Towards a Scalable Architecture for Flight Data Management

Iván García[1], Miguel A. Martínez-Prieto[2], Aníbal Bregón[2], Pedro C. Álvarez[1,3] and Fernando Díaz[2]
[1]*Instituto de Investigación en Matemáticas (IMUVA), Universidad de Valladolid, 47011 Valladolid, Spain*
[2]*Departamento de Informática, Universidad de Valladolid, 40005 Segovia, Spain*
[3]*Departamento de Estadística e Investigación Operativa, Universidad de Valladolid, 47011 Valladolid, Spain*

Keywords: Data Lake, ADS-B, Big Data, Flight Analytics, Hadoop.

Abstract: The dramatic growth in the air traffic levels witnessed during the last two decades has increased the interest for optimizing the Air Traffic Management (ATM) systems. The main objective is being able to cope with the sustained air traffic growth under safe, economic, efficient and environmental friendly working conditions. The ADS-B (Automatic Dependent Surveillance - Broadcast) system is part of the new air traffic control systems, since it allows to substitute the secondary radar with cheaper ground stations that, at the same time, provide more accurate real-time positioning information. However, this system generates a large volume of data that, when combined with other flight-related data, such as flight plans or weather reports, faces scalability issues. This paper introduces an (on-going) Data Lake based architecture which allows the full ADS-B data life-cycle to be supported in a scalable and cost-effective way using technologies from the Apache Hadoop ecosystem.

1 INTRODUCTION

There is an increasing interest for optimizing air traffic management (ATM) to deal with the fast growing number of flight movements around the world. For instance, EUROCONTROL predicts that there will be 11.6 ± 1.2 million flight movements in the European airspace by 2023, 14% more than in 2016 (Eurocontrol, 2017). It is a fact that the airspace is getting more and more congested, and aviation authorities work to modernize air traffic control (ATC) systems.

ADS-B *(Automatic Dependent Surveillance-Broadcast)* is one of the core technologies of the new generation of ATC systems and is designed to improve the safety, capacity, and efficiency of airspaces. ADS-B enables flight trajectories to be tracked by using a Global Navigation Satellite System (GNSS) instead of traditional radar communications. Although airspace systems are still in the transition period to incorporate ADS-B technology, the corresponding equipment will be mandatory for some aircraft in Europe by the end of the current 2017 (European Commission, 2011). Nevertheless, some vehicles have been already equipped with ADS-B transponders and broadcast messages with information about the flight trajectory. These time-stamped messages comprise the 24-bit aircraft *hex code*, the flight *callsign*, geolocation, altitude, or speed.

ADS-B messages are broadcasted twice per second (Strohmeier et al., 2015), so large amounts of data are streamed during the flight. It is obvious that potential scalability issues arise when ATM systems deal with data from many flights (e.g. all flights within a particular airspace over a period of time). In this work, we focus on ATM systems that preserve ADS-B data for subsequent processing. In this case, large streams of ADS-B messages must be ingested into the system and stored for different types of analytics. It is usual that these analyses demand complementary data to enhance trajectory information (e.g. flight-plans, weather, baggage ticketing, etc.). As a result, huge repositories of heterogeneous data are consolidated and ATM systems must deal with them to make the corresponding decisions and forecasts. Thus, implementing ATM systems is a particular case of exploiting Big Data for flight-related analytics.

This paper describes our *in-progress* experience to deploy a Big Data platform for ATM. More specifically, we propose a scalable data-center architecture which implements all flight data life-cycle, from raw data acquisition to highly-refined data load into an end-user system. This class of solutions, commonly referred to as *Data Lake* (Terrizzano et al., 2015), is designed as a centralized repository allowing large amounts of (structured and/or unstructured) raw data to be ingested and stored for subsequent transformation and/or integration purposes. All data manipulations are continuously tracked by a data governance

García, I., Martínez-Prieto, M., Bregón, A., Álvarez, P. and Díaz, F.
Towards a Scalable Architecture for Flight Data Management.
DOI: 10.5220/0006473402630268
In *Proceedings of the 6th International Conference on Data Science, Technology and Applications (DATA 2017)*, pages 263-268
ISBN: 978-989-758-255-4

component to ensure high-quality data to be finally delivered for particular analytics.

Our main contribution in this paper is **AIRPORTS DL**, a data lake oriented architecture which enables massive of ADS-B data collections to be efficiently managed and integrated with other flight-related data. The AIRPORTS DL components are designed for scalable data management along its full life-cycle, ensuring high-quality data to be delivered for flight analytics. AIRPORTS DL is implemented by a Hadoop-based ATM system which exploits flight-related data to reconstruct gate-to-gate trajectories and to derive parameters, such as the predictability or the fuel consumption, that can be later used to measure and ultimately improve the flight efficiency. This is our main objective in this project.

The rest of the paper is organized as follows. Section 2 delves into more detail about ADS-B and reviews some use-cases using this technology. Then, Section 3 provides a detailed description of the Data Lake architecture proposed in this paper. Finally, Section 4 concludes about our current achievements and explains our next steps in this project.

2 ADS-B

The *Automatic Dependent Surveillance* (ADS) is a surveillance technology that allows aircraft to automatically provide, through data link, the data extracted from the on board navigation and position determination systems (Blythe et al., 2011). ADS has three main features: *i)* it is *automatic*, since it requires no intervention from any operator (such as the pilot) or external input to transmit data; *ii)* it is *dependent*, since the data generation depends on the on board aircraft system (such as the navigation system); and *iii)* it provides *surveillance* data similar to radar data for both the ground controllers and other aircraft.

ADS provides technologies for *broadcasting* (ADS-B) and *contract* (ADS-C). ADS-B automatically and periodically transmits on board parameters to all possible receivers within its range of influence (it can transmit to both a ground station or to other suitably equipped aircraft within the range). On the other hand, ADS-C transmits similar information, but only to on ground *Air Traffic Services Units* (ATSUs) and based on an established contract.

We will focus on ADS-B because it provides foundational technology for the Next Generation Air Transportation System (NextGen) and the Single European Sky ATM Research (SESAR) programme (Richards et al., 2010). Thus, it is a fundamental building block of the future ATM systems. One of the main ADS-B advantages is that it provides ATCs with real-time position information obtained from a navigation system that is usually more accurate than that provided by a radar-based system. Increased accuracy means increased safety and increased capacity of the airspace and the airport. Additionally, ADS-B works with low altitudes and even at ground level, which allows its use to monitor aircraft ground manoeuvring, greatly improving the resolution of ASDE *(Airport Surface Detection Equipment)*.

ADS-B has two components: *i) ADS-B Out*, that broadcasts aircraft data, such as the aircraft identification, position, altitude, and velocity; and *ii) ADS-B In*, that receives useful information for the pilot such as the traffic information about surrounding aircraft or information transmitted through the *Flight Information Service-Broadcast* (FIS-B), such as weather reports or flight information (e.g. temporary flight restrictions). Deploying ADS-B technology requires two avionic components, *i)* a navigation system and a datalink within the aircraft, and *ii)* a receiving (typically ground) station to receive the ADS-B data.

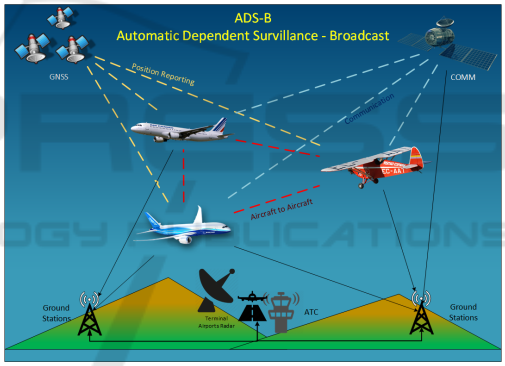

Figure 1: ADS-B operating scheme.

Figure 1 illustrates the ADS-B operating scheme. Aircraft obtain their position through the built-in sensors along with positioning data from the GNSS satellites, and broadcast such data to any receiver within its area of coverage. ATC ground stations receive such data and use it as a replacement for the secondary radar. Nearby aircraft can also receive such data to provide situational awareness and allow self separation. In situations where the aircraft is away from other aircraft or ground stations (such as uncovered areas or transatlantic flights), communication satellites (COMM) are used to communicate with the receivers, thus allowing to have global coverage.

Some research and industrial literature have been published about ADS-B. The earliest papers, such as (Hicok and Lee, 1998) and (Zeitlin and Strain, 2002), describe the system, its components and its implementation, while more recent works focus on how

ADS-B infrastructure is currently being deployed and the open challenges around it (Ali, 2016; Strohmeier et al., 2014). Collaborative projects like OpenSky[1] demonstrate how a network of ADS-B sensors can be deployed to capture and share ADS-B data to the community (Schäfer et al., 2014). A recent paper (Strohmeier et al., 2015) reports the experience of implementing OpenSky and emphasize on its data architecture. The authors report that its original MySQL-based deployment lacked of scalability and it was replaced by a Lambda-oriented architecture (see Section 3). Another paper (Boci and Thistlethwaite, 2015) reports a preliminary experience of designing a data lake for ADS-B data. In contrast to our approach, this deployment is restricted to a single type of surveillance data (CAT033) and does not devise how other data streams can be managed and integrated to obtain more valuable knowledge.

3 AIRPORTS DL

A considerable effort has been carried out during the past decade in Big Data solutions and scalable data systems, and terms like *Hadoop* or *NoSQL* are two of the new buzzwords in computational circles. On the one hand, Hadoop is able to run large-scale batch computations in a parallelized fashion, at the price of high latency time. On the other hand, NoSQL databases are highly scalable solutions, but face some limitations regarding traditional relational databases. However, these technologies excel when they are combined intelligently with other tools (in the Big Data ecosystem) to build scalable and fault tolerant systems which are able to deal with variable and complex amounts of data (Marz and Warren, 2015). These systems are also extensible and allows ad-hoc queries to be performed over the big data repository.

The *Lambda* architecture (Marz and Warren, 2015) is the main reference to build such type of systems. It isolates real-time Big Data management needs into three layers: *(i)* the *Batch layer* is responsible of preserving the master dataset, and computes batch views transforming (raw) data for particular end-user purposes; *(ii)* the *Serving layer* enables batch views to be efficiently accessed, and *(iii)* the *Speed layer* assumes real-time data management.

Our current needs must be satisfied by only implementing *Batch* and *Serving* layers, because real-time data management is not currently addressed. Although different approaches can be adopted, we choose, as previously explained, the *data lake* one. This topic has received much attention recently (Miloslavskaya and Tolstoy, 2016; Madera and Laurent, 2016; Hai et al., 2016). A data lake comprises a set of centralized repositories with no schema-on-write restrictions. That is, structured and unstructured data can be effectively stored and only on-read restrictions are made. Descriptive metadata must be also maintained to avoid the data lake to be turned into a data swamp (Gartner, 2014). The data lake also assumes the traditional *ETL (Extract-Transformation-Load)* responsibilities, while preserving all ongoing data for traceability and analysis purposes. Thus, the data lake implements storage and data computation responsibilities of the *Batch* layer.

Data lakes are usually deployed using *Hadoop*-based technology (White, 2015) to ensure cost-effective storage and processing using the Hadoop Distributed File System (HDFS) and the MapReduce computation model, respectively. Regarding the resulting batch views, which comprises highly-curated data, they must be managed outside of the data lake. The *Serving* layer implementation depends on how data are finally exploited by end-user systems. Although it is common to use data warehouse technology, NoSQL databases are increasingly adopted to deploy scalable Serving layer implementations.

AIRPORTS DL[2] combines these foundations to design an scalable architecture able to deal with voluminous ADS-B data streams, and a variety of flight-related datasets. Figure 2 provides a big picture of AIRPORTS DL, including the data lake itself, external data sources, and the system which implements the Serving layer. All these elements are described by following their numeric identifiers; we also mention the technologies[3] used to implement each one.

1. Data Sources. This element is *"the world around AIRPORTS DL"* and includes all external databases or live services which feed data into the data lake. We collect information from many ADS-B providers to get a wide coverage of the air space. It includes[4] the aforementioned *OpenSky* community network, but also comercial providers. *Flight plans*, *weather information*, provided by the *Global Fore-*

[1] https://opensky-network.org/

[2] AIRPORTS DL relies on the Aviation Data Analytics Platform Testbed (ADAPT) by Boeing Research and Technology-Europe (BR&T-E) in Madrid, Spain

[3] These technologies are usually available within Hadoop distributions, so more information about them can be found in references like (White, 2015).

[4] We also feed ADS-B data captured by the *Frambuesa* BR&T-E sensor that currently operates at the Madrid-Barajas Adolfo Suárez Airport (LEMD).

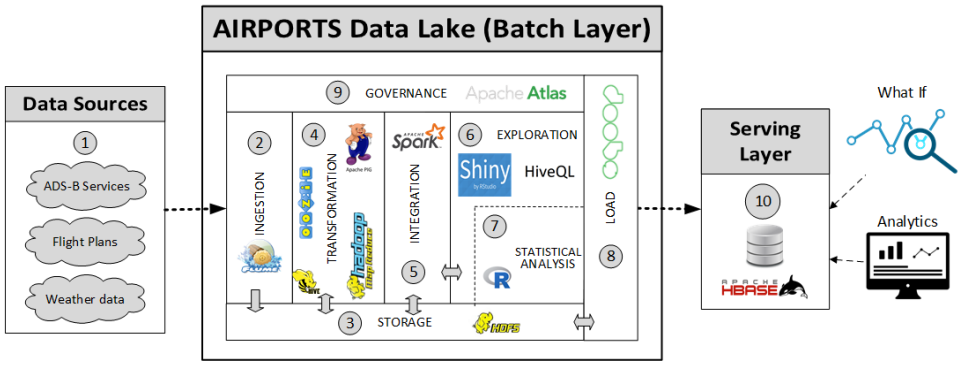

Figure 2: AIRPORTS DL Architecture.

cast System[5], or information about *lightning strikes*, *flights*, *airlines*, and *airports*, are also ingested into the data lake to consolidate a huge data collection.

2. Ingestion. This architectural component is responsible for importing data from the data sources. Many data is directly streamed from the providers, but the component must also get data from available databases or flat files. We use *Apache Flume* to deal with real-time continuous data streams, while the other ingestion methods are implemented using *Apache Sqoop* or Hadoop command utilities. All ingested data is preserved by the storage component.

3. Storage. It is the cornerstone of our architecture and it is designed as an scalable and flexible component which allows any type of (big) data to be stored with no restrictions. This component is implemented on top of the *HDFS* (Shvachko et al., 2010) file system, ensuring reliable and low cost data persistence for large files. Our storage component forces *data immutability* to ensure data traceability. It means that the original raw data will never be modified within the data lake, although it can be copied and progressively refined for different purposes. This component arranges multiple refinement levels where data at different levels of quality will be preserved. Lower levels are used for *transforming* data within the scope of their corresponding data source, while higher levels are used for heterogeneous data *integration*. This component allows data to be accessed using HDFS commands, but also provides a SQL-like interface using external Hive tables.

4. Transformation. It is the "low-level" *data refinery* of AIRPORTS DL and allows data to be manipulated and transformed within their original scope (with no addition of external data). The MapReduce computation model (Dean and Ghemawat, 2008) is the core technology of this component, although other high-level tools such as *Apache Hive* or *Apache Pig*, are also available to implement and run the corresponding jobs. These jobs interact massively with the storage component. In practice, transformation is a complex task which needs many jobs to be effectively orchestrated (we use *Apache Oozie* for this purpose).

5. Integration. It is the "high-level" *data refinery* and enables heterogeneous data to be integrated and aligned to satisfy end-user application requirements. Thus, the transformation and integration components assume the batch view computations (Lambda Batch layer). The integration component also runs complex dataflows implemented using the same technologies.

6. Exploration. It implements an interface to the storage component that enables on-going data to be analyzed before being loaded into the Serving layer. Although it is a dispensable component at architectural level, it is a highly useful tool for data scientists. The Hadoop command line is used for simple data exploration, and more advanced data analysis can be implemented using HiveQL queries.

7. Statistical Analysis. It is a particular exploration subcomponent which provides a high-level interface enabling statistical analysis on top of the storage component. It is currently implemented using RStudio[6], but also makes exhaustive use of *Apache Spark* and visualization libraries like Shiny.

8. Load. This is a simple component which moves batch views from their storage in the data lake to the Serving Layer. Although *Apache Sqoop* is a reference technology at this level, this component is designed

[5]http://www.emc.ncep.noaa.gov/index.php?branch=GFS

[6]https://www.rstudio.com/

to adjust particular load requirements from the Serving layer. As current data loading is implemented using logical Hive tables which play a mediator role between the storage component and our Serving Layer.

9. Governance. It is a transversal component that monitors data manipulations within the data lake and preserves enough information to trace them. Thus, it is in continuous communication with all components. We use *Apache Atlas* to implement data governance and lineage policies. It can be easily integrated with the other tools in the data lake, allowing metadata to be easily managed for each data manipulation.

10. Serving Layer. This component is out of the data lake and there are no restrictions about its implementation. Nevertheless, it must be deployed to ensure ATM analytics to be effectively obtained on top of the batch views loaded from the storage component. We currently choose *Apache HBase* since it provides straightforward integration within the Hadoop ecosystem, and allows batch views to be queried in real-time. This deployment is suitable for the current status of the project, but it could be replaced if new ATM requirements demand it.

The main ADS-B processing dataflow is as follows:

- Independent *Flume* agents are deployed, in the ingestion component, to (continuously) capture ADS-B data from different providers.
- These corresponding message streams are delivered to the storage component, which preserves them in particular repositories.
- A transformation *Oozie* workflow is daily launched for ADS-B data cleansing and fact discovering. It includes simple tasks (e.g. discarding useless messages or aligning field values to satisfy the AIRPORTS data model), but also more complex ones (e.g. determining trajectories when different flights use the same callsign or relevant messages are lost). As a result, messages and trajectories are effectively cleaned in the scope of each ADS-B provider. This data is made available for integration, but also for exploration purposes.
- A second *Oozie* workflow is then launched for integration. It is performed from two complementary perspectives. On the one hand, ADS-B messages are grouped to reduce the existing gaps in flight trajectories. This issue is due to each data provider covers particular airspace areas, but does not capture information outside of them. Thus, more realistic trajectories are reconstructed after grouping. On the other hand, we use other repositories in AIRPORTS DL to enhance such trajectories; e.g. ADS-B does not provide information about departure/arrival airports, but it is inferred by integrating data from our flight/airport repositories. All batch views are preserved in the storage component and then loaded into HBase.

It is just an example, but other dataflows are currently implemented, and preliminary end-user prototypes also interact with the Serving layer to obtain relevant analytics for ATM.

4 CONCLUSIONS

The data lake oriented architecture AIRPORTS DL has been motivated and explained in the current paper. Although it is an on-going deployment, AIRPORTS DL has been successfully used to ingest, store, and process ADS-B messages, and other flight related data, for almost the last two years. During this period, an average amount of 20 GB of new flight data (from the most relevant sources) and 20 GB of weather data have been collected per day.

This data has been refined within the data lake to satisfy particular needs for ATM optimization and further decision making. Although these end-used requirements are part of our future work, initial flight-related pattern analysis and visualization prototypes have been deployed. For instance, Figure 3 shows a trajectory density map which enables the main traffic patterns, in the Spanish air space, to be identified from reconstructed trajectories. Regarding visualization, Figure 4 shows a screenshot of our on-going dashboard. It just allows to visualize different flight trajectory dimensions (GPS positions, speed or altitude profiles, etc.), but future versions will enable all batch views (including refined data and/or computed analytics) to be efficiently accessed.

Our future work also includes the integration of more data sources (e.g. lightning strikes), and an exhaustive benchmarking of all data lake components to ensure their scalability in a production ATM scenario.

ACKNOWLEDGEMENTS

This work is part of a joint project (AIRPORTS) with Boeing Research and Technology Europe (BRT&E), and is funded by CDTI and MINECO (FEDER) ref. IDI-20150616, CIEN 2015. Authors are partly funded by MINECO (PGE and FEDER) projects TIN2013-46238-C4-3-R, TIN2016-78011-

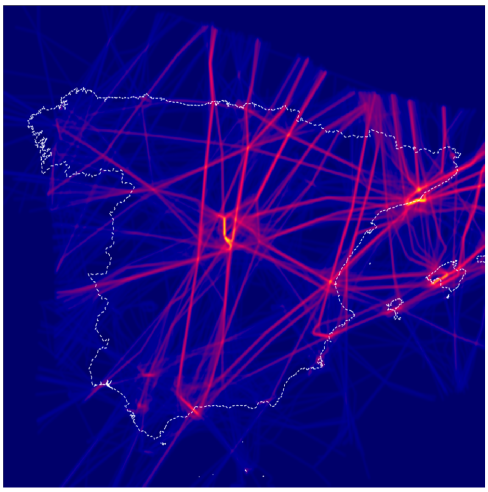

Figure 3: Trajectories density map.

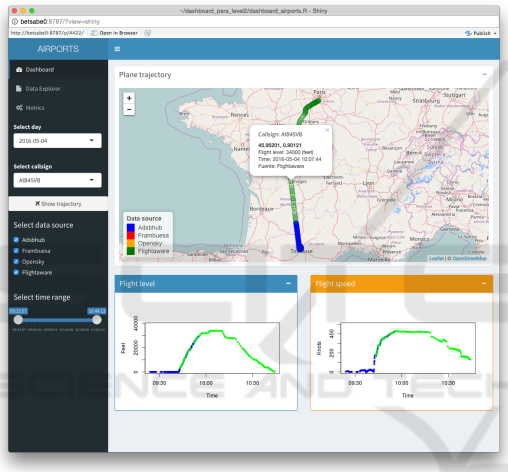

Figure 4: AIRPORTS dashboard.

C4-1-R, MTM2014-56235-C2-1-P and MTM2014-56235-C2-2, and Consejería de Educación, JCyL, project VA212U13.

REFERENCES

Ali, B. (2016). System Specifications for Developing an Automatic Dependent Surveillance-Broadcast (ADS-B) Monitoring System. *International Journal of Critical Infrastructure Protection*, 15:40–46.

Blythe, W., Anderson, H., and King, N. (2011). ADS-B Implementation and Operations Guidance Document. *International Civil Aviation Organization. Asia and Pacific Ocean*.

Boci, E. and Thistlethwaite, S. (2015). A Novel Big Data Architecture in Support of ADS-B Data Analytic. In *Integrated Communication, Navigation and Surveillance Conference*, pages C1–1–C1–8.

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communications of ACM*, 51(1):107–113.

Eurocontrol (2017). Flight Movements and Service Units 2017-2023. Technical report, Eurocontrol. *http://www.eurocontrol.int/sites/default/files/content/documents/official-documents/forecasts/seven-year-flights-service-units-forecast-2017-2023-Feb2017.pdf*.

European Commission (2011). Commission Implementing Regulation (EU) No 1207/2011. Official Journal of the European Union. *http://data.europa.eu/eli/reg_impl/2011/1207/oj*.

Gartner (2014). Gartner Says Beware of the Data Lake Fallacy. *http://www.gartner.com/newsroom/id/2809117*.

Hai, R., Geisler, S., and Quix, C. (2016). Constance: an intelligent Data Lake System. In *International Conference on Management of Data*, pages 2097–2100.

Hicok, D. and Lee, D. (1998). Application of ADS-B for Airport Surface Surveillance. In *17th Digital Avionics Systems Conference*, volume 2, pages F34/1–F34/8.

Madera, C. and Laurent, A. (2016). The Next Information Architecture Evolution: the Data Lake Wave. In *8th International Conference on Management of Digital EcoSystems*, pages 174–180.

Marz, N. and Warren, J. (2015). *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co.

Miloslavskaya, N. and Tolstoy, A. (2016). Application of Big Data, Fast Data, and Data Lake Concepts to Information Security Issues. In *4th International Conference on Future Internet of Things and Cloud*, pages 148–153.

Richards, W., O'Brien, K., and Miller, D. (2010). New Air Traffic Surveillance Technology. *Aero*, 2:7–14.

Schäfer, M., Strohmeier, M., Lenders, V., Martinovic, I., and Wilhelm, M. (2014). Bringing up OpenSky: a Large-Scale ADS-B Sensor Network for Research. In *13th International Symposium on Information Processing in Sensor Networks*, pages 83–94.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The Hadoop Distributed File System. In *26th Symposium on Mass Storage Systems and Technologies*, pages 1–10.

Strohmeier, M., Martinovic, I., Fuchs, M., Schäfer, M., and Lenders, V. (2015). OpenSky: A Swiss Army Knife for Air Traffic Security Research. In *34th Digital Avionics Systems Conference*, pages 4A1–1–4A1–14.

Strohmeier, M., Schäfer, M., Lenders, V., and Martinovic, I. (2014). Realities and Challenges of NextGen Air Traffic Management: the case of ADS-B. *IEEE Communications Magazine*, 52(5):111–118.

Terrizzano, I., Schwarz, P., Roth, M., and Colino, J. (2015). Data Wrangling: The Challenging Journey from the Wild to the Lake. In *7th Biennial Conference on Innovative Data Systems Research*. Paper 2.

White, T. (2015). *Hadoop: The Definitive Guide*. O'Reilly.

Zeitlin, A. and Strain, R. (2002). Augmenting ADS-B with Traffic Information Service-Broadcast. In *21st Digital Avionics Systems Conference*, volume 1, pages 3D2–1–3D2–7.