Received: December 04, 2022; **Accepted:** November 09, 2023.
DOI. 10.30473/coam.2023.66151.1222
Winter-Spring (2024) Vol. 9, No. 1, (81-96)
Research Article

Open Access

Control and Optimization in Applied Mathematics - COAM

Big Data Analytics and Data Mining Optimization Techniques for Air Traffic Management

Abbas Ali Rezaee[1], **Hadis Ahmadian Yazdi**[2], **Mahdi Yousefzadeh Aghdam**[3], **Sahar Ghareii**[4]

[1]Department of Computer Engineering and Information Technology, Payame Noor University, Tehran, Iran,
[2]Department of Computer Engineering, Neyshabur Branch, Islamic Azad University, Neyshabur, Iran,
[3]Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran,
[4]Aviation Engineer, Mashhad Airport, Mashhad, Iran.

Correspondence:
Abbas Ali Rezaee
E-mail:
a_ rezaee@pnu.ac.ir

Abstract. With the advancements in science and technology, the industrial and aviation sectors have witnessed a significant increase in data. A vast amount of data is generated and utilized continuously. It is imperative to employ data mining techniques to extract and uncover knowledge from this data. Data mining is a method that enables the extraction of valuable information and hidden relationships from datasets. However, the current aviation data presents challenges in effectively extracting knowledge due to its large volume and diverse structures. Air Traffic Management (ATM) involves handling Big data, which exceeds the capacity of conventional acquisition, matching, management, and processing within a reasonable timeframe. Aviation Big data exists in batch forms and streaming formats, necessitating the utilization of parallel hardware and software, as well as stream processing, to extract meaningful insights. Currently, the map-reduce method is the prevailing model for processing Big data in the aviation industry. This paper aims to analyze the evolving trends in aviation Big data processing methods, followed by a comprehensive investigation and discussion of data analysis techniques. We implement the map-reduce optimization of the K-Means algorithm in the Hadoop and Spark environments. The K-Means map-reduce is a crucial and widely applied clustering method. Finally, we conduct a case study to analyze and compare aviation Big data related to air traffic management in the USA using the K-Means map-reduce approach in the Hadoop and Spark environments. The analyzed dataset includes flight records. The results demonstrate the suitability of this platform for aviation Big data, considering the characteristics of the aviation dataset. Furthermore, this study presents the first application of the designed program for air traffic management.

Keywords. Data mining, Air traffic management, Clustering, K-Means algorithm, Hadoop platform, Spark platform optimization.

MSC. 68T07.

How to Cite
Rezaee, A.A., Ahmadian Yazdi, H., Yousefzadeh Aghdam, M., Ghareii, S. (2024). "Big data analytics and data mining optimization techniques for air traffic management", Control and Optimization in Applied Mathematics, 9(1): 81-96.

1 Introduction

There are various definitions for Big data. Big data can be defined as data that exceeds the processing capacity of current systems [11], or it can be described as the data generated from a large amount of information. Storing, processing, and analyzing such data using traditional technologies is extremely challenging. Nowadays, Big data is of particular interest in the aviation industry [8, 13]. These data possess three main characteristics: variety, volume, and veracity.

1.1 Variety

Currently, there is an immense amount of data sources and various types of data. As a result, these variables produce different data structures that can be categorized into three groups: structured, semi structured, and unstructured. Structured data can be stored quickly, but analyzing unstructured data is complex due to their random production. Unstructured data do not consist of constant fields, but they have tags for decomposition [13]. The majority of the world's data volume consists of unstructured data. Some portions of the existing data are stored in databases, web pages, JSON, and XML formats, while the remaining data is stored in files with various formats, making their processing complex in practice.

1.2 Data volume

In today's era, phenomena such as the internet, electronic devices, mobile devices, network infrastructures, and other resources generate data within and outside organizations. Due to the exponential growth of data, individuals face larger sizes and volume of data than terabyte and petabytes. The generation veracity or rate refers to data generated by application programs and sensors in the environment at high speeds and in real-time. Most of these data should be processed and stored in real-time. The aviation industry utilizes most of the applications of this data [6, 9].

1.3 Veracity

Regarding the receipt of data from different sources, it is important to note that many of these sources may be unreliable. For instance, in a social network, there can be numerous perspectives on a specific subject. It is not reasonable to assume that all of these perspectives are truthful and dependable, particularly when dealing with a large volume of information. Consequently, relying solely on the veracity of the data may not be suitable for certain applications. In other words, the data may lack sufficient validity for certain applications [2, 6]. Additionally, the value of data can change over time, rendering it different. During this period, data may lose their relevance and acquire new value. While long-term information retention is important for change analysis and data volatility, extending the information retention period will inevitably incur significant implementation costs that must be taken into consideration.

One of the complex challenges in the field of Big data is data visualization [1, 4]. To enhance the comprehensibility and readability of vast amounts of information with intricate relationships, appropriate information analysis and visualization methods can be employed. Currently, proposed approaches in Big data processing possess several standard features: execution on existing hardware, which allows for parallel processing and low-cost hardware upgrades; utilization of cost-effective analysis and advanced visualization techniques for the convenience of end users; simultaneous utilization of various tools and libraries that form the data architecture of an organization; and incorporation of No-SQL databases as part of the organizational architecture and data platform. The two main common approaches for Big data processing and analysis are Hadoop and Spark.

In this paper, we will first examine the evolutionary trends in processing aviation Big data, and subsequently explore and discuss methods for analyzing this data. We will implement the K-Means algorithm optimization using the map-reduce method in both Hadoop and Spark environment. K-Means map-reduce is one of the most critical and widely used clustering methods. Finally, we will conduct a case study to analyze and compare aviation Big data of air traffic management systems in the United States using the K-Means map-reduce approach in Hadoop and Spark environment. This case study includes working records in flights. The results obtained demonstrate the suitability of this platform for aviation Big data, as indicated by the features of the aviation dataset. Furthermore, this study represents the first implementation of the application program for air traffic management systems.

2 Literature Review

In today's world, information is continuously growing and being generated. There is typically a vast amount of data underlying various internet services and applications, and this data is increasing rapidly. In some cases, the growth rate is exponential. We will divide Big data into two forms: batch and streaming (see Figure 1). In this section, we will discuss the various processing methods and challenges that exist in handling Big data.

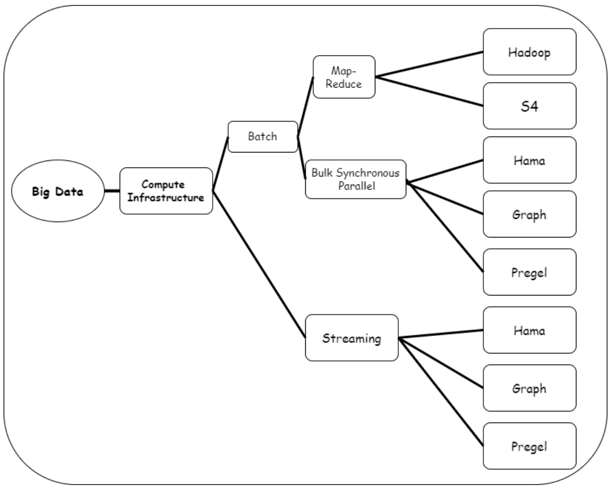

Figure 1: Infrastructure for Big data computation.

2.1 Batch data processing

Currently, the most important model for processing Big data is the map-reduce model. Many companies use this model to process their data. The inventor of this method is Google, which introduced Google map-reduce but does not grant access permission to it. Therefore, if someone needs access to the main code, they should use Hadoop [12]. Hadoop and Spark have introduced an open-source version of map-reduce. Hadoop is an open-source tool and framework based on Java (see Figure 2). This software stores data in its system file and execute tasks in a cluster of standard systems using Hadoop Map-Reduce [10].

As the name of the model implies, each map-reduce program consists of two simple functions: map and reduce. Programmers only need to write the map and reduce functions and provide them to the program.

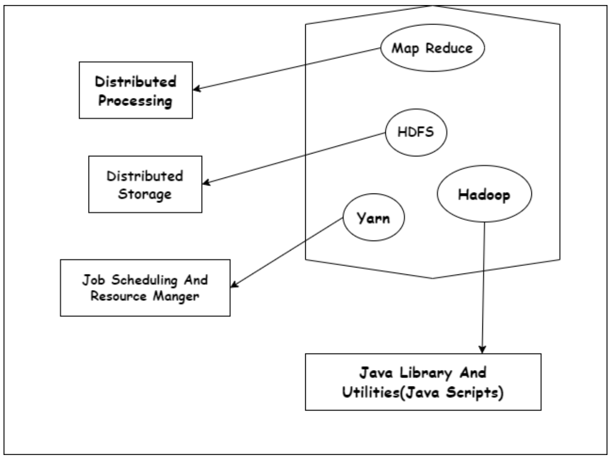
Map Reduce
Distributed Processing
HDFS
Distributed Storage
Hadoop
Yarn
Job Scheduling And Resource Manger
Java Library And Utilities(Java Scripts)

Figure 2: HDFS and map-reduce structure.

The system itself will execute the remaining operations. When the programmer provides his map-reduce functions to the system, the system converts these two functions into a workflow. The map function receives a list from input and then converts them to key-value pairs. The map-reduce library evaluates all of the intermediate values generated by the map function, groups values with the same key into a category, and then sends them to the reduce function. Each reduce function receives a key with a list of values as the input and converts this input to a smaller list, typically reducing it to a maximum of one value. In most cases, the number of reduce functions is fewer than of map functions. At any given moment in the system, only two map functions and one reduce function can be activated. Hadoop map-reduce has two versions, each of which will be explained below. Hadoop had to create a separate resource manager for each of its products, and the task scheduling problems across different machines and resources were one of the weaknesses of this version [10]. To address this problem, Hadoop introduced the next version of map-reduce, known as YARN. YARN serves as a negotiator for resources, and all Hadoop products, such as map-reduce, graph, Spark, and Storm, utilize YARN for their resource management (see Figure 3). The two management nodes of this version are the resource manager and the node manager. The resource manager is the main process responsible for allocating resources. It estimates the capacity of each node and then strives to allocate resources fairly. To send input data to map functions, it is essential to divide the data into smaller segments. Additionally, on each computer, a segment of the data is stored to increase processing speed. After the mapping phase is completed, the intermediate values are stored in the local memory of the computer. During this phase, the outputs related to keys are combined and sent to the reduce functions. There is an action called “shuffling” that the hash function performs by default, and we utilize this action to accomplish this task.

2.2 Stream data processing

The purpose of stream processing is to derive commercial value from continuously generated moving data. We can differentiate between batch data processing and stream processing is that batch data processing comes to an end after a certain period, whereas stream processing continues uninterrupted unless we abort the process. The objective of stream processing is to enable real-time or near-real-time decision-making based on sequentially entering data into the system. Therefore, in this section, we will explore a set of proposed parallel methods for stream processing. Storm is a real-time data stream processor in

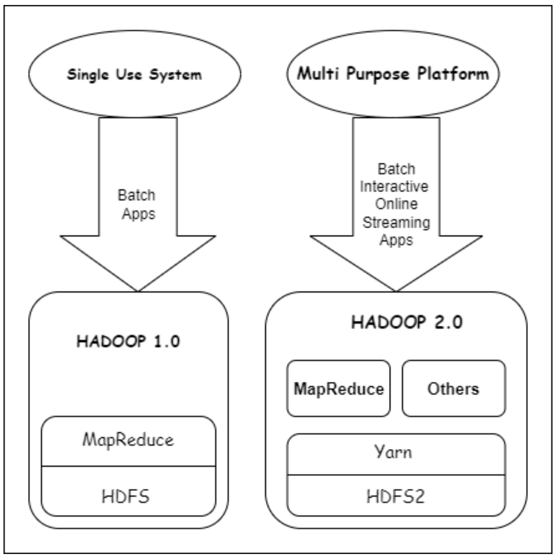

Figure 3: Hadoop structure version 1 and 2 YRAN in Hadoop.

distributed systems that are freely available as an open-source tool for public use [10]. A Storm program consists of a workflow, with each workflow consisting of tuples. The input to the system is a stream of tuples, referred to as a topology. A topology is a directed graph in which the vertices represent computations, and the edges represent data streams between computations. We can categorize the vertices into two types: spouts and bolts. Spouts serve as data sources, providing tuple data to other vertices. Bolts receive data, perform their corresponding processing, and, if necessary, forward the data to another bolt. Companies such as Twitter and Yahoo use Storm. However, this model has certain drawbacks; including architectural complexity and the fact that program execution time is unpredictable. To address these issues, the Heron system was introduced. As the volume of data generated by Twitter exceeded Storm's capabilities in terms of management, debugging, and scheduling, Twitter introduced Heron as a new technology in 2015. Heron is one of the latest programs for stream processing [12]. This program revamped the internal structure and reduced the complexity of the Storm system. Externally, there is no difference between Heron and Storm, and any programs defined for Storm can be executed on Heron without modification. In Heron, each job is treated as a separate process, unlike Storm where all tasks were executed on threads within the executor, making those threads unpredictable. In Heron, each task is executed as a predictable and independent process called a Heron Instance. On each computer in a cluster, there is an entity known as the stream manager, which is responsible solely for sending and receiving streams between systems and managing them.

2.3 Graph processing

Graph processing serves as another important aspect of handling Big data. A prominent example in this domain is the analysis of social networks, where the relationships between individuals or countries are examined by processing friendship graphs (see Figure 4). In the past, researchers utilized algorithms like map-reduce for graph processing. However, big graph processing poses several challenges. Firstly, the locality of access to graph memory is typically low. Secondly, processing on the vertices of each graph is intensive. Lastly, the degree of parallelization of vertices during runtime varies due to the large size of the graph. Consequently, traditional programs like map-reduce struggle to effectively process graphs. To address these challenges, Google introduced a new system called Pregel, specifically designed for efficient graph processing. Pregel operates based on the Bulk synchronous parallel (BSP) model. In Pregel, users define a function to be executed at each vertex. In the graph, each vertex possesses an identifier, a value and functions as an independent entity. Edges have values but no names, and they are defined based on vertices they connect. Therefore, edges are not independent entities. Each vertex can process its values, modify them dynamically, receive messages from connected vertices, or send messages to them. When the values of a vertex no longer change, it becomes inactive, and if it receives a message, it becomes active again. The algorithm terminates when all vertices become inactive. Pregel finds applications in various tasks, including solving the shortest route problem.

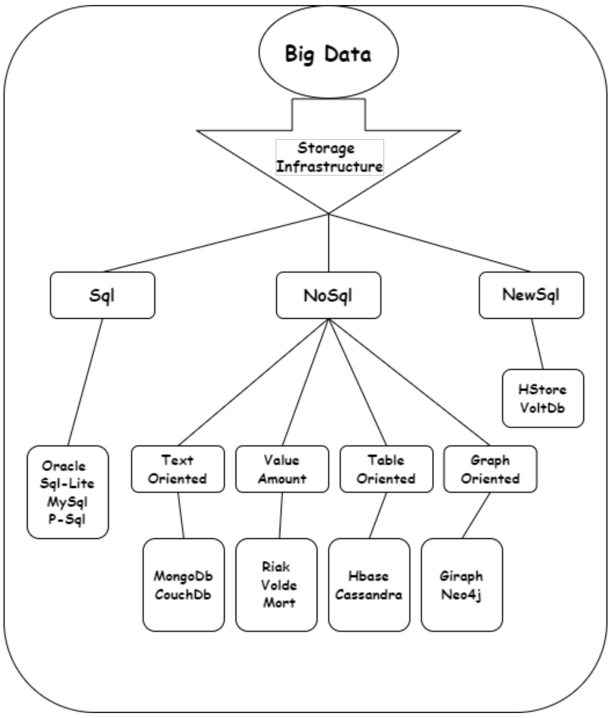

Figure 4: Infrastructure and technologies for storing Big data.

3 Big Data Structure in Aviation

A significant volume of data is generated every second by a multitude of sensors and other data sources in airplanes. Real-time or near-real-time analysis of this data is particularly crucial for flight-related predictions, especially in emergency situations. Before undergoing data processing, the acquired data requires a preprocessing phase. Extracting meaningful patterns from Big data sets necessitates the application of machine learning and statistical techniques. The accurate representation of the discovered knowledge is imperative for comprehension and utilization. The analysis of Big data encompasses five stages (see Figure 5), which are independent of the application field. Figure 5 illustrates these stages as applied in the context of aviation. It is necessary to integrate data from diverse, heterogeneous sources such as sensors, cameras, radar, or air into a unified structure. The integration stage involves combining data from multiple sources to provide a cohesive view of the data. Data management involves the retrieval and extraction of relevant data for analysis. Pre-processing ensures that the data is cleaned, normalized, and reduced, thereby preparing it for the analyzing process [5, 3]. Subsequently, prediction and classification methods can be utilized to extract meaningful information. The final stage of Big data analysis is knowledge representation, which highlights the extracted knowledge for various aviation applications.

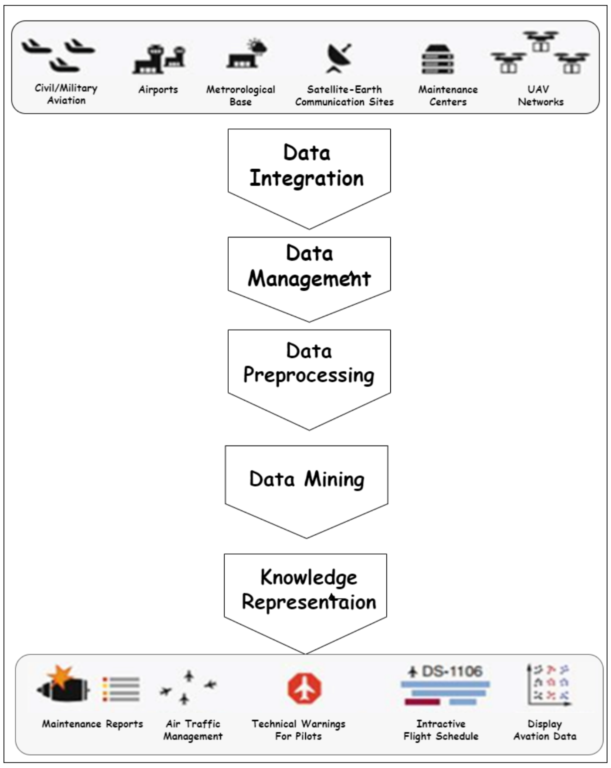

Figure 5: Big data management in the aviation industry.

3.1 Machine learning for data mining of Big data

We could classify research trends in machine learning for Big data analysis into two categories:

- These contain information about assessments in module presentations.The first category focuses on machine learning algorithms on parallel operating systems.
- The second category aims to redesign machine learning algorithms for parallel computations, such as neural network algorithms.

Machine learning algorithms can be utilized to explore and analyze various problems among different data mining algorithms to solve specific issues. These algorithms are commonly used as “search” algorithms to find the desired solution (refer to Figure 6). Most machine learning algorithms can be employed to find an approximate solution for problem optimization. For example, the genetic algorithm is one of the machine learning algorithm that can be used not only for solving clustering problems but also for solving iteration algorithm extraction problems. Furthermore, machine learning is not only suitable for solving problems encountered in knowledge analysis (KDD), but it can also enhance performance in other aspects of KDD, such as improving input performance reduction capabilities. In research [10], a group of researchers demonstrated that various methods, including traditional exploration algorithms, statistical methods, pre-processing solutions, and even graphical interfaces are employed in profile tools and operating systems for Big data analysis. The results of the study indicate that machine learning algorithms are a fundamental component of Big data analysis (see Figure 6).

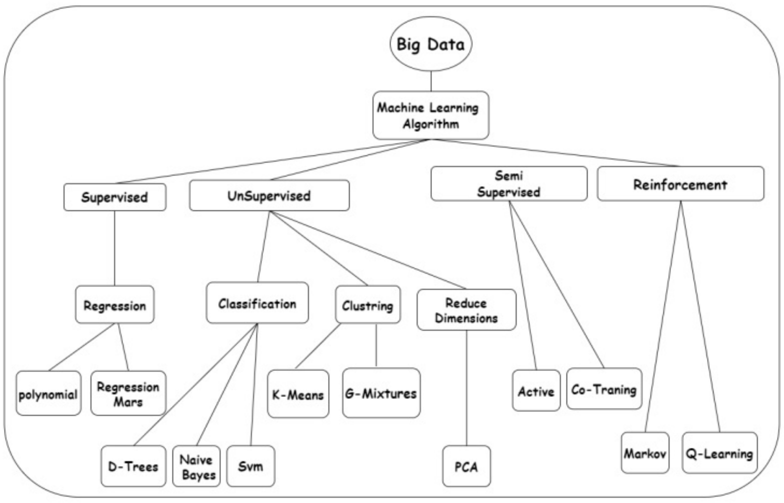

Figure 6: Machine learning algorithms in the field of Big data.

4 Big Data Analyzing Methods Using Data Mining in Aviation

One of the common and significant issues in the field of Big data is memory constraints. Since it is not feasible to store all data in a single memory, various approaches need to be considered for their analysis. Examples of Big data include transactions, network flow monitoring, gens analysis, weather forecasting, and astronomy. Big pre-processing, which involves data analysis methods, plays a crucial role in this context. Data mining is employed to discover important information from large dynamic datasets, extracting meaningful patterns, relationships, and filtering out irrelevant items. Forecasting models enable future predictions. For instance, Data mining techniques can assist in identifying individual or combined factors contributing to incidents and airplane accidents. data mining techniques are classified into the following areas: classification (supervised learning), clustering, relationship analyzing, time-series analyzing, and remote analysis. The classification process assigns data to predefined classes. Several classification methods have been mentioned in the literature, including Support Vector Machine (SVM), decision tree, KNN, and Bayesian network

4.1 Clustering

Clustering refers to a method of grouping objects or data, where each instance of data (or object) is assigned to a group (cluster) or remains unassigned after group definition. In clustering, there is no prior knowledge about data, and data classification is based on their similarity. We outline two primary objectives of clustering below:

- Data within a cluster exhibit the highest similarity.
- Data across different clusters display the greatest dissimilarity.

One of the most critical aspects in clustering is how to reduce data complexity. Clustering can be divided into two categories: machine-performed techniques, such as sampling and dimension reduction, and techniques involving multiple machines, such as parallelization and map-reduce. This implies that traditional solutions have applications for Big data, but it is necessary to address the memory requirements by utilizing sampling and dimension reduction to reduce data complexity. In sampling, the number of data entering the analysis stage is reduced, while in dimension reduction, the dataset size is decreased. In Reference [7], the authors classify clustering algorithms into five categories based on their performance, which are briefly explained in Table 1.

Table 1: The performance of various clustering algorithms.

row	Various clustering algorithms	How does the algorithm work?	Examples
1	Partitioning of algorithms	Clustering algorithms are used to separate n samples of data into k clusters based on a distance criterion. The purpose of these algorithms is to ensure that each cluster contains at least one instance of data and that each sample of data belongs to exactly one cluster.	K-Means, K-medoids , K-modes, PAM,CLARA, CLARAS, FCM
2	Hierarchical algorithms	Hierarchical clustering methods organize data into a hierarchical tree based on the distance criterion. These methods can be categorized as top-down or bottom-up and are typically distributed using greedy algorithms.	BRICH ,CURE,ROCK ,Chameleon ,Wards, SNN, CACTUS, GRIDCLUST
3	Density-based algorithms	Density-based algorithms cluster data based on their density in the neighborhood. Clusters are defined as dense regions that grow in the direction of density, and their growth rate can be controlled with a threshold value. These algorithms are suitable for finding clusters of desired shapes and protect against the presence of noisy data.	DBSACN,OPTIC, BCLASD, GDBSCAN , DENCLU, SUBCLU
4	Grid-based algorithms	Grid-based algorithms divide the data space into a lattice-like grid with a limited number of cells. The main advantage of this method is its fast processing, which is a linear function. Data are clustered based on the statistical parameters of each cell in the network, and the efficiency of these methods depends on the size of the grid.	STING, Wave Cluster, BANG,CLIQUE, OptiGrid, MAFIA,ENCLUS, PROCLUS, FC, STRR
5	Model-based algorithms	Model-based algorithms aim to find a fit between the data and a predefined mathematical model. These methods assume that the data set is generated by a random distribution method, and the number of clusters is automatically calculated based on standard statistics such as noise level.	EM, COBWEB, CLASSIT, SOM, SLINK

4.2 K-Means algorithm

The K-Means clustering aims to partition n objects into clusters, where each object belongs to the cluster with the nearest mean. This method generates exactly k distinct clusters of the greatest possible dissimilarity. Let $C_1, \ldots, C_k$ be a partition of the n samples into k clusters. The quality of the clustering can be evaluated using the following cost function:

$$\Sigma_{j=1}^{k} \Sigma_{i \in C_j} ||x_i - \mu_j||^2, \tag{1}$$

$$\mu_j = \frac{1}{|c_j|} \Sigma_{i \in C_j} x_j, \tag{2}$$

where j is the center of cluster j.

This cost function represents the squared error resulting from vector quantization, where n data samples are replaced by their respective cluster centers. Interpreting each data sample as a point particle with unit mass, the cost function can also be seen as the total moment of inertia of the system, where

$$\Sigma_{i \in C_j} ||xi - \mu_j||^2, \tag{3}$$

represents the moment of inertia of cluster j. Minimizing the cost function for all partitions $C_1, \ldots, C_k$ is an NP-hard problem. The K-Means algorithm provides an approximate solution. The best number of clusters, k, which leads to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance or the squared error function. The algorithm starts from an arbitrary location of the cluster centers, finds the partition induced by the nearest cluster of each data sample, updates the cluster centers, and iterates. The cost function decreases at each iteration, unless the clusters remain unchanged. This can be formulated as the following optimization problem:

$$\min_{c_1,\ldots,c_k;\mu_1,\ldots,\mu_k} \Sigma_{j=1}^{k} \Sigma_{i\in C_j} ||x_i - \mu_j||^2. \tag{4}$$

By observing that the K-Means algorithm performs alternating minimization in the clusters $C_1, \ldots, C_k$ and the cluster centers $\mu_1, \ldots, \mu_k$, it is evident that the cost function cannot increase, ensuring convergence of the algorithm. It is important to note that the convergence point does not necessarily minimize the cost function. The pseudo-code for the algorithm is provided below. The algorithm stops when the clusters remain unchanged (see Table 3).

Algorithm 4 The pseudo-code of K-Means algorithm

```
1: procedure K-Means Clustering
2:     Choose the number of clusters ($k$) and obtain the data points.
3:     Place the centroids $C_1, C_2, \ldots, C_k$ randomly.
4:     Repeat steps 4 and 5 until convergence or until the end of a fixed number of iterations.
5:     For each data point $x$:
         • Find the nearest centroid ($C_1, C_2, \ldots, C_k$),
         • Assign point $x$ to that cluster.
6:     For each cluster $j$, $(j = 1 \ldots k)$, set the new centroid to the mean of all points assigned
       to that cluster.
7: end procedure
```

5 Customizing Map-Reduce based K-Means on Hadoop and Spark Platform for Aviation Big Data

No-SQL databases typically employ a key-value structure, developed as non-relational databases, with horizontal scaling and structure-less storing method. Modeling in the relationship-based method focuses on the complex relation of multiple schemas, but these aspects are not considered in the case of No-SQL and shame-less databases. Consequently, traditional modeling methods are no longer efficient, necessitating a new methodology for managing Big data. In this study, we select a large scale dataset of flights information in an airport TMA environment, and utilize the map-reduce method to reduce and store the Big data. This phase involves the following steps:

- Collecting aviation data from ADS-B and ATN network.
- Modifying flight-related variables.
- Pre-processing the collected data.
- Creating a data-store object for flight dataset retrieved from ADS-B.
- Developing a map function that calculates the maximum flight time within each data chunk of the data store.
- Implementing a reduce function that determines the maximum value among all the calculated values from the map function.

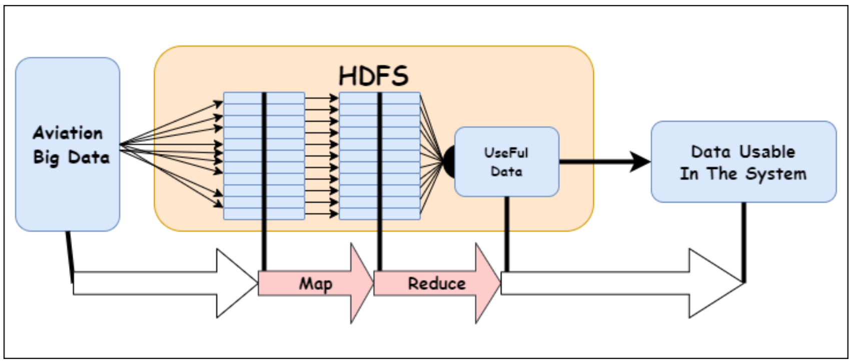

Figure 7: A schematic structure of the HDFS system.

This example illustrates the utilization of the map-reduce function and data store to process a substantial volume of file-based data (Big data). The map-reduce algorithm forms the foundational basis for numerous contemporary Big data programs. While this example runs on a computer, the code can scaled to leverage the Hadoop and Spark platforms.

5.1 Dataset

The datasets used in this study are historical records of internal airplane flights within the United States between 1987 and 2008 in the national air traffic management network. The complete collection of data is available at the following address: http://stat-computing.org/dataexpo/2009/the-data.html (see Table 2).

Variable names and descriptions and dataset characteristics of USA internal Flights (1987-2008) in the national air traffic management network. In his example, we aim to find the longest flight time among all available records in the dataset. The following steps should be taken:

1. Create storage for the airline dataset.
2. Develop a map function that calculates the maximum flight time for each data chunk in the storage.
3. Create a reduce function that determines the overall maximum value among all the maximums from the map function.
4. The formula for estimating Hadoop or Spark storage (H) is as follows:

$$H = c \times r \times \frac{s}{(1-i)}, \tag{5}$$

where, c and r represent the average compression ratio and replication factor, respectively. It depends on the type of compression used (e.g., Snappy, LZOP) and the data size. When no compression is used, $c = 1$. In production clusters, r is typically set to 3. S indicates the size of data to be transferred to Hadoop or Spark. This can be a combination of historical data and incremental data. For example, the incremental data may be daily and projected over a certain period (e.g., three years). i represents intermediate factor. The working space of Hadoop or Spark is dedicated to storing intermediate results of map phases. It is usually set to $\frac{1}{3}$ or $\frac{1}{4}$. The formula for estimating the number of data nodes (n) is as follows:

$$n = \frac{h}{d} = c \times r \times \frac{s}{(1-i)d}, \tag{6}$$

where d is the available disk space per node. All other parameters remain the same as in (5).

Terminal Manoeuvring Area

Table 2: The performance of various clustering algorithms.

row	Variable name	Feature description
1	Year	1987-2008
2	Month	1-12
3	DayofMonth	1-31
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time, in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

5.2 Creating data storage and map and reduce functions

We utilize data storage to access tabular textual files on disk using the Hadoop and Spark distributed file system (DFS). This mechanism also serves as a means to invoke the map function in a timely manner for map-reduce operations. The data center automatically analyzes the input data and makes the best guess for the data type of each column. In cases where the name-value binary argument is missing, it must be correctly replaced to ensure proper data preservation. The data-store objects possess an internal pointer that prevents the read operator from returning the wrong part of the data. The data and read functions are used to traverse the entire dataset and apply filters to obtain the intended subset. The primary use of map-reduce is to find the longest flight duration among all airplane datasets (see Table 3). To accomplish this:

1. The "mapper" function calculates the maximum time for each segment of the database.
2. The "reducer" function determines the maximum value among all the calculated maximum times from the "mapper" function.

Table 3: Implementation steps for Hadoop environment in K-Means clustering.

row	Description of doing work	Execution codes
1	step1: to enter the Hadoop environment and start it, first open the terminal and then execute the following commands.	Su hduser enter password Cd /usr/local/Hadoop/sbin ./start-all.sh
2	Step2: Create a directory for the 20newdata database and unzip the data	mkdir /tmp/20newdata cdtmp/20newdata tar-xzvf /tmp/20news-bydate.tar.gz
3	Step 3: There are two subfolders of the 20 new data folder called 20newdata-test and 20newdata-train. Create a folder called 20newsdataall and merge the test and train sections and go to the Home directory and execute the command	mkdir /tmp/20newdataall cd -R /20newsdata/*/*/tmp/20newsdataall
4	Step 4: Create a directory in Hadoop and save the data in HDFS:	hdfsdfs-mkdir /usr/hue/20newdata hdfsdfs -put /tmp/20newsdataall /usr/hue/20newsdata
5	Step 5: Execute the following command to see if put is done or not	hdfsdfs -ls /usr/hue/20newsdata/20newsdataall
6	Step 6: The Mahout accepts the data in vector format so we have to create the sequential file. For this purpose first In order for Mahout to read the data from Hadoop and not from the local file, we execute command 1 and then crate an order file using commands 2	1- Unset MAHOUT_LOCAL 2- export MAHOUT_HOME=/usr/local/mahout export PATH=PATH:MAHOUT_HOME/bin export MAHOUT_HEAPSIZE= 1000 MAHOUT_HOME bin/mahout seqdirectory - i/usr/hue/20newsdata/20newsdataall - o/usr/20newsdataseq-out
7	Step 7: Convert sequential file to vector First we change the hip size (command 1) then we execute command 2 nv Name : - wt The type of weight we used. Here we mean tfidf. - : .I norm is used for the output vector.	1- export MAHOUT_HEAPSIZE="2048" 2 - $MAHOUT_HOME/bin/mahout seq2sparde -i /usr/hue/20newsdataseq-out/part-m- ..-i/usr/hue/20newsdata/20newsdatasll -nv -lnorm - tfidf
8	Step 8: Execute clustering: K-Means We change the hip size (command (and then do command 2) input : –output output : –clusters The file contains the primary clusters : - num Clusters is the number of clusters that we considered 10 clusters. : -Distance Measure is the measured distance that we have used the Euclidean criterion. : -max Iter Maximum repetition that we considered in here 20. – :method is the type of method used, which is either series or .map-reduce. We used map-reduce here.	1- export MAHOUT_HEAPSIZE="2048" 2- $MAHOUT_HOME/bin/mahoutkmeans –input /usr/20newsdatavec/tfidf-vectors/ – output /usr/hue/Kmeansout –clusters /usr/hue /kmeanscener -numClusters1. V distanceMeasureorg.apache.mahout.common.distance. Euclidean Distance Measure e –maxIter 20 –method mapreduce-clustering
9	Step 9 : By executing the following command, we will see the output folder	hdfsdfs ls /usr/hue/Kmeansout
10	Step 10: The clustering output is displayed to us in sequential file. And we convert its foemat to readable. By executing the following command	$MAHOT_HOME/bin/mahoutclusterdump -i /usr/hue/Kmeansout/clusters-20-final - o/tmp/usr/clusterdumpout-d /usr/hue/20newsdatavec/dictionary.file-0 - dtsequencefile -.points Dir /usr/hue/Kmeansout/clusteredpoints -n 20 -b 100
11	Step 11: The cluster dump results are displayed in this section. To do this, open the Cluster dump file using the following command:	cat/usr/hue/clusterdumpout.txt

The use of keys in map-reduce causes the map function to add intermediate results to one or more packets referred to as "key". If the map function adds values to multiple keys, it results in several calls to the reduce function. For each airline found in the input data, the add function is employed to add a vector of values. This vector represents the number of flights for each airline over a span of 21 year. This approach ensures that all airline data is classified into single group, providing a comprehensive dataset for the reduce function.

Table 4: Comparing the K-Means Hadoop and K-Means Spark for the aviation dataset.

Number of points	Dimensions	Number of clusters	Hadoop time	Spark time
1000	3	7	63.2	41.5
1000	7	13	65.6	40.3
10000	3	7	61.8	39.9
10000	7	13	62.8	43.1
100000	3	7	43.1	48.7
100000	7	13	68.4	46.5

5.3 Results representation

First, we need to install the Hadoop and Spark platforms. To perform the installation, we start by installing VMware virtual machine on the system. Then, we install Linux operating system on the virtual machine. Finally, we run some commands in the operating system's terminal section (the specific commands are not provided in this section) to install the Hadoop platform. Since the project focuses on clustering, with the K-Means algorithm (see Table 2), we also need to install the Mahout platform. The algorithm is executed 10 times for each dataset, with a threshold set to 0.0001 and a maximum iteration limit of 50. Considering that the K-Means algorithm is sensitive to the initial centroids and we used random initialization, we will present the average execution time for the iterations. By observing the execution times in seconds obtained from various runs programs written in Spark and Hadoop, it is evident that Spark demonstrates significantly faster execution times compared to Hadoop. Additionally, the execution time can be further reduced by utilizing caching for persistence in the implementation of K-Means for Hadoop map-reduce and Spark for the aviation dataset. The following result can be observed (see Table 4):

1. For non-iterative aviation tasks, Spark initially performs better than Hadoop, but Hadoop catches up with Spark.
2. For iterative aviation tasks, Spark outperforms Hadoop significantly.
3. In all aviation tasks, enabling Spark's dynamic allocation leads to performance gains.

6 Conclusion

Due to the rapid growth of data, there is a need for new methods of storage and processing that are more efficient in terms of cost and time compared to traditional sequential approaches. For batch data processing and knowledge extraction from aviation Big data, we proposed the K-Means-map-reduce model on the Hadoop and Spark platforms for the aviation dataset in air traffic management. The K-Means-map-reduce model on the Hadoop platform reads and writes data from a disk, which slows down the processing speed. However, it is designed to handle batch processing efficiently. This model operates as a high-latency computing framework, without an interactive mode. On the other hand, the K-Means-map-reduce model on the Spark platform reduces the number of read/write cycles to disk by storing intermediate data in memory, resulting in faster-processing speed. This model is designed to handle real-time data efficiently, offering low-latency computing capabilities and interactive data processing. The results obtained demonstrate that applying the Spark platform to aviation Big data is suitable, considering the features of the aviation dataset. This study represents the first application of the proposed model in air traffic management systems, leading to the design of an application program. Future work can involve developing an application based on the presented model for air traffic management, and exploring the use of deep learning methods to enhance the model's capabilities.

Appendix

Appendix: Abbreviations and their descriptions.

Abbreviation	Description	Abbreviation	Description
ACP	Acceptance	CPL	Current flight plan
ADS-B	Automatic dependent surveillance broadcast	DEP	Flight departure message
ALR	Alerting	DLA	Flight delay message
ASP	Aircraft sequencing problem	EST	Estimate
ATC	Air traffic control	ETA	Estimated time of arrival
ARR	Flight arrival message	ETD	Estimated time of departure
ATFM	Air traffic flow management	FDS	Flight data system
ATM	Air traffic management	FPL	Filed flight plan
ATN	Aeronautical telecommunication network	IATA	International air transport association
CDN	Coordination	HDFS	Hadoop distributed file system
CHG	Flight modification message	KNN	K-nearest neighbor method
CNL	Flight cancellation message		

Declarations

Availability of supporting data

All data generated or analyzed during this study are included in this published paper.

Funding

No funds, grants, or other support were received for conducting this study.

Competing interests

The authors have no competing interests to declare that are relevant to the content of this paper.

Authors' contributions

The main manuscript text is written collectively by the authors.

References

[1] Bonyani, M., Soleymani, M. (2022). "Towards improving workers' safety and progress monitoring of construction sites through construction site understanding", arXiv preprint arXiv:2210.15760.

[2] Bonyani, M., Ghanbari, M., Rad, A. (2022). "Different gaze direction (dgnet) collaborative learning for iris segmentation", Available at SSRN 4237124.

[3] Bonyani, M., Rahmanian, M., Jahangard, S., Rezaei, M. (2023). "Dipnet: Driver intention prediction for a safe takeover transition in autonomous vehicles", IET Intelligent Transport Systems.

[4] Borne, K. (2014). "Top 10 big data challenges a serious look at 10 big data v's", Blog Post, 11.

[5] Burmester, G., Ma, H.,Steinmetz, D., Hartmannn, S. (2018). "Big data and data analytics in aviation", Advances in Aeronautical Informatics: Technologies Towards Flight, 4.0, 55-65.

[6] Del Río, S., López, V., Benítez, J.M., Herrera, F. (2014). “On the use of mapreduce for imbalanced big data using random forest”, Information Sciences, 285, 112-137.

[7] Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A. (2014). “A survey of clustering algorithms for big data: Taxonomy and empirical analysis”, IEEE Transactions on Emerging Topics in Computing, 2(3), 267-279.

[8] Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U. (2015). “The rise of “big data” on cloud computing: Review and open research issues”, Information Systems, 47, 98-115.

[9] Kasturi, E., Devi, S.P., Kiran, S.V., Manivannan, S. (2016). “Airline route profitability analysis and optimization using big data analyticson aviation data sets under heuristic techniques”, Procedia Computer Science, 87, 86-92.

[10] Li, R., Hu, H., Li, H., Wu, Y., Yang, J. (2016). “Mapreduce parallel programming model: A state-of-the-art survey”, International Journal of Parallel Programming, 44, 832-866.

[11] Minelli, M., Chambers, M., Dhiraj, A. (2013). “Big data, big analytics: Emerging business intelligence and analytic trends for today’s businesses”, 578, John Wiley & Sons.

[12] White, T. (2012). “Hadoop: The definitive guide”, O’Reilly Media, Inc.

[13] Zikopoulos, P., Eaton, C. (2011). “Understanding big data: Analytics for enterprise class Hadoop and streaming data”, McGraw-Hill Osborne Media.