaging system for log processing." In: *Proceedings of the NetDB* (2011, June), pp. 1–7.

[7] C. Lam. *Hadoop in action.* Simon and Schuster, 2010.

[8] X. Liao et al. "An enforcement of real time scheduling in Spark Streaming". In: *2015 Sixth International Green and Sustainable Computing Conference (IGSC)* (2015, December), pp. 1–6.

[9] I. Mavridis and E. Karatza. *Log file analysis in cloud with Apache Hadoop and Apache Spark.* 2015.

[10] T. Oktay and A. Sayar. "Analyzing big security logs in cluster with apache spark". In: *INNS Conference on Big Data* (2016, October), pp. 131–138.

[11] P. Parthiban and S. Selvakumar. "Big data architecture for capturing, storing, analyzing and visualizing of web server logs." In: *Indian Journal of Science and Technology* 9.4 (2016), pp. 1–9.

[12] S. Patil. "Use of web log file for web usage mining". In: *International Journal of Engineering Research & Technology (IJERT)* 2.4 (2013).

[13] S. Shahrivari. "Beyond batch processing: towards real-time and streaming big data." In: *Computers* 3.4 (2014), pp. 117–129.

[14] V. Verma, A. K. Verma, and S. S. Bhatia. "Comprehensive analysis of web log files for mining." In: *International Journal of Computer Science Issues (IJCSI)* 8.6 (2011).

[15] Wingerath W. et al. "Real-time stream processing for Big Data." In: *it-Information Technology* 58.4 (2016), pp. 186–194.

[16] T. White. *Hadoop: The definitive guide.* O'Reilly Media, Inc., 2012.

[17] M. Zaharia et al. "Resilient distributed datasets". In: *A fault-tolerant abstraction for in-memory cluster computing in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation.* (2014).