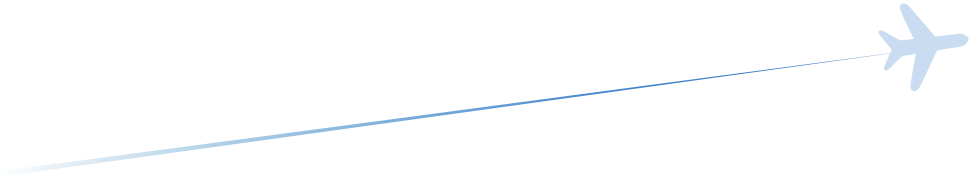

Stream-based data model

D4.2	PU
AURORA	
Grant:	699340
Call:	H2020-SESAR-2015-1
Topic:	Sesar-11-2015 ATM Performance
Consortium coordinator:	CRIDA A.I.E.
Edition date:	11 August 2017
Edition:	00.01.00

Founding Members
EUROPEAN UNION EUROCONTROL
SESAR JOINT UNDERTAKING

Authoring & Approval

Authors of the document

Name/Beneficiary	Position/Title	Date
Brian Mac Namee / CeADAR	Task Leader	09/08/2017
Shen Wang / CeADAR	Project Member	09/08/2017
Aditya Grover / CeADAR	Project Member	09/08/2017

Reviewers internal to the project

Name/Beneficiary	Position/Title	Date
Pablo Sánchez-Escalonilla / CRIDA	Project Coordinator	10/08/2017
Fernando Celorrio/CRIDA	Project Member	10/08/2017

Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
Pablo Sánchez-Escalonilla / CRIDA	Project Coordinator	11/08/2017
Javier López-Leones / BR&T	Project Member	11/08/2017
Brian Mac Namee / CeADAR	Project Member	11/08/2017
Philip Phantholt / FR24	Project Member	11/08/2017

Rejected By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date

Document History

Edition	Date	Status	Author	Justification
00.00.01	09/08/2017	Draft	B. Mac Namee S. Wang A. Grover	Initial version
00.00.02	10/08/2017	Draft	F. Celorrio	Refined version
00.00.03	10/08/2017	Draft	S. Wang A. Grover	Refined version
00.00.04	11/08/2017	Draft	P. Sánchez-Escalonilla	Refined version
00.01.00	11/08/2017	Final draft	Several Authors	Final version for submission

AURORA

ADVANCED USER-CENTRIC EFFICIENCY METRICS FOR AIR TRAFFIC PERFORMANCE ANALYTICS

This report is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 699340 under European Union's Horizon 2020 research and innovation programme.

Abstract

This document addresses the availability of stream-based data model prototype in support of the online calculation of user centric air traffic efficiency indicators.

Table of Contents

Executive summary *5*

1 Introduction *6*

1.1 Purpose of the document 6

1.2 Intended readership 6

1.3 Glossary of terms 6

1.4 Acronyms and Terminology 6

2 Availability Note Form *8*

3 References *14*

List of Tables

Table 1 - Glossary of terms 6

Table 2 - Acronyms and terminology 7

Table 3 - Throughput statistics in number of messages per 10 minutes 11

Table 4 - Latency statistics in seconds 12

List of Figures

Figure 1 - Stream-based Data Model 9

Figure 2 - Throughput of stream producer 11

Figure 3 - Latency of stream-based data model 11

Executive summary

This document is the availability note of the stream-based data model to calculate user-centric air traffic efficiency indicators on-line. This prototype has been developed and successfully tested by CeADAR at its premises and is ready to be integrated in the ADAPT platform at BRTE premises for its further evaluation.

The main functionality provided by the prototype, following the technical specification described in AURORA deliverable D4.1 [1], is to calculate user centric air traffic efficiency indicators over a flight's duration by leveraging state-of-the-art big data technologies and distributed system design. The testing of this prototype is based on the evaluation plan described in D3.1 [2]. The testing results highlight that the stream-based data model can successfully process an input surveillance data stream calculating five key efficiency indicators for each flight every 30 seconds, with an average latency of 27 seconds per message. Testing is based on the dataset collected for AURORA of all European flights on February 20th 2017.

In the current version testing is performed against pre-calculated reconstructed and generated trajectories so calculation of these is simulated. Online services to perform trajectory reconstruction and generation are being created and will be integrated with this prototype.

1 Introduction

1.1 Purpose of the document

The Availability Note implicitly confirms that the stream-based data model prototype was developed in conformance with the appropriate technical requirements [1], and was tested in conformance with the Verification Plan fulfilling the defined Acceptance Criteria [2].

The deviations from the planned technical specification and evaluation plan are:

- Online services for trajectory reconstruction and generation are not yet available so these operations are simulated in the current implementation.
- As online trajectory reconstruction and generation are not yet included, then evaluation of calculated online indicators against offline methods is not included as calculated indicators will, by design, be exactly the same as offline calculations.

1.2 Intended readership

This document is intended to be used by AURORA's members and by the SJU official reviewers.

1.3 Glossary of terms

Term	Description
FEA-DW	It quantifies the extra-fuel consumption of the actual trajectory in comparison with the geodesic trajectory (i.e. GEO_FP).
FEA-FW	It quantifies the extra-fuel consumption of the actual trajectory in comparison with the optimal fuel-based trajectory (i.e. FREE_C0).
CEA-CW1	It quantifies the extra-costs of the actual trajectory in comparison with the optimal cost-based trajectory (time and fuel) (i.e. FREE_CI).
CEA-CW2	It quantifies the extra-costs of the actual trajectory in comparison with the optimal cost-based trajectory following the flight plan (i.e. OPT_CI).

Table 1 - Glossary of terms

1.4 Acronyms and Terminology

Term	Definition
ADAPT	Aviation Data Analytics Platform Testbed
API	Application Program Interface
ATC	Air Traffic Control
ATM	Air Traffic Management
AU	Airspace User
BADA	Base of Aircraft Data

Term	Definition
BRTE	Boeing Research and Technology Europe
CeADAR	Centre for Applied Data Analytics
CI	Cost Index
CPU	Central Processing Unit
CRIDA	Centro de Referencia I+D+i (ATM)
CSCI	Computer Software Configuration Items
CSV	Comma-Separated Values
E-ATMS	European Air Traffic Management System
ER	Exploratory Research
FP	Flight Plan
FR24	FlightRadar24
IBP	Industrial Based Platform
IRS	Interface Requirements Specification
Item	"Item" is here used as a generic name for prototype(s), platform(s), platform component(s), etc.
KEA	Key Performance Environment Indicator based on Actual Trajectory
LTS	Long Term Support
MFA	Multilateral Framework Agreements
NM	Network Manager
SESAR	Single European Sky ATM Research Programme
SESAR Programme	The programme which defines the Research and Development activities and Projects for the SJU.
SJU	SESAR Joint Undertaking (Agency of the European Commission)
SJU Work Programme	The programme which addresses all activities of the SESAR Joint Undertaking Agency.
SPR	Safety and Performance Requirements
TS	Technical Specification
UCD	University College Dublin
VV	Verification and Validation
XML	Extensible Markup Language

Table 2 - Acronyms and terminology

2 Availability Note Form

Availability Note by MEMBER(S):	CeADAR
Availability Note Date:	09/08/2017
Person Responsible Item's Availability:	Brian Mac Namee (CeADAR)
Deliverable Code:	D4.2
Item's Name and Version:	AURORA Stream-based Data Model edition v.00.01.00
Details of Availability	
Step and Maturity Level addressed by the Item implementation:	V01 according to E-OCVM
Documents and their versions used for the Item's development:	• 699340 - D2.1 - Definition of User-Centric Air Traffic Efficiency Indicators - v.00.01.01 - Feb, 2017 • 699340 - D4.1 - Data Streaming Requirements - v.00.01.00 - Mar, 2017
Item's Verification related documentation:	699340 - D3.1 - Description of experimental plans - v.00.01.00 - Feb, 2017
Date and location of the Item's Verification:	CeADAR's Premises, August 2017
Item's Verification witnessed by:	• Brian MAC NAMEE - brian.macnamee@ucd.ie • Shen WANG - shen.wang@ucd.ie • Aditya GROVER - aditya.grover@ucd.ie
Verification Overall Result:	The stream-based data model is successfully tested using 24 hours data on Feb 20, 2017, with approximately 15,000 flights in European area, for calculating 5 flight efficiency indicators that are updated every 30 seconds, with all message processing latencies of less than 52 seconds.
Information on the Content:	

PROTOTYPE ARCHITECTURE

The main technologies used for the implementation of the data pipeline are Apache Spark [5] and Apache Kafka [6] using JAVA 8. Figure 1 shows an overview of the architecture of the system implemented. There are five main components in this stream-based data model prototype:

- **Stream Producer**: Writes reconstructed trajectory messages to the data stream every 5 seconds to the Input Stream Buffer. Implemented in Java 1.8 with Producer API of Apache Kafka 0.10.2.0 that can guarantee reliable streaming communications (i.e. no data missing and out-of-sequence issues).
- **Stream Buffer**: A dynamic distributed data storage that adapts the incoming and outgoing rate of messages among stream producers/consumers/processors and data stores. Implemented

using Apache Kafka (version 0.11.0.0).

- **Stream Processor**: Reads reconstructed trajectory data every 1 minute from the Input Stream Buffer; processes this data by calculating flight efficiency indicators; and writes these indicators back to the Output Stream Buffer. Implemented in Java 1.8 using Streaming API of Apache Spark 2.2.0.
- **Stream Consumer**: Reads indicators from the Output Stream Buffer and writes these to a persistent Data Store. Implemented in Java 1.8 with Producer API of Apache Kafka 0.10.2.0 that can guarantee reliable streaming communications (i.e. no data missing and out-of-sequence issues)
- **Generated Trajectory Data Store**: Stores and persists the input reference data (i.e. generated trajectories). That is loaded by the stream processor for calculation. Implemented as CSV files using Linux (Ubuntu 14.04LTS) file system.
- **Efficiency Indicator Data Store**: Stores and persists the final calculated efficiency indicators of the stream-based data model. This allows airspace users or network managers to perform complex queries on the efficiency indicators' results. It has been implemented in CSV files using Linux (Ubuntu 14.04LTS) file system.

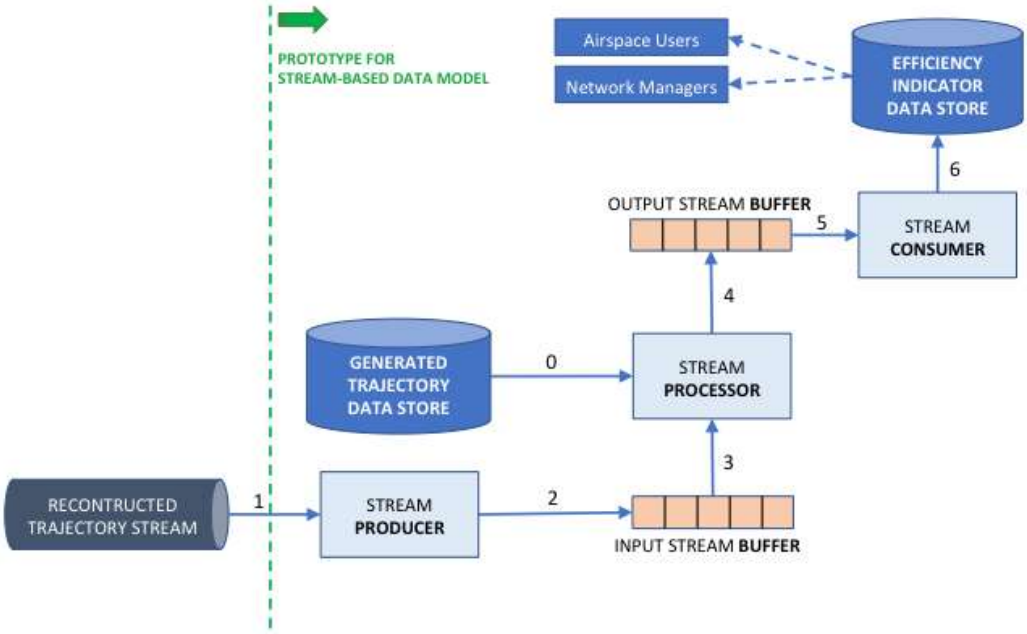

Figure 1 - Stream-based Data Model

DATA FLOW

Figure 1 also indicates the sequence order of the whole data flow in the stream-based data model. The numbers labelling the arrows in the figure indicate the order in which the data flows through the pipeline.

As one part of the external input, data such as operational context, flight plan, BADA, and weather forecast, combined with BRTE's trajectory generation algorithm, can generate the required optimal trajectories for each flight as a reference to calculate flight efficiency indicators. These generated trajectories are made available through a persistent data store, in this case CSV files. Another part of the external input is the surveillance data which contains status information of aircraft updated every 5 seconds. These data can be reconstructed with the data of aircraft mass inferred by BRTE's trajectory reconstruction algorithm. Such reconstructed trajectory data files are filtered in CSV files, and simulated

as a real-time data stream used by the stream producer.

Firstly, the generated trajectories are pre-loaded in the stream processor (Flow 0). Then, the reconstructed trajectory stream sends simulated surveillance messages one point at a time at every viable time interval (i.e. according to the time gap between each consecutive data points), to the stream producer (Flow 1). The stream producer forwards messages to the input stream buffer (Flow 2). Next, the stream processor reads reconstructed trajectory data at a fixed and larger time interval (30 seconds) in micro-batch manner from the input stream buffer (Flow 3), performs efficient distributed calculation using a series of map and reduce (i.e. MapReduce [4] is a de-facto standard programming model for processing large-scale data efficiently in distributed systems) transformations, and then pushes the results of flight efficiency indicators to the output stream buffer (Flow 4). The stream consumer can be subscribed to, at the request of the users (Flow 5). Consumer application can persist data from the output stream buffer into a file system data store (Flow 6), which can be used by airspace users and network managers to query on.

Remarks, Notes, Guidelines

VERIFICATION RELATED INFORMATION

The following describes the verification tests performed on the system:

- **Testing Scenario**: Roughly 15,000 flights depart and arrive in European area over a full day on Feb 20th, 2017. All generated and reconstructed trajectories used in this test are filtered from raw XML trajectory files on BRTE's ADAPT platform.
- **Testing Environment**: A server machine in UCD's premise. It consists of 24 CPUs (i.e. each one is Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz), 64GB RAM, 896GB disk storage, and has equipped Ubuntu 14.04LTS operating system.
- **Testing Methodology**: Five flight efficiency indicators (i.e. KEA, FEA-DW, FEA-FW, CEA-CW1, CEA-CW2) are calculated in this testing. The real-time indicator results are calculated as the ratio of the cost value of the latest updated reconstructed trajectory point, to the cost value of its corresponding nearest reference trajectory point. Note that the value of CEA-CW1/2 are computed regardless of route charges. The batch interval of Spark Streaming is set as 30 seconds, and the stream processor is running on 4 CPUs with 8GB of driver's memory and 4GB of executors' memory.
- **Testing Results**: The key metrics for online data processing systems are throughput and latency. The evaluation of the system implemented according to these metrics is as follows:
  - **Throughput**: Figure 2 shows the throughput achieved by the system. Summary statistics of throughput measures are shown in Table 3. The mean throughput is 131,780 surveillance messages processed every 10 minutes.

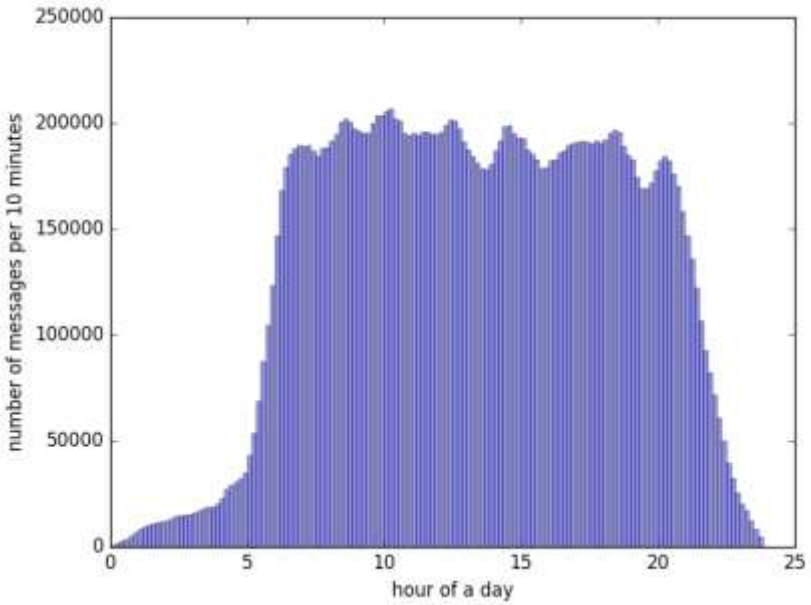

Figure 2 - Throughput of stream producer

mean	std	min	25%	median	75%	max
131,780	78,132.92	92	32,177.5	182,306.5	192,371.75	206,759

Table 3 - Throughput statistics in number of messages per 10 minutes

- **Latency**: For each data message, latency is measured as the time duration from stream producer to the output stream buffer. The distribution of all messages (i.e. in total 18,968,070 messages) latency are shown in, while the main statistics are shown in Table 4. The average latency per message is 27.43 seconds.

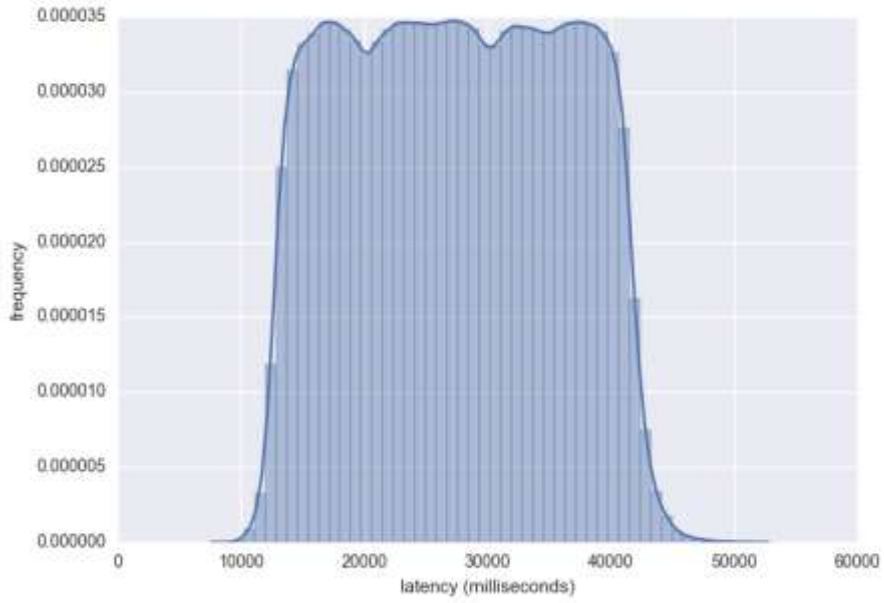

Figure 3 - Latency of stream-based data model

mean	std	min	25%	median	75%	max
27.43	8.54	8.50	20.10	27.40	34.76	51.90

Table 4 - Latency statistics in seconds

INTEGRATION RELATED REMARKS

The steps required to run and evaluate the prototype are listed below:

- **Step 0 Preparations on Data Store**: This is the pre-processing step where the reconstructed and generated trajectory XML files are filtered and converted into CSV files. There are three python 2.7 scripts that need to be run for pre-processing:
  1. **xml_2_csv.py**. Example: *python xml_2_csv.py --input /traj/ --d 2017-02-20 --output /filtered_traj/* This script first reads and filters the XML files from the BRTE ADAPT platform (placed in directory /traj/) to include only those data fields required for the calculation of efficiency indicators. It then filters it into CSV files (placed in directory /filtered_traj/) by converting timestamp into epoch time (according to provided date 2017-02-20) and calculating aggregated travelled great circle distance using Haversine formula.
  2. **add_cost.py**. Example: *python add_cost.py --input /traj/ --output /traj_cost/* This script reads each data row within each CSV file obtained from the script in step 1 (placed in directory /traj/), by adding cost values, for reference trajectories FREE_CI and OPT_CI (placed in directory /traj_cost/).
  3. **hour_script.py**. This script is to be run on the reconstructed CSV files obtained from running the first and second script. After running this script, the reconstructed trajectory data gets organized into 24 CSV files, 1 for each hour, sorted by timestamp. This replicates the surveillance data stream. To run, type *"python hour_script.py"* .
- **Step 1 Start Stream Buffer Service**:
  1. First start Zookeeper. For example, from the home directory of Kafka run:

     ./bin/zookeeper-server-start.sh config/zookeeper.properties

     where zookeeper.properties is the configuration file for zookeeper server.
  2. Next start the Kafka server. For example, from the home directory of Kafka run:

     ./bin/ kafka-server-start.sh config/server.properties

     where server.properties is the configuration file to which one needs to add "log.message.timestamp.type=LogAppendTime" for tracking latency metrics.
- **Step 2 Start Stream Processor**: From the apache Spark home directory run:

  bin/spark-submit --class "processor.LightStreamProcessorStr" --conf spark.driver.memory=8G --conf spark.executor.memory=4G sparkprocessor.jar

  Related configurations (e.g. name of kafka topics, address and port number of kafka server, reference trajectory directory and cost index directory, batch interval for spark, and submission URL for spark) should be set in *sparkprocessor.properties* before run the above command.
- **Step 3 Start Stream Producer**: Should start after all required data (generated trajectories and cost index files) are successfully preloaded on stream processor using the command:

  java -jar reconstream.jar

  Then it starts to write reconstructed trajectory at the pace according to the time gap between each consecutive data message. When it finishes reading and sending all reconstructed trajectory data, it writes the throughput statistics in the pre-configured file (e.g. througput.csv). Related configurations (e.g. name of input kafka topic, address and port number of kafka server,

reconstructed trajectory directory, acceleration for simulation, and the file name of throughput statistics) should be set in *reconstream.properties* before run the above command.

- **Step 4 Start Stream Consumer**: Started using the command:

  java -jar persist.jar

  This keeps reading the results of calculated indicators. When user wants to stop, one should press "ctrl+c" (i.e. sending signal "SIGINT"), then it stops and write all data read so far and latency statistics into separate files for further analysis.

  Related configurations (e.g. name of output kafka topic, address and port number of kafka server, and the file name of latency statistics and efficiency indicators results) should be set in *persist.properties* before run the above command.

ADDITIONAL NOTES, ETC

Because of resource and complexity issue, connectors (which move data between database and stream buffer) as well as on-line trajectory reconstruction and generation are neither implemented nor tested.

On-line trajectory reconstruction and generation are not implemented. Therefore, the accuracy of the on-line flight efficiency indicators results is not tested.

3 References

[1] AURORA D4.1, “Data Streaming Requirements”, Edition 00.01.00, Mar 2017.

[2] AURORA D3.1, “Description of Experimental Plans”, Edition 00.01.00, Feb 2017.

[3] AURORA D2.1, “Definition of User-Centric Air Traffic Efficiency Indicators”, Edition 00.01.01, Feb 2017.

[4] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Commun. ACM, 51(1):107–113, 2008.

[5] Apache Spark http://spark.apache.org/

[6] Apache Kafka https://kafka.apache.org/

-END OF DOCUMENT-