In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, F. Banaei-Kashani, G. Trajcevski, R. H. Güting, L. Kulik, and S. Newsam, eds., pages 219–228, Chicago, IL, November 2019.*

Data Management and Analytics System for Online Flight Conformance Monitoring and Anomaly Detection

Samet Ayhan*
University of Maryland
College Park, Maryland
sayhan@cs.umd.edu

Hanan Samet
University of Maryland
College Park, Maryland
hjs@cs.umd.edu

ABSTRACT

Air Navigation Service Providers (ANSP) worldwide have been making a considerable effort for the development of a better method to monitor conformance to the planned routes and detect anomalies within a particular airspace. Conformance monitoring and anomaly detection are crucial for a better managed airspace, both strategically and tactically, yielding a higher level of automation and thereby reducing the air traffic controller's workload. Although the prior approaches with limited amount of static air traffic data have been able to address the problem to some extent, data management and query processing of ever-increasing vast volume of streaming air traffic data at high rates for online conformance monitoring and anomaly detection still remain a challenge. In this paper, we present a novel data management and analytics system to continuously conformance monitor flights and accurately detect anomalies within the National Airspace System (NAS). The incoming Traffic Flow Management (TFM) data is streaming, big, uncorrelated and noisy. In the overall data pipeline, the system monitors flights and detects anomalies in 3 steps: In the preprocessing step, the system continuously processes the incoming raw flight data and makes it available for the next step where an interim Key-Value data store is created and maintained for efficient query processing. In the final step, the system learns from historical trajectories and pertinent weather parameters and builds a Long Short-Term Memory (LSTM) model. As the flights progress, the non-conforming trajectory segments as part of the live data stream are raised as anomalies. Evaluations on real air traffic and weather data in the U.S. verify that our system efficiently and accurately detects anomalies.

CCS CONCEPTS

• **Information systems → Stream management**; • **Computing methodologies → Neural networks**; • **Applied computing → Aerospace**;

KEYWORDS

Big Data Analytics; Anomaly Detection; Air Traffic Management

*Senior Engineer at Boeing Research & Technology.

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6909-1/19/11…$15.00
https://doi.org/10.1145/3347146.3359378

1 INTRODUCTION

It is estimated that by 2040 the U.S. alone can expect an increase of more than 68% in commercial air traffic [3]. The FAA's Office of Performance Analysis estimates show in 2017, the cost of delayed flights increased by 11.3%, from $23.9 to $26.6 billion. Most of this increase was due to inefficiency in the National Airspace System (NAS), resulting in the sum of costs to airlines, passengers, lost demand, and direct costs [11]. The increasing inefficiency in the NAS along with the expected growth in commercial air traffic will require the expansion of Air Traffic Control (ATC) services together with the implementation of advanced Air Traffic Management (ATM) concepts such as dynamic resectorization, free flight, enhanced ground delay program, and airspace redesign, all with the goal of improving the efficiency of the National Airspace System (NAS) [2]. The expansion and deployment of these new concepts and services are expected to enable higher degrees of automation and predictability yielding a reduction in the air traffic controllers workload. Changing airspace configurations and air traffic patterns quantifies the controller's workload and is highly impacted by anomalies, non-conformance to established patterns or dissimilarities between the generated model's prediction and data observation. Despite the fact that previous research offered various solutions to the problem, *i)* they were limited with a particular dataset i.e. Flight Operations Quality Assurance (FOQA) data [1] that is exclusively available to the operating airline [25] or *ii)* they were designed to run offline on static data [30], or *iii)* they lacked scalability, i.e., they supported tracking a certain number of aircraft at a time [10], or *iv)* they enabled monitoring conformance only at a particular phase of a flight, i.e. enroute phase [10]. Overall, the previous efforts didn't offer a scalable solution, addressing online conformance monitoring and anomaly detection during the entire flight using streaming flight big data.

The FAA's TFM Data Service is a pure Java Message Service (JMS) and it provides near real-time streaming flight and flow data to users via System Wide Information Management(SWIM). The flight data includes flight plan, departure, arrival, track position, boundary crossing and flight management related messages in Flight Information Exchange (FIXM) format from each of the 20 Air Route Traffic Control Centers (ARTCC) systems. Although each message is assigned a unique flight identifier, they are uncorrelated, noisy, streaming at higher rates, and large in volume. Figure 1 shows the number of flight data messages based on their types processed throughout May, 2018. An average of 6 million flight messages is processed daily, where each flight plan and flight management (trajectory) related messages is formed by tens of position records, reaching up to over 20 million of processed points

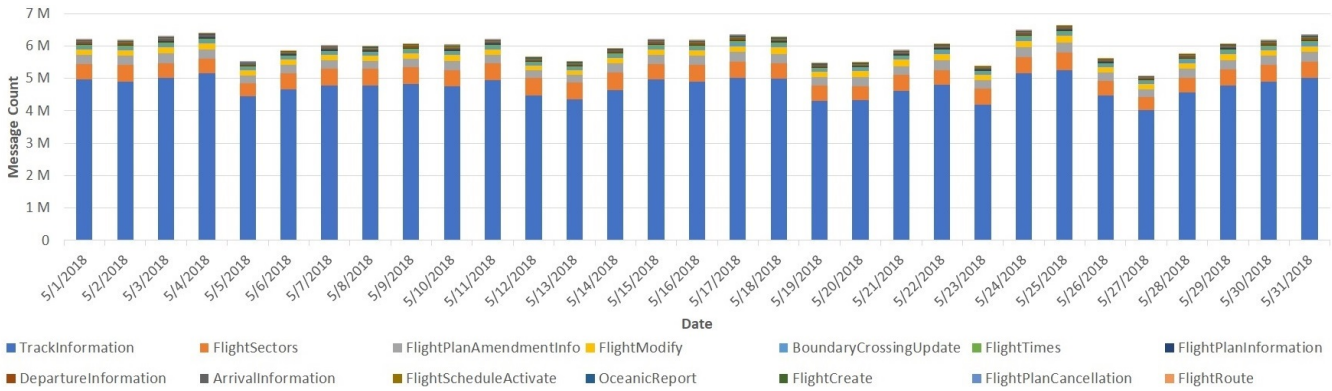


Figure 1: Counts of daily processed flight messages broken down by the message type over May 2018.

from over 100,000 unique flights per day. Clearly, for any analytics to be useful over this dataset a data management solution is required. Hence, in response to the online conformance monitoring and anomaly detection problem, we have implemented a novel data management and analytics system using the FAA's TFM big data. The system continuously processes streaming flight data by cleansing and correlating it. The processed data is maintained in a Key-Value data store for efficient query processing, where unique patterns are discovered during the descriptive analytics stage in the pipeline. Using historical trajectories, salient features are collected and fed into various LSTM models. The best performing model is adaptively selected and used for predictive analytics. The resulting anomalies in the NAS are presented to the user.

Although this paper presents a custom pipeline to address online conformance monitoring and anomaly detection problem, the proposed system is flexible to accommodate other pipelines in support of various use-cases in the aviation domain. With its unified storage and computing engine, the system offers an enhanced distributed computing paradigm based on MapReduce. As the main building block, the system utilizes Apache Spark [12], an open-source distributed computing framework. Although Spark achieves high-performance distributed computing, it poses limitations with its imperfect indexing mechanisms and inefficient runtime data persistence. Hence, the system complements Apache Spark with Apache Ignite [9], an in-memory distributed Key-Value store for efficient query processing and robust data management. In summary, the contributions of this paper are as follows:

- We propose an end-to-end flight big data management and analytics system that is flexible to accommodate custom pipelines for various aviation specific use cases.
- Our scalable system leverages the MapReduce paradigm complemented by distributed in-memory data store in support of efficient analytics query processing using TFM big data.
- Extensive experiments on massive real flight and weather data demonstrate that our system efficiently monitors conformance and detects anomalies in the NAS.

Our data management and analytics system can be used as a ground-based decision support tool by airlines and ANSPs. Accurate anomaly detection translates to an improved efficiency of the NAS, resulting in higher degrees of automation and thereby reduced amount of air traffic controller's workload. The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 introduces preliminary concepts followed by Section 4 where our data management and analytics system is elaborated. Section 5 presents the results of our experiments while concluding remarks are drawn in Section 6.

2 RELATED WORK

Trajectories are usually discussed for cars along roads [42] with an emphasis on queries (e.g., [35, 36, 39–41]). Here we are interested in aircraft trajectories. Big trajectory data management and analytics remains an active area of research both in the data management and data mining domains as well as in the transportation domain. Several systems have been proposed in the data management and data mining domains thus far. BerlinMOD [27], PIST [21], and TrajStore [24] offer a similar storage and indexing technique using traditional database engines. Simba [46] and SpatialHadoop [28] enable distributed spatial analytics based on the MapReduce programming model. Although they are capable of processing big trajectory data, they suffer from inefficiency due to high overhead rates of the underlying programming model. A column-oriented storage system SharkDB [48] offers efficient query processing and analytics capabilities. However, due to being a complete in-memory approach, it falls short when handling massive volumes of trajectory data. Unlike other systems, CloST [44], Elite [47], PARADASE [33], and a cloud based system [20] offer a specific partitioning technique in distributed environments for a potential of higher query processing efficiency. UlTraMan [26] is another system aiming to deliver efficient query processing in a distributed environment, with a more flexible partitioning approach. Nonetheless, it doesn't integrate the meta table construction into global indexes. There are a few other systems that support distributed storage and computing by integrating Apache Spark [12]. These systems include Apache GemFire [5], IndexRDD [13], and SnappyData [37]. Although these systems leverage Apache Spark in support of efficient streaming, transactions, and interactive analytics, they do not provide flexible operations and optimizations for trajectory data analytics.

In fact none of the above systems is specifically developed for management and analytics of flight big data. They are all generic purpose data management and analytics systems with no specific domain knowledge included in their design and implementation.

Hence, adapting these systems to our specific use case in the aviation domain requires considerable amount of extra effort. In contrast, our system has been specifically designed and optimized for efficient management and analytics of flight big data to address the online conformance monitoring and anomaly detection problem.

In their survey, Gupta et al. [31] present an organized overview of the various techniques proposed for outlier detection on temporal data. Chandola et al. [23] present an excellent survey to provide a broad overview of extensive research on anomaly detection techniques. They formulate the anomaly detection problem on the basis of several different factors. One way to categorize them is based on the nature of the input data. Input data can be discrete, categorical, or continuous. The attributes of input data can be univariate or multivariate. The input data can also be classified based on the relationship present among data instances, such as sequence data, spatial data, graph data, and profile data. Prior research [25, 30, 34] has extensively used FOQA data, which is a multivariate dataset formed by discrete and categorical input. Though, the data is collected by the operating airline and made proprietary. An important aspect of an anomaly detection technique is the nature of the desired anomaly. Anomalies can be classified into point, contextual, and collective anomalies. The labels associated with a data instance denote whether that instance is normal or anomalous. Based on the extent to which the labels are available, anomaly detection techniques can operate in supervised, semisupervised and unsupervised mode. Anomalies can also be detected based on classification techniques such as neural network-based, bayesian-network based, support vector machines-based, and rule-based. The rule-based approach is such that given the input and the predefined ranges, if the exceedance is observed an anomaly is reported [4]. Although simple and fast, this approach is limited since it examines each feature independently, omitting potential correlations among them. The concept of nearest neighbor analysis has been used in several anomaly detection techniques. Such techniques are based on the assumption that normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors. Li et al used a similar approach called ClusterAD [32]. Gorinevsky et al. [30] propose a method based on a custom linear regression model to describe the aerodynamic forces acting on an aircraft. However, the proposed model requires significant domain knowledge, thereby limiting its design. Das et al. [25] introduce the MKAD multiple kernel learning approach to detect potential safety anomalies in FOQA data. They pose a general anomaly detection problem which includes both discrete and continuous data streams, where they assume that the discrete streams have a causal influence on the continuous streams. They also assume that atypical sequences of events in the discrete streams can lead to off-nominal system performance. Although the results are promising, the algorithm lacks scalability since the kernel matrix needs to be updated for each test flight. Melnyk et al. [34] also focus on the aviation safety domain, where data objects are flights and time series are sensor readings and pilot switches. In this context the goal is to detect anomalous flight segments, due to mechanical, environmental, or human factors in order to identify operationally significant events and highlight potential safety risks. For this purpose, they propose a framework which represents each flight using a semi-Markov switching vector autoregressive (SMS-VAR) model. Detection of anomalies is then based on measuring dissimilarities between the model's prediction and data observation. They position SMS-VAR as a short-memory model and propose an anomaly detector that specifically targets short-term anomaly events. However, their analysis only focuses on a limited portion of the flight below 10000 ft until touchdown.

Although these efforts with a focus on aviation safety case yielded valuable research, a complete data management and analytics system for online conformance monitoring and anomaly detection has not yet been proposed. In fact, except for En Route Automation Modernization (ERAM) [10], none of the prior research in anomaly detection has ever used flight big data provided by the FAA's SWIM TFM Data Service, which is a non proprietary flight data source and is available to its subscribers. In 2015, the FAA has deployed ERAM at 20 FAA ARTCCs nationwide to monitor conformance of enroute flights. However, due to the estimated growth of commercial air traffic along with expected inclusion of massive number of unmanned aircraft in the NAS within the next decade, ERAM's limited support of conformance monitoring for only 1,900 flights for each control center is concerning. Our previous work [16] has aimed at addressing flight big data management and analytics. However, it used an obsolete data model, Aircraft Situation Display to Industry (ASDI) [6], which the FAA has stopped supporting. It also used RDBMS with a traditional database engine that suffered from inefficiency. Unlike the previous systems, our current proposed system uses the latest TFM data, provided by the FAA and is optimized for efficient storage and query processing of flight big data. With its unified platform based on Apache Spark, which is seamlessly integrated with a distributed, in-memory Key-Value store, the system achieves scalability, efficiency, persistence, and flexibility. The system continuously monitors conformance and accurately detects anomalies in the NAS.

3 BACKGROUND AND OVERVIEW

Our system spans across multiple domains including data management using high-performance distributed computing and data mining using neural networks, both towards addressing the conformance monitoring and anomaly detection in the aviation domain. Consequently, we introduce the underlying concepts, building blocks, and overview of our system, followed by the specifics of TFM and weather data.

3.1 Concepts

Definition 1. **Air traffic clearance** *means an authorization by ATC for an aircraft to proceed under specified traffic conditions within a controlled airspace for the purpose of preventing collision between known aircraft.*

Definition 2. **Anomalies** *are patterns in data that do not conform to a well defined notion of normal behavior.*

Definition 3. **Anomaly detection** *defines a region representing normal behavior, and declares an anomaly when observing a behavior in the data that does not belong to this normal behavior.*

Definition 4. **Conformance monitoring** *is an essential function in ATC to determine if aircraft are adhering to their assigned clearances so that safety, security, and efficiency can be maintained.*

In the context of ATM, assigned clearances are the regions representing normal behavior in the air traffic domain. However, the normal behavior, i.e. assigned clearances keep evolving based on ever

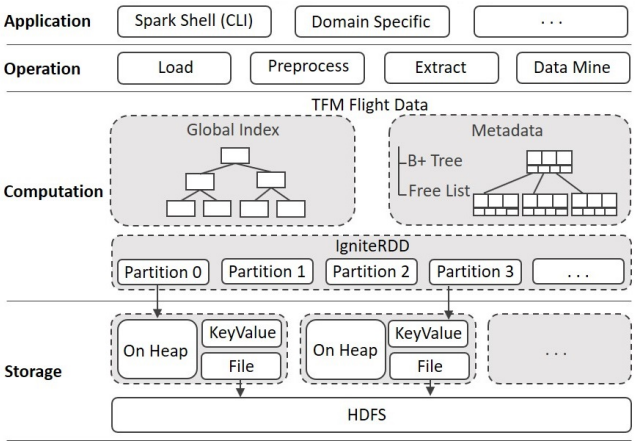


Figure 2: The high level system architecture.

changing factors such as air traffic and weather parameters. Moreover, it is unfeasible to communicate and compare each assigned clearance with the current position of each aircraft along their flight routes throughout the entire flight when massive amounts of flight data are considered. The current system used by the FAA, ERAM, monitors conformance of aircraft only during the enroute phase of a flight. Other flight phases such as climb and descent are monitored for conformance by towers and Terminal Radar Approach Controls (TRACON) and the pertinent data are stored in insulated silos. In contrast, our system continuously queries flight big data throughout the entire flight and discovers the normal behavior by performing descriptive analytics in aggregation. Next, the system adaptively builds various LSTM models based on evolving normal behavior and detect anomalies. To achieve these goals, the system utilizes a unified platform, made up of a number of building blocks that deliver scalability, efficiency, persistence, and flexibility.

3.2 Building Blocks

Apache Spark [12] is a distributed computing framework to address big data problems. In Spark, distributed computing tasks are submitted through an abstraction called Resilient Distributed Datasets (RDD), which is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. The tasks are physically conducted by executors, which are Java Virtual Machine (JVM) processes running on a worker node, that internally uses a block manager to manage its cached data partitions. Spark utilizes in-memory caching and recomputation-based fault tolerance to enable high-performance distributed computing. However, Spark's design trade-offs also introduce shortcomings such as an unexpected overhead due to massive on-heap caching, and high cost of recomputation. Hence, due to these limitations, we complement Spark with Apache Ignite.

Apache Ignite [9] is a memory-centric distributed database, caching, and processing platform, which is designed for transactional, analytical, and streaming workloads, delivering in-memory performance at scale. Combining Apache Spark with Ignite provides a number of significant benefits: *i)* true in-memory performance at scale can be achieved by avoiding data movement from a data source to Spark workers and applications, *ii)* RDD, DataFrame and SQL performance can be boosted, and *iii)* State and data can be more easily shared among Spark jobs. Ignite offers an implementation of the SparkRDD. This implementation allows any data and state to be shared in memory as RDDs across Spark jobs. This RDD is called IgniteRDD and it provides a shared, mutable view of the same data in-memory in Ignite across different Spark jobs, workers, or applications.

3.3 Overview

Our system is designed to have a master/slave architecture, formed by a driver node and multiple worker nodes. The driver node is responsible for orchestrating the tasks by assigning and scheduling them across the worker nodes. Hence, data storage and computation are distributed among the worker nodes. Figure 2 illustrates the high level system architecture. The system is comprised of four layers: *i) Storage layer* is the bottom layer that rests on HDFS, where all data and indexes are managed. When an RDD is cached, all data partitions in on-heap and off-heap Ignite instances are stored. Hence, both on-heap and off-heap instances can be randomly accessed, providing optimization for the layers that reside over this layer. *ii) Computation layer* is on top of Storage layer and is designed to support distributed computing. By utilizing IgniteRDD along with index and partition schema, distributed tasks can be scheduled at particular partitions. *iii) Operation layer* is located above the Computation layer and used for preprocessing operations such as Extract Load Transform (ETL) and postprocessing operations such as aggregation and mining. *iv) Application layer* is the top layer above the Operation layer and exposes a number of interfaces to interact with the system.

Figure 3 captures the data flow and processing steps within the overall system pipeline. The FAA's SWIM TFM data is provided through a set of Java Message Service (JMS) topics. Ignite JMS Data Streamer consumes from the pertaining topic, and streams the data into Ignite caches. The data is automatically partitioned between Ignite nodes and made available for preprocessing steps including transformation, cleaning, and correlation. Each preprocessing step is elaborated in Section 4. The preprocessed data in the form of a Key-Value documents is stored in HDFS, as cold flight data. The system extracts and collects the salient features, and periodically builds a new LSTM model using the stored data. As the new position records for each flight keep coming in, the system stores them in the Ignite cache and compares with the predicted values to detect anomalies, if any. Hence, the higher volume and the more recent the data, the more accurate the anomaly detection becomes.

3.4 TFM Data

The FAA's TFM Data Service is a pure JMS and it provides near real-time streaming flight and flow data to users via SWIM. The flow data includes reroutes, ground stops, ground delay programs, airspace flow programs, collaborative trajectory options program, etc. Although our system consumes and stores flow data type [8], our focus in this research is the flight data as in [17]. Figure 4 shows the high level TFM flight data schema as part of the TFM Data Service. The flight data includes *track information, flight plans, flight plan amendments, flight plan cancellations, departure and arrival time notifications, oceanic information, boundary crossings,* and *flight*

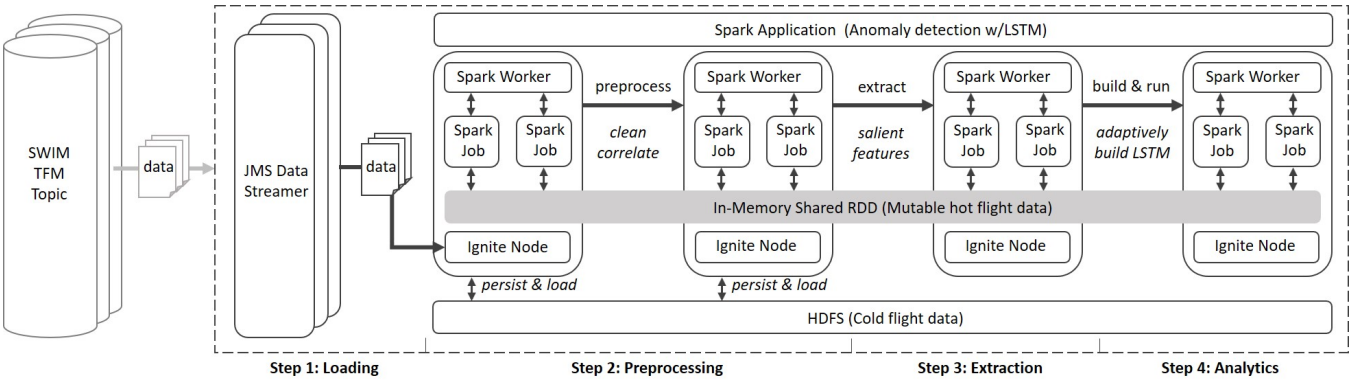


Figure 3: The overall system pipeline.

management information. Among these message types *track information, flight plan*, and *flight management information* are more critical for development of conformance monitoring and anomaly detection use case. The *track information* messages are used to provide a position record with a 1 minute update rate.

3.5 Weather Data

The source of the weather data is NOAA National Centers for Environmental Prediction (NCEP) Rapid Refresh (RAP) [7] which is the continental-scale NOAA hourly-updated assimilation/modeling system. RAP covers North America and is comprised primarily of a numerical forecast model and an analysis/assimilation system to initialize that model. It has 13 km horizontal resolution with 50 vertical levels. The RAP weather data is stored as a set of *grib2* [7] files each hour and made available for use by our system.

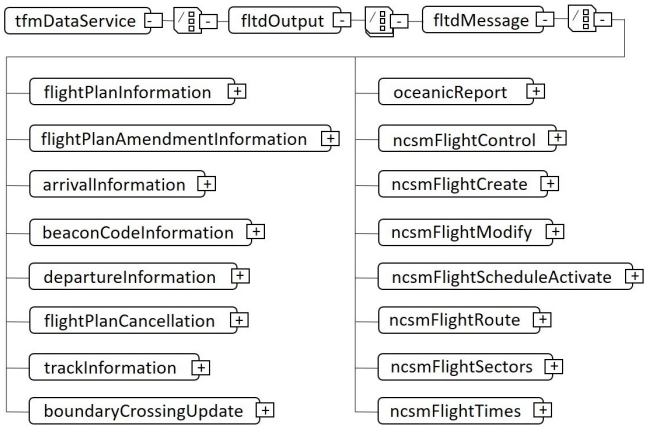


Figure 4: The high level TFM flight data schema.

4 DATA MANAGEMENT AND ANALYTICS SYSTEM

The system performs data management, where the streaming flight big data is ingested, processed and and stored, and *ii)* analytics, where the LSTM models are adaptively built and anomalous trajectory segments across the NAS are detected using neural networks.

4.1 Data Management

Data management tasks are performed across all steps in the system pipeline as illustrated in Figure 3. These steps include *i)* loading, *ii)* preprocessing, *iii)* extraction, and *iv)* analytics.

In the ***loading*** step, the streaming TFM flight big data provided by the FAA's SWIM Program is consumed of JMS topics by Ignite JMS Data Consumers. With the support of threading, consumers listen to a wide range of topics providing flight messages in different patterns, i.e., time-based or event based and concurrently consume them. The data is automatically partitioned in the cache across Ignite nodes and made available for preprocessing steps.

In the ***preprocessing*** step, the partitioned data across the Ignite nodes are made available for correlation. The purpose of the correlation process is to tag flight plans and track messages from multiple ATC centers relating to the same flight with a unique identifier. The Ignite JMS Data Streamer keeps consuming the TFM flight data without the knowledge of which messages are associated with each flight. This is partly due to lack of a global unique identifier used by ATC centers. To correlate position related messages with flight plans, the system builds a list of all active flights with a matching aircraft identifier, as well as departure and arrival airport. If there is only a single matching flight, the message is correlated with that flight. If there are multiple matching active aircraft identifiers, the flight plan with the closest matching last known position is correlated. Hence, index construction is included in this step.

As part of the cleaning process, string data type used for date and time objects are corrected by transforming them into `DateTime` data type. In addition, as diverse set of units and definitions are available for various flight data elements such as speed *(indicated air speed, true air speed, calibrated air speed, ground speed, etc.)* and altitude *(flight level, pressure altitude, absolute altitude, etc.)* in various units *(feet, miles, nautical miles, etc.)*, they are transformed to a single definition and measuring unit based on the element to ensure unity across the entire flight dataset. Next, the raw TFM flight data in mutable RDD is converted from XML to a Key-Value pair. The Key-Value pairs for *track information* type flight messages are transferred to HDFS as soon as a new position record is received for the same flight at the end of a 1 minute time interval. The same logic applies to other message types. Hence, historical *cold* flight data is persisted in HDFS.

date	time	speed	altitude	latitude	longitude	aircraftType	temperature	windSpeed	windDirection	humidity	pressure
20140811	17.58.37	218	42.4959	33.610303	-84.547098	B752	18.76200104	2.021774292	230.2754669	87.8347702	86902.66406
20140811	17.59.37	369	67.0758	33.597624	-84.610052	B752	14.55711842	3.984108448	281.4566345	79.17433167	79238.73438
20140811	18.00.37	248	103.2677	33.50593	-84.556668	B752	9.169438362	2.798942327	317.9927368	62.9315567	68961.70313

Figure 5: A sample feature set for a flight between Atlanta and NewYork City.

In the ***extraction*** step, the current unique airborne flights are identified based upon the streaming position reports received in IgniteRDD. Using these unique flights, two processes are performed in parallel: *1)* for each unique airborne flight, the pertinent historical trajectories are found in the *cold* flight data in HDFS. For each trajectory segment, salient features such as spatio-temporal data *(latitude, longitude, altitude,* and *timestamp)* and *aircraft type* are extracted. Using timestamps in the historical trajectory data, pertinent weather parameters such as *temperature, wind speed, wind direction, humidity,* and *pressure* are collected. This data extraction and collection is periodically performed as a Spark batch job on HDFS. *2)* For each new position report received for the current unique airborne flight, pertinent weather parameters are collected in IgniteRDD. Hence, at the end of the *extraction* step, a feature set is created for the current position report in the IgniteRDD, as well as for the historical trajectory segments in HDFS for the same unique airborne flight across the NAS. A sample feature set for flight between Atlanta and NewYork City is shown in Figure 5

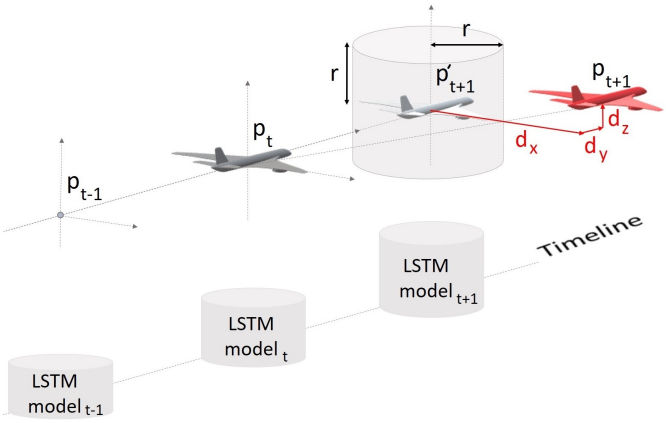


Figure 6: Illustration of anomaly detection.

4.2 Analytics

Upon collection of a salient feature set in the form of a time series, a predictive model is created for each unique airborne flight across the NAS. Our system uses LSTM model to predict the time series of aircraft positions. Now, we briefly review the LSTM.

Given a time series data $x_1, x_2, ..., x_t$, the Recurrent Neural Network (RNN) is defined by the following recurrent relation:

$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

where, $x_t \in \mathbb{R}^d$ is the input at time t, $W \in \mathbb{R}^{n \times d}$, $U \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ are the hidden state parameters, h_t and $h_{t-1} \in \mathbb{R}^n$ are the hidden state vectors at times t and $t-1$, respectively, and σ is the logistic sigmoid function. The main issue with the RNN model is that it can suffer from vanishing and exploding gradients. To address the vanishing/exploding gradient problem by backpropagating a constant error gradient, LSTM has been introduced. The LSTM defines a new hidden state, a *cell*. Each cell has its own state, *cell state*, which acts like a memory, and various control mechanisms, *gates*, enable modification of the cell memory. With these components, the LSTM can learn when to forget the old memories, *forget gate*, when to add new memories from the current input, *input gate*, and what memories to present as output from the current cell, *output gate*.

An aircraft trajectory is nothing more than a set of 3D positions in the form of a timeseries [15, 18, 19]. Hence, LSTM is well-suited for predicting the next position(s) of an aircraft, given the features for the current position and the features for the historically traversed positions. We use both Standard LSTM *(LSTM-STD)* and Sequential Encoder-Decoder LSTM *(LSTM-SED)* [43] in this work. To address the prediction problem, our system builds an LSTM model for each unique airborne flight based on multivariate and variable length historical trajectory data along with other salient features. As illustrated in Figure 6, the last position of the aircraft at time instance $t-1$ is p_{t-1}, the current position of the aircraft at time instance t is p_t, and the next position of the aircraft at time instance $t+1$ is p_{t+1}. As the flight progresses, a new LSTM model (LSTM-STD) is adaptively created at each time instance. Using the current LSTM model (LSTM-STD), $LSTMmodel_t$, the next position of the aircraft at time instance $t+1$ is predicted, which is p'_{t+1}. In the case of LSTM-SED, a new LSTM model is created at each k time instance and the next k positions of the aircraft are predicted as a batch process. Due to the fact that the distance value on x axis, d_x is greater than the configurable threshold value r, i.e., $d_x \geq r$, the system declares an anomaly. Note that our system assumes a virtual cylinder surrounding each aircraft, where the radius of the virtual cylinder is determined based on the norms computed at that time instance. Hence, upon receipt of a new position report for each unique airborne flight, the system computes the distance between the current aircraft position and the predicted aircraft position and declares an anomaly when any of the distance values (d_x, d_y, d_z) is greater than the norm (r), which is similar to computing a point in polygon algorithm [14] and declaring an anomaly when point doesn't reside in the surrounding virtual cylinder. This approach also relies on the assumption that the mean error of our prediction method is less than or equal to the norm $(\mu \leq r)$.

5 EVALUATION

In this section, we evaluate the performance, scalability, and accuracy of our system using real flight and weather data. We conduct experiments on data processing, extraction, and anomaly detection.

5.1 Setup

All experiments were conducted on a cluster of 12 nodes. Each node is equipped with two 12-core processors, 128GB RAM, and

Table 1: Statistics of the dataset.

Attribute	*FlightData*	*WeatherData*
Raw size	1.2TB	73.4GB
# Trajectories	26,624,018	-
# Points	4,216,048,661	4,216,048,661
Processed size	130,2GB	50.4GB
LSTM size	45.6GB	37.7GB

a Gigabit Ethernet. Each cluster node is equipped with a Ubuntu 14.03.3 LTS Operation System with Hadoop 2.7.1, Spark 2.3.0, and Ignite 2.6. In the cluster, one node is chosen as the master node and driver node, and the remaining nodes are slaves nodes. In each slave node, 64GB main memory is allocated for Ignite, of which 32GB is allocated for data storage and the remaining 32GB is allocated for temporary objects. We use the TFM Flight data provided by the FAA along with pertinent weather data provided by the NOAA [7] for the time period of 6 months (180 days, between 1 February and 31 July) in 2018 to test the performance of our system. Table 1 lists the statistics of the dataset for the duration of 6 months. The processed historical *cold* flight and weather data reside in HDFS. The online data including streaming new flight messages and LSTM model reside in the memory. The LSTM size refers to cumulative size. To evaluate the analytics capability of our system, we use the top 9 busiest routes in the U.S. and compute the mean error as the Root Mean Square Error (RMSE) for each route. Table 2 shows the airports with their International Civil Aviation Organization (ICAO) codes, that form these routes.

Table 2: A set of airports with their ICAO codes.

AirportCode	*AirportName*
KATL	Hartsfield-Jackson Atlanta Int'l Airport
KDCA	Reagan National Airport
KJFK	John F. Kennedy Int'l Airport
KLAX	Los Angeles Int'l Airport
KLGA	LaGuardia Airport
KMCO	Orlando Int'l Airport
KMIA	Miami International Airport
KMSP	Minneapolis Saint Paul Int'l Airport
KORD	Chicago O'Hare International Airport
KSFO	San Francisco Int'l Airport

5.2 Results

We first evaluate the running times of various preprocessing tasks in our system. To show the performance of our system on different data sizes, we split the same dataset into 2 equal parts and present the results for *i)* the dataset of 3-months (between 1 February 2018 and 30 April 2018) and *ii)* the dataset of 6-months (the entire dataset). Table 3 shows the results as an average of 100 random queries for these techniques on TFM flight and weather dataset of 3 months and 6 months in 2018.

Note that *Partitioned* mode in Ignite is the most scalable distributed cache mode and data is split equally into partitions, creating a massive distributed store. With this approach, our system is able to store all flight and weather data in the total memory (RAM and disk). Unlike *Replicated* mode, updates are relatively less expensive due to fact only one primary node needs to be updated for every key. However, reads are relatively more expensive due to the fact that only certain nodes have the data cached. To avoid extra data movement, it is critical to access the data exactly on the node that has the particular data cached. Although persistence is optional in Ignite, we persist the *cold* flight and weather data in HDFS. When turned off, Ignite becomes a pure in-memory store. Our system uses a multi-field index that allows accelerating queries with complex conditions.

Table 3: Preprocessing times.

Preprocessing Task	*Time(sec) ((1/2)Dataset)*	*Time(sec) (FullDataset)*
Partitioning	34.546	50.721
Indexing	0.197	0.318
Key-Value Persistence	5.651	14.543
Data Extraction	3.026	6.523

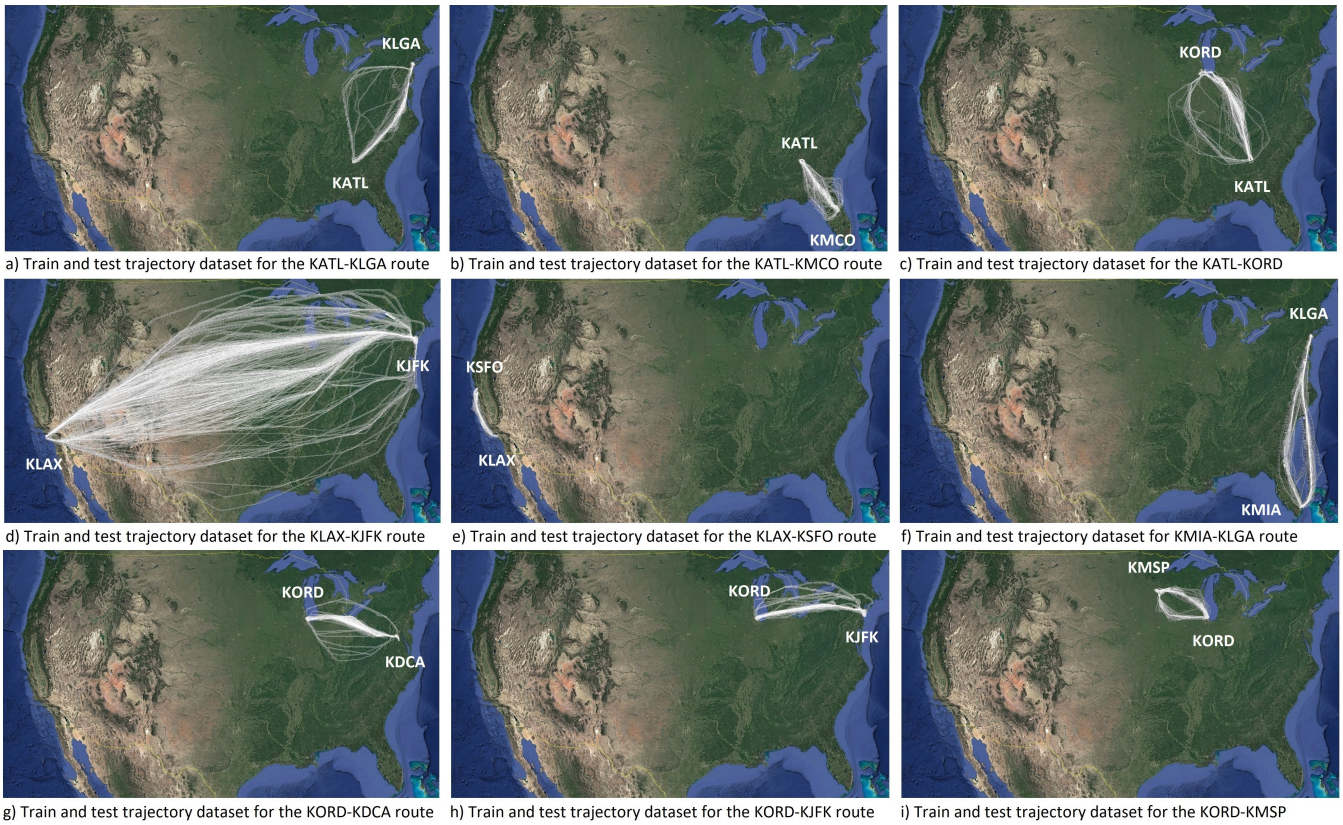
We can make a few remarks based on the preprocessing times of various tasks on various data sizes. *i)* The largest figures in the table are for partitioning. Although it is an automated process in Ignite, it still is a major time consuming task. *ii)* The running time for partition for different dataset sizes shows that it is proportional to the dataset size. *iii)* The Key-Value Persistence is relatively slow when the dataset size is doubled, and this is because additional time is needed to serialize the data to be stored in off-heap memory when the entire data doesn't fit in memory. *iv)* The indexing time is the smallest figure among all preprocessing times.

Next, we evaluate the running times of ID queries on the same dataset, measuring the query throughput based on a number of operations per second and the query latency in milliseconds. As previously employed, we range-partition the dataset based on two date ranges of the first 3 months and the entire dataset, and present the results for the query performance as an average of 100 random queries. Table 4 shows the ID query performance on TFM flight and weather dataset of 3 months and 6 months in 2018.

Table 4: ID query performance.

DatasetSize	*Throughput (Ops/sec)*	*Latency (ms)*
(1/2) of Dataset	17,422	3.11
Full Dataset	9,128	4.47

The linearly decreasing pattern is obvious based on the throughput figures for the first half of the dataset versus the entire dataset in Table 4. This is because the query is executed over partitioned data, i.e., the query is parsed and split into multiple map queries and a single reduce query. All map queries are executed on all the nodes where required data resides. All the nodes provide result sets to the query initiator (reducer), where the reduce phase is performed by merging provided result sets. Although we may not be able to make the same comment for the latency, i.e., it doesn't linearly scale with respect to the data size, the latency of ID queries on the off-heap



a) Train and test trajectory dataset for the KATL-KLGA route
b) Train and test trajectory dataset for the KATL-KMCO route
c) Train and test trajectory dataset for the KATL-KORD
d) Train and test trajectory dataset for the KLAX-KJFK route
e) Train and test trajectory dataset for the KLAX-KSFO route
f) Train and test trajectory dataset for KMIA-KLGA route
g) Train and test trajectory dataset for the KORD-KDCA route
h) Train and test trajectory dataset for the KORD-KJFK route
i) Train and test trajectory dataset for the KORD-KMSP

Figure 7: Illustration of the training and test trajectories for the top 9 busiest routes in the U.S.

persisted IgniteRDDs increases faster than those on the on-heap IgniteRDDs. Moreover, data persisted on Key-Value store can be queried faster than that persisted on off-heap.

Finally, we evaluate the analytics capability of our system by following a two-step process: *i)* we randomly replace a set of original trajectory points with anomalous trajectory points in the TFM flight data for the top 9 busiest routes in the U.S., *ii)* we generate a confusion matrix and compute the accuracy, precision, recall, and F1 values for our system. However, in order for our system to detect anomalous trajectory segments, i.e. declare anomaly, it should accurately predict the next position of an aircraft based upon the LSTM model built by historical data. Hence, first we present the trajectory prediction accuracy of our system. To do that, we use TFM flight data along with weather data for the top 9 busiest routes in the U.S. and build an LSTM model for each route. Figure 7 shows the training and test trajectories for the top 9 busiest routes in the U.S. As the flight progresses, the system continuously receives and processes streaming flight data including position information for the top 9 busiest routes in the U.S. The system uses historical flight and weather data to build an LSTM model for each route. As elaborated in Section 4.2, our system uses both the Standard LSTM (LSTM-STD) and Sequential Encoder-Decoder LSTM (LSTM-SED). In the case of LSTM-STD, the system predicts the next 3D position of each aircraft in the same route. In the case of LSTM-SED, the system predicts k number of next 3D positions at each k number of time steps. In either case, the system defines each LSTM model with 50 neurons in the first hidden layer and 3 neurons in the output layer for predicting 3D position(s). The system uses the Mean Absolute Error (MAE) loss function and the efficient Adam version of stochastic gradient descent. The model is fit for 100 training epochs with a batch size of 72. Although, our evaluation uses both LSTMs, we present results for the LSTM-STD, as it reaches higher accuracy with the cost of increased running times.

To assess the accuracy of each LSTM model, we analyze their behavior over the epochs while they are built. The training history for each LSTM model can be used to diagnose the behavior of the model. In the case of an overfit model, the train loss slopes down and the validation loss slopes down, hits an inflection point, and starts to slope up again. In the case of an underfit model, the training loss is lower than the validation loss, and the validation loss has a trend that suggests further improvements are possible. As illustrated in Figure 8, an overall good fit can be observed on all routes, where the train and validation loss decrease and stabilize around the same point. The only exception may be considered Figure 8d, where test loss drops below train loss at particular positions on the x-axis. Hence, the model may be slightly overfitting the training data which may be due to a high variance. We can easily verify that on Figure 7d, where wide range of lateral flight patterns between KLAX and KJFK is observed.

Next, we compute the trajectory prediction mean error of our system, by comparing the predicted next position of an aircraft with the ground truth, i.e. the actual next position of an aircraft.

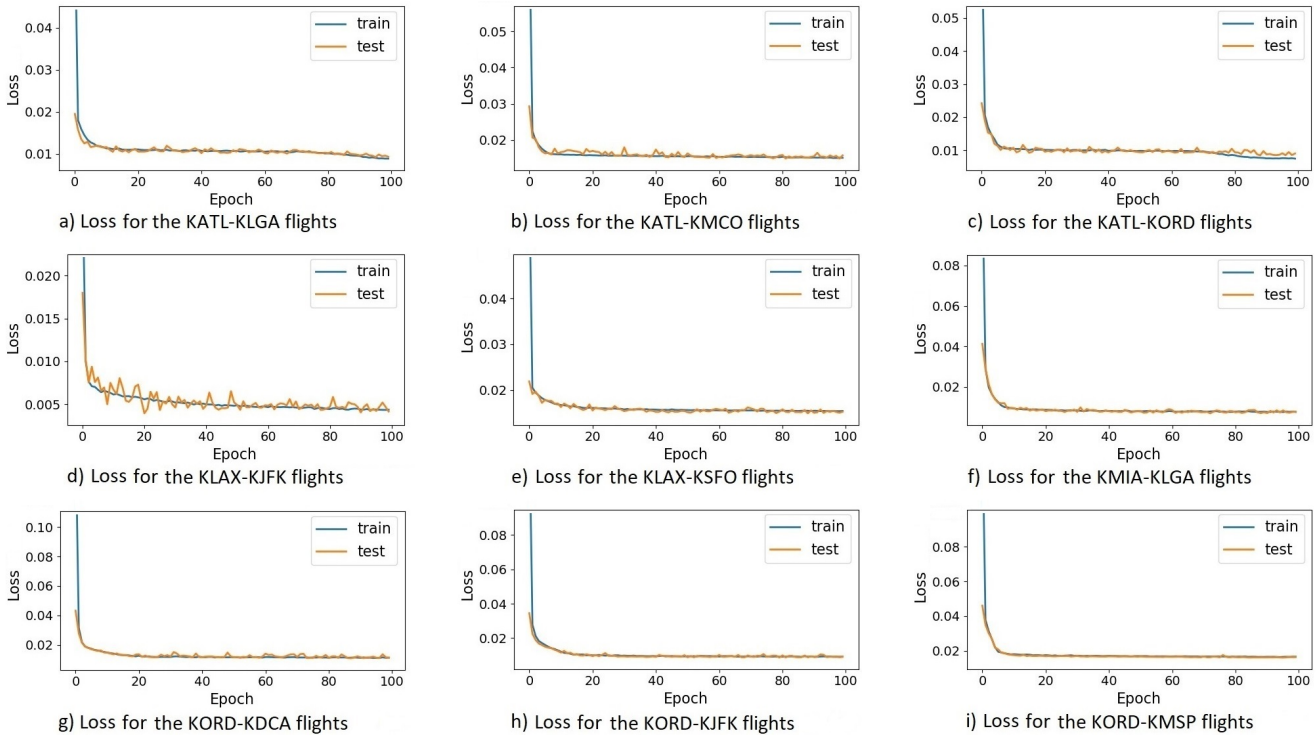


Figure 8: Illustration of LSTM loss on the training and test datasets for the top 9 busiest routes in the U.S.

Note that, lateral prediction error is formed by two components, prediction error in *latitude* and prediction error in *longitude*. The vertical prediction error is computed by comparing the predicted next altitude of aircraft with the actual next altitude of aircraft. Table 5 shows the top 9 busiest routes in the U.S. with their lateral position prediction mean error in nautical miles (nmi) and vertical position (altitude) prediction mean error in feet (ft) by LSTM. To evaluate the anomaly detection capability of our system, we first bootstrap by drawing 100,000 trajectories with replacement from a total of 3,461,122 historical trajectories for the top 9 busiest routes in the U.S. and replace an existing trajectory point with an anomalous trajectory point. This is to generate a set of test cases with a virtual ground truth to test our system. The process allows us to plot 95% confidence interval for the mean error of our trajectory samples. Next, we run our system to see how many of those anomalous trajectory points are correctly and incorrectly found. We present the results in a confusion matrix as shown in Table 6. Given the results presented in Table 6, the first observation is that our system is able to detect 97,892 anomalies and unable to detect 2,108 anomalies out of 100,000 virtually created anomalies. The reason for the 2,108 cases where our system was unable to find anomaly is due to the accuracy of LSTM model built at that particular time instance, as the LSTM model is adaptively built at each time instance. The second observation is that our system detects 37 anomalies when none of them is one of the 100,000 virtually created anomalies. However, a close analysis reveals that although these 37 anomalies don't belong to the virtually created anomaly set, they are indeed anomalies. Though, in our confusion matrix, we still consider them as False Positives. With these True Positive, True Negative, False Positive and False Negative counts, our system reaches 99.9%, 100.0%, 97.9%, and 98.9% accuracy, precision, recall, and F1 values, respectively. In addition, we also build an LSTM-SED for each route once every 5 minutes and compare the outcome with the LSTM-STD's. The results verify a slightly degraded anomaly detection accuracy in exchange of decreased cost of running times by the LSTM-SED.

Table 5: 3D position prediction mean errors by LSTM for the top 9 busiest routes in the U.S.

Route	*Latitude RMSE(nmi)*	*Longitude RMSE(nmi)*	*Altitude RMSE(ft)*
KATL-KLGA	23.54	27.82	520.6
KATL-KMCO	27.88	19.03	769.3
KATL-KORD	26.58	11.43	758.3
KLAX-KJFK	9.46	33.22	925.8
KLAX-KSFO	19.51	17.34	499.5
KMIA-KLGA	31.38	19.86	612.8
KORD-KDCA	14.56	30.43	627.7
KORD-KJFK	7.62	27.83	695.6
KORD-KMSP	15.65	20.21	791.4

6 CONCLUSION

We have presented a novel data management and analytics system that continuously conformance monitors flights and accurately detects anomalies using streaming TFM flight big data. The system achieves high-performance, low latency, and linear scalability while

Table 6: Confusion matrix.

		Predicted		
		Anomaly	No anomaly	Total
Actual	Anomaly	97,892	2,108	100,000
	No anomaly	37	3,361,085	3,361,122
	Total	97,929	3,363,193	3,461,122

accurately detecting anomalies in the NAS. Our system can be used as a ground-based decision support tool by airlines and ANSPs. An accurate anomaly detection translates to an improved efficiency of the NAS, resulting in higher degrees of automation and thereby reduced amount of air traffic controller's workload.

Some future work could involve adding a spatial browsing capability [22, 29, 38] for the trajectories as well as incorporating our methods in a distributed spatial environment [45].

7 ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation of the US under grant IIS-1816889.

REFERENCES

[1] 2004. Flight Operations Quality Assurance (FOQA). https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_120-82.pdf. (2004). Accessed: 2019-01-19.
[2] 2009. FAA TFM. https://www.fly.faa.gov/Products/Training/Traffic_Management_for_Pilots/TFM_in_the_NAS_Booklet_ca10.pdf. (2009). Accessed: 2019-01-19.
[3] 2015. FAA TAF. http://taf.faa.gov/Downloads/TAFSummaryFY2015-2040.pdf. (2015). Accessed: 2019-01-19.
[4] 2015. Performance Data Analysis and Reporting System. https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/systemops/perf_analysis/perf_tools/. (2015). Accessed: 2019-01-19.
[5] 2016. Apache Geode. https://geode.apache.org. (2016). Accessed: 2019-01-19.
[6] 2016. FAA ASDI. http://www.fly.faa.gov/ASDI/. (2016). Accessed: 2019-01-19.
[7] 2016. NCEP WMO GRIB2 Documentation. http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml. (2016). Accessed: 2019-01-19.
[8] 2017. Flow Information Data. https://cdm.fly.faa.gov/?page_id=2321. (2017). Accessed: 2019-01-19.
[9] 2018. Apache Ignite. https://ignite.apache.org. (2018). Accessed: 2019-01-19.
[10] 2018. En Route Automation Modernization. https://www.faa.gov/air_traffic/technology/eram/. (2018). Accessed: 2019-01-19.
[11] 2018. FAA ATO. https://www.faa.gov/air_traffic/by_the_numbers/media/Air_Traffic_by_the_Numbers_2018.pdf. (2018). Accessed: 2019-01-19.
[12] 2019. Apache Spark. https://spark.apache.org/. (2019). Accessed: 2019-01-19.
[13] 2019. Indexrdd. https://github.com/amplab/spark-indexedrdd. (2019). Accessed: 2019-01-19.
[14] S. Ayhan, P. Comitz, and R. LaMarche. 2008. Implementing Geospatially Enabled Aviation Web Services. In *2008 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. Herndon, VA, 1–8.
[15] S. Ayhan, P. Costas, and H. Samet. 2018. Prescriptive Analytics System for Long-Range Aircraft Conflict Detection and Resolution. In *Proc. of the 26th ACM SIGSPATIAL Int'l Conference on Advances in GIS*. Seattle, WA, 239–248.
[16] S. Ayhan, J. Pesce, P. Comitz, G. Gerberick, and S. Bliesner. 2013. Predictive Analytics with Aviation Big Data. In *Proc. of the 2013 Integrated Communications Navigation and Surveillance (ICNS)*. 1–13.
[17] S. Ayhan and H. Samet. 2015. DICLERGE: Divide-Cluster-Merge Framework for Clustering Aircraft Trajectories. In *Proc. of the 8th ACM SIGSPATIAL IWCTS*. Seattle, WA, 7–14.
[18] S. Ayhan and H. Samet. 2016. Aircraft Trajectory Prediction Made Easy with Predictive Analytics. In *Proc. of the 22nd ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, 21–30.
[19] S. Ayhan and H. Samet. 2016. Time Series Clustering of Weather Observations in Predicting Climb Phase of Aircraft Trajectories. In *Proc. of the 9th ACM SIGSPATIAL IWCTS*. San Francisco, CA, 25–30.
[20] J. Bao, R. Li, X. Yi, and Y. Zheng. 2016. Managing Massive Trajectories on the Cloud. In *Proc. of the 24th ACMGIS*. Burlingame, CA, 41:1–41:10.
[21] V. Botea, D. Mallett, M. A. Nascimento, and J. Sander. 2008. PIST: An Efficient and Practical Indexing Technique for Historical Spatio-Temporal Point Data. *Geoinformatica* 12, 2 (June 2008), 143–168.
[22] F. Brabec and H. Samet. 2007. Client-based spatial browsing on the world wide web. *IEEE Internet Computing* 11, 1 (January/February 2007), 52–59.
[23] V. Chandola, A. Banerjee, and V. Kumar. 2009. Anomaly Detection: A Survey. *ACM Computing Survey* 41, 3 (July 2009), 15:1–15:58.
[24] P. Cudre-Mauroux, E. Wu, and S. Madden. 2010. TrajStore: An Adaptive Storage System for Very Large Trajectory Data Sets. In *Proc. of the 26th ICDE*. Long Beach, CA, 109–120.
[25] S. Das, B. L. Matthews, A. N. Srivastava, and N. Oza. 2010. Multiple Kernel Learning for Heterogeneous Anomaly Detection: Algorithm and Aviation Safety Case Study. In *Proc. of the 16th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. Washington, DC, 47–56.
[26] X. Ding, L. Chen, Y. Gao, C. S. Jensen, and H. Bao. 2018. UlTraMan: A Unified Platform for Big Trajectory Data Management and Analytics. *Proc. of the VLDB Endowment* 11, 7 (March 2018), 787–799.
[27] C. Düntgen, T. Behr, and R. Güting. 2009. BerlinMOD: A Benchmark for Moving Object Databases. *The VLDB Journal* 18, 6 (December 2009), 1335–1368.
[28] A. Eldawy and M. F. Mokbel. 2015. SpatialHadoop: A MapReduce Framework for Spatial Data. In *Proc. of the 31st ICDE*. Seoul, South Korea, 46–50.
[29] C. Esperança and H. Samet. 2002. Experience with SAND/Tcl: a scripting tool for spatial databases. *Journal of Visual Languages and Computing* 13, 2 (April 2002), 229–255.
[30] D. Gorinevsky, B. Matthews, and R. Martin. 2012. Aircraft anomaly detection using performance models trained on fleet data. In *Proc. of the 2012 Conference on Intelligent Data Understanding*. Boulder, CO, 17–23.
[31] M. Gupta, J. Gao, C. Aggarwal, and J. Han. 2014. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge & Data Engineering* 26, 1 (September 2014), 15:1–15:58.
[32] Lishuai Li, Santanu Das, R. John Hansman, Rafael Palacios, and Ashok N. Srivastava. 2015. Analysis of Flight Data Using Clustering Techniques for Detecting Abnormal Operations. *Journal of Aerospace Information Systems* 12, 9 (March 2015), 587–598.
[33] Q. Ma, B. Yang, W. Qian, and A. Zhou. 2009. Query Processing of Massive Trajectory Data Based on Mapreduce. In *Proc. of the First Int'l Workshop on Cloud Data Management*. Hong Kong, China, 9–16.
[34] I. Melnyk, A. Banerjee, B. Matthews, and N. Oza. 2016. Semi-Markov Switching Vector Autoregressive Model-Based Anomaly Detection in Aviation Systems. In *Proc. of the 22nd ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, 1065–1074.
[35] S. Nutanong and H. Samet. 2013. Memory-efficient algorithms for spatial network queries. In *Proc. of the 29th ICDE*. Brisbane, Australia, 649–660.
[36] S. Peng, J. Sankaranarayanan, and H. Samet. 2016. SPDO: High-Throughput Road Distance Computations on Spark Using Distance Oracles. In *Proc. of the 32nd ICDE*. Helsinki, Finland, 1239–1250.
[37] J. Ramnarayan, S. Menon, S. Wale, and H. Bhanawat. 2016. SnappyData: A Hybrid System for Transactions, Analytics, and Streaming: Demo. In *Proc. of the 10th ACM Int'l Conference on Distributed and Event-based Systems*. Irvine, CA, 372–373.
[38] H. Samet, H. Alborzi, F. Brabec, C. Esperança, G. R. Hjaltason, F. Morgan, and E. Tanin. 2003. Use of the SAND spatial browser for digital government applications. *Commun. ACM* 46, 1 (January 2003), 63–66.
[39] J. Sankaranarayanan, H. Alborzi, and H. Samet. 2006. Distance Join Queries on Spatial Networks. In *Proc. of the 14th ACMGIS*. Arlington, VA, 211–218.
[40] J. Sankaranarayanan and H. Samet. 2009. Distance oracles for spatial networks. In *Proc. of the 25th ICDE*. Shanghai, China, 652–663.
[41] J. Sankaranarayanan and H. Samet. 2010. Query processing using distance oracles for spatial networks. *IEEE Transactions on Knowledge and Data Engineering* 22, 8 (August 2010), 1158–1175.
[42] J. Sankaranarayanan and H. Samet. 2010. Roads belong in databases. *IEEE Data Engineering Bulletin* 33, 2 (June 2010), 4–11.
[43] N. Srivastava, E. Mansimov, and R. Salakhutdinov. 2015. Unsupervised Learning of Video Representations Using LSTMs. In *Proc. of the 32nd Int'l Conference on Machine Learning*. Lille, France, 843–852.
[44] H. Tan, W. Luo, and L. M. Ni. 2012. CloST: A Hadoop-based Storage System for Big Spatio-temporal Data Analytics. In *Proc. of the 21st ACM CIKM*. Maui, HI, 2139–2143.
[45] E. Tanin, A. Harwood, and H. Samet. 2005. A distributed quadtree index for peer-to-peer settings. In *Proc. of the 21st ICDE*. Tokyo, Japan, 254–255.
[46] D. Xie, F. Li, B. Yao, G. Li, L. Zhou, and M. Guo. 2016. Simba: Efficient In-Memory Spatial Analytics. In *Proc. of the 2016 Int'l Conference on Management of Data*. San Francisco, CA, 1071–1085.
[47] X. Xie, B. Mei, J. Chen, X. Du, and C. S. Jensen. 2016. Elite: An Elastic Infrastructure for Big Spatiotemporal Trajectories. *The VLDB Journal* 25, 4 (August 2016), 473–493.
[48] B. Zheng, H. Wang, K. Zheng, H. Su, K. Liu, and S. Shang. 2018. SharkDB: An In-memory Column-oriented Storage for Trajectory Analysis. *World Wide Web* 21, 2 (March 2018), 455–485.