The Lab Series

Tracking commercial aircraft in near real-time using a Raspberry Pi, Kafka and Vertica

Mark Whalley
Vertica Systems Engineer
12th October 2017

www.vertica.com
www.myvertica.com
Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
Agenda

▪ **Background to “The Lab Series” and the Big Data & Machine Learning Meetups**

▪ Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization

▪ What’s next?
Meetup #7 PLEASE NOTE: Limit of 140 attendees (see below)
Welcome to Meetup #7, and what we hope will be another interesting evening of presentations and lightning talks...
And then there were three: London, Munich & Cambridge
The Lab Series - Background

▪ Mini projects
  - Incubate
  - Subject
  - Technology
  - Direction
  - Presentations
▪ Inclusive to all
Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
The Lab Series – Project #1

Automatic Dependent Surveillance - Broadcast (ADS-B)
Contrary to popular belief...
This got me thinking...

https://vimeo.com/110348926
RAdio Detection and Ranging (RADAR)
Automatic Dependent Surveillance – Broadcast (ADS-B)
FlightRadar24

https://www.flightradar24.com/
Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - **Using a Raspberry Pi to capture and decode ADS-B signals**
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
Raspberry PI and USB RTL-SDR

DUMP1090
The “Portable” Raspberry PI
Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
DUMP1090

- **DUMP1090 is a Mode S decoder** specifically designed for RTLSDR devices.
- Some of the main features include:
  - Robust decoding of weak messages.
  - Network support: **TCP 30003 stream** (MSG5...), Raw packets, HTTP.
  - Embedded HTTP server that **displays the currently detected aircrafts on Google Map**.
  - Single bit errors correction using the 24 bit CRC.
  - Ability to decode DF11, DF17 messages.
  - Ability to decode DF formats like DF0, DF4, DF5, DF16, DF20 and DF21 where the checksum is xored with the ICAO address by brute forcing the checksum field using recently seen ICAO addresses.
Live (DUMP1090) tracking
Analysing DUMP1090 data
## Analysing DUMP1090 data

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG,1</td>
<td>ES Identification and Category</td>
<td>DF17 BDS 0,8</td>
</tr>
<tr>
<td>MSG,2</td>
<td>ES Surface Position Message</td>
<td>DF17 BDS 0,6, Triggered by nose gear squat switch.</td>
</tr>
<tr>
<td>MSG,3</td>
<td>ES Airborne Position Message</td>
<td>DF17 BDS 0,5, Triggered by ground radar.</td>
</tr>
<tr>
<td>MSG,4</td>
<td>ES Airborne Velocity Message</td>
<td>DF17 BDS 0,9, Triggered by ground radar. Not CRC secured.</td>
</tr>
<tr>
<td>MSG,5</td>
<td>Surveillance Alt Message</td>
<td>DF4, DF20, MSG,5 will only be output if the aircraft has previously sent a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSG,1, 2, 3, 4 or 8 signal.</td>
</tr>
<tr>
<td>MSG,6</td>
<td>Surveillance ID Message</td>
<td>DF5, DF21, MSG,6 will only be output if the aircraft has previously sent a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSG,1, 2, 3, 4 or 8 signal.</td>
</tr>
<tr>
<td>MSG,7</td>
<td>Air To Air Message</td>
<td>DF16, Triggered from TCAS. MSG,7 is now included in the SBS socket output.</td>
</tr>
<tr>
<td>MSG,8</td>
<td>All Call Reply</td>
<td>DF11, Broadcast but also triggered by ground radar.</td>
</tr>
</tbody>
</table>
## Analysing DUMP1090 data

<table>
<thead>
<tr>
<th>Field 1:</th>
<th>Message type</th>
<th>(MSG, STA, ID, AIR, SEL or CLK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field 2:</td>
<td>Transmission Type</td>
<td>MSG sub types 1 to 8. Not used by other message types.</td>
</tr>
<tr>
<td>Field 3:</td>
<td>Session ID</td>
<td>Database Session record number</td>
</tr>
<tr>
<td>Field 4:</td>
<td>AircraftID</td>
<td>Database Aircraft record number</td>
</tr>
<tr>
<td>Field 5:</td>
<td>HexIdent</td>
<td>Aircraft Mode S hexadecimal code</td>
</tr>
<tr>
<td>Field 6:</td>
<td>FlightID</td>
<td>Database Flight record number</td>
</tr>
<tr>
<td>Field 7:</td>
<td>Date message generated</td>
<td>As it says</td>
</tr>
<tr>
<td>Field 8:</td>
<td>Time message generated</td>
<td>As it says</td>
</tr>
<tr>
<td>Field 9:</td>
<td>Date message logged</td>
<td>As it says</td>
</tr>
<tr>
<td>Field 10:</td>
<td>Time message logged</td>
<td>As it says</td>
</tr>
</tbody>
</table>
## Analysing DUMP1090 data

| Field 11: Callsign       | An eight digit flight ID - can be flight number or registration (or even nothing). |
| Field 12: Altitude       | Mode C altitude. Height relative to 1013.2mb (Flight Level). Not height AMSL. |
| Field 13: Ground Speed   | Speed over ground (not indicated airspeed). |
| Field 14: Track          | Track of aircraft (not heading). Derived from the velocity E/W and velocity N/S. |
| Field 15: Latitude       | North and East positive. South and West negative. |
| Field 16: Longitude      | North and East positive. South and West negative. |
| Field 17: Vertical Rate  | 64ft resolution |
| Field 18: Squawk         | Assigned Mode A squawk code. |
| Field 19: Alert (Squawk change) | Flag to indicate squawk has changed. |
| Field 20: Emergency      | Flag to indicate emergency code has been set. |
| Field 21: SPI (Ident)    | Flag to indicate transponder Ident has been activated. |
| Field 22: IsOnGround     | Flag to indicate ground squat switch is active. |
### Analysing DUMP1090 data

<table>
<thead>
<tr>
<th>MSG 1</th>
<th>MSG 2</th>
<th>MSG 3</th>
<th>MSG 4</th>
<th>MSG 5</th>
<th>MSG 6</th>
<th>MSG 7</th>
<th>MSG 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>TT</td>
<td>SID</td>
<td>AID</td>
<td>Hex</td>
<td>FID</td>
<td>DMG</td>
<td>TMG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Data Columns

- **11**: CS
- **12**: Alt
- **13**: GS
- **14**: Trk
- **15**: Lat
- **16**: Lng
- **17**: Alt
- **18**: GS
- **19**: Trk
- **20**: Lat
- **21**: LNG
- **22**: Alt

### Data Values

- **Gnd**: Ground
- **Alt**: Altitude
- **GS**: Ground Speed
- **Trk**: Track
- **Lat**: Latitude
- **Lng**: Longitude
- **Emer**: Emergency
- **SPI**: Special Purpose Information
- **VR**: Vertical Reference
- **Sqr**: Square
- **Alrt**: Alert

Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
    - Introduction to Vertica
    - Kafka / Vertica integration and Management Console
    - Vertica integration tools and simple visualisations
    - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
    - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
Defining KAFKA Topics
Defining KAFKA Topics

Topics:
One per message type (MSG_1, MSG_2 etc)
Single broker
Single partition (per topics)
Feeding DUMP1090 data into KAFKA topics
Kafka Topics – Receiving messages
Agenda

▪ Background to “The Lab Series” and the Big Data & Machine Learning Meetups
▪ Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
    - **Introduction to Vertica**
      - Kafka / Vertica integration and Management Console
      - Vertica integration tools and simple visualisations
      - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
      - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
▪ What’s next?
How the World’s Leading Data-Driven Businesses Came to Rely on Vertica
Our Customers:
The World’s Most Data-Driven Enterprises

“Digital Darwinism is unkind to those who wait.”
- Ray Wang, Constellation Research, June 2015
C-Store: A Column-oriented DBMS

Mike Stonebraker, Daniel J. Abadi†, Adam Batkin, Xuedong Chen‡, Mitch Cheeck†, Miguel Ferrante, Edmund Lau, Amrison Lin†, Sam Madden, Elizabeth O’Neil†, Pat O’Neil†, Alex Russinovich‡, Nga Tran†, Stan Zdonik‡

MIT CSAIL
Cambridge, MA

Stanford University
Stanford, CA

Abstract

This paper presents the design of a read-optimized relational DBMS that contrasts sharply with most current systems, which are write-optimized. Among the many differences in its design are: storage of data by column rather than by row, careful coding and packing of objects into storage including main memory during query processing, storing an overlapping collection of column-oriented projections, rather than the current use of tables and indices, a non-traditional implementation of transactions which includes high availability and snapshot isolation for read-only transactions, and the extensive use of bitmap indexes to complement B-tree structures.

We predict preliminary performance data on a subset of TPC-H and show that the system we are building, C-Store, is substantially faster than popular commercial products. Hence, the architecture looks very encouraging.

1. Introduction

Most major DBMS vendors implement row-oriented storage systems, where the attributes of a record (or tuple) are placed contiguously in storage. With this row store architecture, a single disk write suffices to push all of the fields of a single record out to disk. Hence, high performance writers are achieved, and we call a DBMS with a row store architecture a write-optimized system. These are especially effective on OLTP-style applications.

In contrast, systems oriented toward ad-hoc querying in which periodically a bulk load of new data is performed, followed by a relatively long period of ad-hoc queries. Other read-ready applications include customer relationship management (CRM) systems, electronic library card catalogs, and other ad-hoc inquiry systems. In such environments, a column store architecture, in which the values for each single column (or attribute) are stored contiguously, should be more efficient. This efficiency has been demonstrated in the warehouse marketplace by products like Sybase IQ [FRED93], SYBASE, Adaptec [ADDA94], and KBI (KBI94). In this paper, we discuss the design of a column store called C-Store that includes a number of novel features relative to existing systems.

With a column store architecture, a DBMS need only read the values of columns required for processing a given query, and can avoid bringing into memory irrelevant attributes. In warehouse environments where typical queries involve aggregates performed over large numbers of data items, a column store has a superior performance advantage. However, there are several other major distinctions that can be drawn between an architecture that is read-optimized and one that is write-optimized.

Current relational DBMSs were designed to pad attributes to byte or word boundaries and to store values in their native data format. It was thought that it was too expensive to shift data values onto byte or word boundaries in main memory for processing. However, CPUs are getting faster at a much greater rate than disk bandwidth is increasing. Hence, it makes sense to trade CPU cycles, which are abundant, for disk bandwidth, which is not. This tradeoff appears especially profitable in a read-only environment.
**Columnar Storage**
Speeds query time by reading only necessary data.

**Compression**
Lowers costly I/O to boost overall performance.

**MPP Scale-out**
Provides high scalability on clusters with no name node or other single point of failure.

**Distributed Query**
Any node can initiate the queries and use other nodes for work. No single point of failure.

**Projections**
Combine high availability with special optimizations for query performance.
An Open Architecture Integrated with Rich Ecosystem

Data Transformation
- Spark
- kafka
- ETL

User-Defined Functions
- R
- Java
- C++
- Python
- SQL

- Geospatial
- Real-Time
- Text Analytics
- Pattern Matching
- Machine Learning
- Regression

User Defined Storage

Security

External tables to analyze in place

BI & Visualization
- looker
- Qlik
- Tableau
- Logi Analytics

ODBC
JDBC
OLEDB

Microsoft Azure
Amazon Web Services
Google cloud Platform
openstack
Hadoop

Parquet
S3
Hadoop
Orc
Machine Learning at Enterprise Scale Is the Future of All Predictive Analytics
Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - **Kafka / Vertica integration and Management Console**
    - Vertica integration tools and simple visualisations
    - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
    - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
Kafka Integration with Vertica

- Vertica schedules loads to continuously **consume** from Kafka
- JSON, Avro data formats
- CLI for easy setup
- In-database monitoring
## Monitoring

### Vertica Management Console

**Continuous** | **Instance**
---|---

**Show MC data collector monitoring streams**

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Microbatch</th>
<th>Source</th>
<th>Target Schema</th>
<th>Target Table</th>
<th>Kafka Cluster</th>
<th>Timestamp Batch Start</th>
<th>Messages Last Batch</th>
<th>Messages Last Hour</th>
<th>Rows Accepted Last Hour</th>
<th>Rows Rejected Last Hour</th>
<th>Errors Last Hour</th>
<th>End Reason</th>
<th>Microbatch Status</th>
<th>Microbatch Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:23...</td>
<td>20</td>
<td>2455</td>
<td>2055</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:22...</td>
<td>0</td>
<td>1300</td>
<td>1300</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:23...</td>
<td>280</td>
<td>92355</td>
<td>92355</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:22...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:23...</td>
<td>1230</td>
<td>450250</td>
<td>450250</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:24...</td>
<td>1335</td>
<td>537620</td>
<td>537620</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:24...</td>
<td>870</td>
<td>488330</td>
<td>488330</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:24...</td>
<td>2700</td>
<td>1049905</td>
<td>1049905</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:25...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:24...</td>
<td>1600</td>
<td>568230</td>
<td>568230</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
<tr>
<td>dump109...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>dump1090...</td>
<td>penardpel...</td>
<td>May 21, 2017 17:30:23...</td>
<td>0</td>
<td>4470</td>
<td>4470</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
<td>Active</td>
<td>select-</td>
</tr>
</tbody>
</table>
## Monitoring

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Microbatch</th>
<th>Source</th>
<th>Target Schema</th>
<th>Target Table</th>
<th>Kafka Cluster</th>
<th>Timestamp Batch Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_air</td>
<td>dump1090_air</td>
<td>dump1090_kafka</td>
<td>dump1090_air</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:22 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_id</td>
<td>dump1090_id</td>
<td>dump1090_kafka</td>
<td>dump1090_id</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:22 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_msg_1</td>
<td>dump1090_msg_1</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_1</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:23 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_msg_2</td>
<td>dump1090_msg_2</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_2</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:23 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_msg_3</td>
<td>dump1090_msg_3</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_3</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:24 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_msg_4</td>
<td>dump1090_msg_4</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_4</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:24 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_msg_5</td>
<td>dump1090_msg_5</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_5</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:23 AM</td>
</tr>
<tr>
<td>dump1090Scheduler</td>
<td>dump1090_msg_6</td>
<td>dump1090_msg_6</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_6</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 7:32:24 AM</td>
</tr>
</tbody>
</table>
## Monitoring

- **Management Console**

<table>
<thead>
<tr>
<th>Messages Last Batch</th>
<th>Messages Last Hour</th>
<th>Rows Accepted Last Hour</th>
<th>Rows Rejected Last Hour</th>
<th>Errors Last Hour</th>
<th>End Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>greater than</td>
<td>greater than</td>
<td>greater than</td>
<td>greater than</td>
<td></td>
<td></td>
</tr>
<tr>
<td>less than</td>
<td>less than</td>
<td>less than</td>
<td>less than</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2405</td>
<td>2405</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>0</td>
<td>1305</td>
<td>1305</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>255</td>
<td>92310</td>
<td>92310</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>1365</td>
<td>451845</td>
<td>451845</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>1469</td>
<td>538294</td>
<td>538294</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>1040</td>
<td>497130</td>
<td>497130</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>2738</td>
<td>1051552</td>
<td>1051552</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
</tbody>
</table>
Monitoring

This microbatch is collecting from the following sources: dump1090_msg_2 (host:pennardpl10:9092)

**Details**
- Node name: v_pennarddell01_node0001
- SessionId: nnarddell01_node0001-10004:0x2fa9
- TransactionId: 45035996288966917
- StatementId: 7
- Batch Number: 0

**Rejection**
- Row Number: 1
- Rejected Data Orig Length: 113
- Rejected Reason: Invalid integer format '51.60221' for column 13 (is_on_ground)

Query 'dump1090_kafka.dump1090_msg_2_rej WHERE transaction_id = 45035996288966917 AND statement_id = 7' to get all the rejected rows.
How many? How fast?

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Microbatch</th>
<th>Source</th>
<th>Target Schema</th>
<th>Target Table</th>
<th>Kafka Cluster</th>
<th>Timestamp Batch-Start</th>
<th>Messages Last Batch</th>
<th>Messages Last Hour</th>
<th>Rows Accepted Last Hour</th>
<th>Rows Rejected Last Hour</th>
<th>Errors Last Hour</th>
<th>End Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump1090sche</td>
<td>dump1090_air</td>
<td>dump1090_air</td>
<td>dump1090_kafka</td>
<td>dump1090_air</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>0</td>
<td>189650</td>
<td>189650</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_id</td>
<td>dump1090_id</td>
<td>dump1090_kafka</td>
<td>dump1090_id</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>0</td>
<td>104121</td>
<td>104121</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_1</td>
<td>dump1090_msg_1</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_1</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>164215</td>
<td>5977750</td>
<td>5977750</td>
<td>0</td>
<td>0</td>
<td>deadline</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_2</td>
<td>dump1090_msg_2</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_2</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>0</td>
<td>927</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_3</td>
<td>dump1090_msg_3</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_3</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>10045</td>
<td>5001012</td>
<td>5001012</td>
<td>0</td>
<td>0</td>
<td>deadline</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_4</td>
<td>dump1090_msg_4</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_4</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>122785</td>
<td>5088580</td>
<td>5088580</td>
<td>0</td>
<td>0</td>
<td>deadline</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_5</td>
<td>dump1090_msg_5</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_5</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>117434</td>
<td>5525492</td>
<td>5525492</td>
<td>0</td>
<td>0</td>
<td>deadline</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_6</td>
<td>dump1090_msg_6</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_6</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>127092</td>
<td>5345799</td>
<td>5345799</td>
<td>0</td>
<td>0</td>
<td>deadline</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_7</td>
<td>dump1090_msg_7</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_7</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_msg_8</td>
<td>dump1090_msg_8</td>
<td>dump1090_kafka</td>
<td>dump1090_msg_8</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>173224</td>
<td>5842357</td>
<td>5842357</td>
<td>0</td>
<td>0</td>
<td>deadline</td>
</tr>
<tr>
<td>dump1090sche</td>
<td>dump1090_sta</td>
<td>dump1090_sta</td>
<td>dump1090_kafka</td>
<td>dump1090_sta</td>
<td>pennardpi_cluster</td>
<td>May 21, 2017 05:29</td>
<td>10</td>
<td>3793396</td>
<td>3793396</td>
<td>0</td>
<td>0</td>
<td>end of stream</td>
</tr>
</tbody>
</table>
How many? How fast?

- Micro batch: 10 seconds
- Messages loaded into Vertica in last micro batch: 814,225
  - 4.9 million / minute
  - 293 million / hour
  - 7 billion / day

- All from:
  - 1x Raspberry PI (1 core, 0.5GB RAM) running DUMP1090
  - 1x Raspberry PI (1 core, 0.5GB RAM) running ETL, Zookeeper & Kafka
  - 1x Laptop (8 core, 16GB RAM) CentOS 7 running:
    - Vertica (single node)
How many? How fast?

Impressive Parallel COPY Performance
Loaded 2.42 Billion Rows (451 GB)
in 7min 35sec on an 8 Node Cluster

Key Takeaways
- Parallel Kafka Reads to Spark RDD (in memory) with
  Parallel writes to a Vertica via tcp server – ROCKS1
- COPY 36 TB/Hour with 81 Node cluster
- No ephemeral nodes needed for ingest
- Kafka read parallelism to Spark RDD partitions
- A priori hash() in Spark RDD Partitions (in Memory)
- TCP Server as a Vertica User Define Copy Source
- Single COPY does not preallocate Memory across nodes

* 270 Nodes (45 Ingest Nodes + 225 Data Nodes [225 ?])
Agenda

- Background to “The Lab Series” and the Big Data & Machine Learning Meetups
- Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
- What’s next?
Application Integration

Vertica

INFRASTRUCTURE / CLOUD
- VMware
- Amazon Web Services
- Red Hat Linux
- Full 360
- Advizex
- Optimal Test
- Fidelis
- IQ IneoQuest
- Vichara Technologies
- Joyent
- Right Scale
- Hewlett Packard Enterprise
- GoodData
- Emulex
- Talend
- Informatica
- TIBCO

BI / VISUALIZATION
- MicroStrategy
- Tableau
- Qlik
- Spotfire
- Pentaho
- Logi Analytics
- Looker
- MapR
- Hortonworks
- Cloudera
- Apache
- Syncsort

HADOOP

DATA INTEGRATION

OEM
Altitude
Geospatial
High vs Low Altitude
Tracking a Single Flight (STK43BR)

STK43BR - Bristol to Cork
Shell, SQL, HTML and Google Maps API

```c
printf "\n" "Usage:"
printf "\n" "$h\_prog\_name"
printf "\n" "--action {start|stop}"
printf "\n" "--logfile {DUMP1090 log file}"
printf "\n" "--schema {schema}"
printf "\n" "--time\_slice\_number {time slice number}"
printf "\n" "--time\_slice\_units {time slice units}"
printf "\n" "--verbose {Y|N}"
printf "\n" "--sleep {number of seconds to sleep between re-runs}"
```

```sql
select
date_trunc('minute', max(msg_gen_ts)) as end_time,
date_trunc('minute', max(msg_gen_ts) - interval '$h\_clv\_time\_slice\_number $h\_clv\_time\_slice\_units') as start_time
from
${h\_clv\_schema}.dump1090_msg_5
```

```html
```
Shell, SQL, HTML and Google Maps API

create local temporary table if not exists dump1090_msg_1 on commit preserve rows as
select distinct
    msg_1.hex_ident as hex_ident,
    max(msg_1.msg_gen_ts) as msg_gen_ts,
    max(msg_1.call_sign) as call_sign
from
    ${h_clv_schema}.dump1090_msg_1 msg_1
where
    msg_1.msg_gen_ts between '${h_clv_start_time}' and '${h_clv_end_time}'
group by
    hex_ident
;

create local temporary table if not exists dump1090_msg_3 on commit preserve rows as
select distinct
    msg_3.hex_ident as hex_ident,
    max(msg_3.msg_gen_ts) as msg_gen_ts,
    max(msg_3.altitude) as altitude,
    max(msg_3.latitude) as latitude,
    max(msg_3.longitude) as longitude
from
    ${h_clv_schema}.dump1090_msg_3 msg_3
where
    msg_3.msg_gen_ts between '${h_clv_start_time}' and '${h_clv_end_time}'
group by
    hex_ident
;
Shell, SQL, HTML and Google Maps API

```html
<html>
  <head>
    <style type="text/css">
      html, body { height: 100%; margin: 0; padding: 0; }
      #map { height: 100%; }
    </style>
    <meta http-equiv="refresh" content="10" />
  </head>
  <body>
    <div id="map"></div>
    <script type="text/javascript">
      var map;
      function initMap()
      {
        var myHomeLatLng = {lat: 59.331, lng: 18.031};
        map = new google.maps.Map(document.getElementById('map'), {
          center: myHomeLatLng,
          zoom: 9,
          panControl: true,
          zoomControl: true,
          zoomControlOptions: {
            style: google.maps.ZoomControlStyle.LARGE,
            position: google.maps.ControlPosition.RIGHT_CENTER
          }
        });
        var planeImage_000_045 = 'black_plane_000_045.gif';
        var planeImage_045_090 = 'black_plane_045_090.gif';
        var planeImage_090_135 = 'black_plane_090_135.gif';
        var planeImage_135_180 = 'black_plane_135_180.gif';
        var planeImage_180_225 = 'black_plane_180_225.gif';
        var planeImage_225_270 = 'black_plane_225_270.gif';
        var planeImage_270_315 = 'black_plane_270_315.gif';
        var planeImage_315_360 = 'black_plane_315_360.gif';
      }
    </script>
  </body>
</html>
```
Shell, SQL, HTML and Google Maps API

while read h_hex_ident h_date_msg_gen h_time_msg_gen h_altitude h_latitude h_longitude h_track h_call_sign do

(( h_nooff_aircraft == 1 ))

if ((0<=h_track & h_track<=44))
then
  h_aircraft="planeImage_000_045"
elif ((45<=h_track & h_track<=89))
then
  h_aircraft="planeImage_045_090"
elif ((90<=h_track & h_track<=134))
then
  h_aircraft="planeImage_090_135"
elif ((135<=h_track & h_track<=179))
then
  h_aircraft="planeImage_135_180"
else
  h_aircraft="planeImage_180_224"

soho "var plane$h_nooff_aircraft = new google.maps.Marker({
  position: (lat: $h_latitude, lng: $h_longitude),
  map: map,
  title: '$h_call_sign - Alt: $h_altitude Lat: $h_latitude Long: $h_longitude Dt: $h_date_msg_gen Tm: $h_time_msg_gen HexId: $h_hex_ident',
  icon: $h_aircraft
});" >> $h_clf_dump1090_flights_html_tmp

<script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSy-1Qw6aKv2ZCvLU8&region=GB&callback=initMap"></script>
Shell, SQL, HTML and Google Maps API
Agenda

▪ Background to “The Lab Series” and the Big Data & Machine Learning Meetups
▪ Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - **Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy**
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
▪ What’s next?
Vertica Analytical Capabilities

**SQL ‘99**
- Aggregate
- Analytical
- Window functions
- Date/Time functions
- String functions
- Mathematical functions

Allows for:
- Standard functionality that performs at scale

**SQL Extensions**
- Pattern matching
- Event series joins
- Time series
- Event-based windows

Allows for:
- Sessionization
- Conversion analysis
- Fraud detection
- Fast Aggregates (LAP)

**SDKs**
- Analytics
  - C++
  - Java
  - R
  - Python
- Connection
  - ODBC/JDBC
  - HIVE
  - Hadoop
  - Flex zone

Allows for:
- Specialized parsers
- Custom data mining
- Semi-structured data processing

**In-database Analytics**
- Regression
- K-means
- Statistical modeling
- Classification algorithms
- Text mining
- Geospatial

Allows for:
- Statistical modeling
- Cluster analysis
- Predictive analytics
- Geospatial analysis
## Rich Set of Tools to Get Data Ready for Modeling

### Capture & Enrich
- Copy
- Flex Tables
- External Tables
- Parsers: Avro, CEF, CSV, Delim, JSON, RegEX
- Streaming Utilities including Kafka Integration
- S3 & ABS
- ORC, Parquet, HIVE, Spark RDD & DF
- Shapefiles & Spatial Data

### Measure & Prepare
- 1000s of functions
- **Time Series Prep** (GFI, Interpolation, Slicing, TSA)
  - **Sessionize**
  - Pattern Matching
  - Event Series Joins
  - Advanced Aggregation
  - Date & Time Algebra
  - Window & Partition
  - Stats & Math
  - Data Type Handling
- Strings
- Sequences
- Geospatial, Joins, Conversions
- Balance
- Sampling
- **Outlier Detection**
- Normalize
- Missing Value Imputation

### Model & Deploy
- Linear Regression
- Logistic Regression
- K-Means
- Naïve Bayes
- SVM
- Model Evaluation & Visualization
- Model Management
- UDX Functions
- Text Analytics

### Additional Features
- ANSI SQL Standards – Algorithms Developed for MPP Execution – Relational Structure at PB Scale
Agenda

▪ Background to “The Lab Series” and the Big Data & Machine Learning Meetups
▪ Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - **Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization**
▪ What’s next?
Example use case – Outlier Detection
Example use case – Outlier Detection

```sql
/*
 * Table counts
 */
SELECT COUNT(*)
FROM dump1090_batch.dump1090_msg_3;
```

| COUNT | 148,342,592 |
Example use case – Outlier Detection

```sql
/*
 * Data Exploration
 */

SELECT DISTINCT hex_ident, msg_gen_ts, altitude, latitude, longitude
FROM dump1090_batch.dump1090_msg_3
ORDER BY 1, 2
LIMIT 10;
```

<table>
<thead>
<tr>
<th>hex_ident</th>
<th>msg_gen_ts</th>
<th>altitude</th>
<th>latitude</th>
<th>longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:31</td>
<td>19,000</td>
<td>50.66429</td>
<td>-0.3768</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:32</td>
<td>19,000</td>
<td>50.66373</td>
<td>-0.37606</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:38</td>
<td>19,150</td>
<td>50.65498</td>
<td>-0.36464</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:40</td>
<td>19,200</td>
<td>50.65196</td>
<td>-0.36074</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:43</td>
<td>19,275</td>
<td>50.64786</td>
<td>-0.35535</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:44</td>
<td>19,300</td>
<td>50.64646</td>
<td>-0.35357</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:45</td>
<td>19,350</td>
<td>50.64455</td>
<td>-0.35106</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:51</td>
<td>19,500</td>
<td>50.63736</td>
<td>-0.3418</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 18:59:52</td>
<td>19,525</td>
<td>50.63599</td>
<td>-0.33998</td>
</tr>
<tr>
<td>00B203</td>
<td>2017-06-27 19:00:00</td>
<td>19,725</td>
<td>50.62468</td>
<td>-0.32533</td>
</tr>
</tbody>
</table>
Example use case – Outlier Detection

```sql
/*
  * Distribution of altitude of aircraft (identified by its registration number HEX_IDENT)
  */

SELECT
  a.NOOf_FLIGHTS,
  a.MIN_ALT,
  b.P05,
  b.P10,
  b.P25,
  b.MEDIAN,
  b.P75,
  b.P90,
  b.P95,
  a.MAX_ALT,
  a.AVG_ALT
FROM

SELECT COUNT(*) AS NOOf_FLIGHTS,
  MIN(altitude) AS MIN_ALT,
  MAX(altitude) AS MAX_ALT,
  AVG(altitude) AS AVG_ALT
FROM
  (SELECT DISTINCT
    hex_ident, altitude
  FROM
dump1090_batch.dump1090_msg_3
ORDER BY hex_ident, altitude ASC) c
) b ;
```
Example use case – Outlier Detection

<table>
<thead>
<tr>
<th>NOOF_FLIGHTS</th>
<th>MIN_ALT</th>
<th>P05</th>
<th>P10</th>
<th>P25</th>
<th>MEDIAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,599,427</td>
<td>-375</td>
<td>28,675</td>
<td>15,225</td>
<td>21,250</td>
<td>28,675</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P75</th>
<th>P90</th>
<th>P95</th>
<th>MAX_ALT</th>
<th>AVG_ALT</th>
</tr>
</thead>
<tbody>
<tr>
<td>34,375</td>
<td>37,675</td>
<td>39,125</td>
<td>124,400</td>
<td>27,396.1801003326</td>
</tr>
</tbody>
</table>
Example use case – Outlier Detection

/*
 * Feature Creation – Detect Outliers
 * (using a z-score of 5.0 or higher)
 */

DROP TABLE IF EXISTS dump1090_batch.dump1090_msg_3_outliers;

SELECT DETECT_OUTLIERS('dump1090_batch.dump1090_msg_3_outliers',
    'dump1090_batch.dump1090_msg_3',
    'altitude',
    'robust_zscore' USING PARAMETERS outlier_threshold=5.0, key_columns='hex_ident');

Detected 1741503 outliers
Example use case – Outlier Detection

```sql
-- View the results

SELECT altitude, COUNT(*)
FROM dump1090_batch.dump1090_msg_3_outlier
GROUP BY 1 ORDER BY 1 ASC LIMIT 10;
```

<table>
<thead>
<tr>
<th>altitude</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>-375</td>
<td>27</td>
</tr>
<tr>
<td>-350</td>
<td>60</td>
</tr>
<tr>
<td>-325</td>
<td>109</td>
</tr>
<tr>
<td>-300</td>
<td>91</td>
</tr>
<tr>
<td>-275</td>
<td>125</td>
</tr>
<tr>
<td>-250</td>
<td>271</td>
</tr>
<tr>
<td>-225</td>
<td>115</td>
</tr>
<tr>
<td>-200</td>
<td>121</td>
</tr>
<tr>
<td>-175</td>
<td>141</td>
</tr>
<tr>
<td>-150</td>
<td>261</td>
</tr>
</tbody>
</table>
Example use case – Outlier Detection

```
SELECT altitude, COUNT(*)
FROM dump1090_batch.dump1090_msg_3_outliers
GROUP BY 1 ORDER BY 1 DESC LIMIT 10;
```

<table>
<thead>
<tr>
<th>altitude</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>124,400</td>
<td>3</td>
</tr>
<tr>
<td>124,200</td>
<td>2</td>
</tr>
<tr>
<td>123,100</td>
<td>4</td>
</tr>
<tr>
<td>122,200</td>
<td>2</td>
</tr>
<tr>
<td>120,400</td>
<td>4</td>
</tr>
<tr>
<td>120,100</td>
<td>1</td>
</tr>
<tr>
<td>119,100</td>
<td>7</td>
</tr>
<tr>
<td>118,400</td>
<td>4</td>
</tr>
<tr>
<td>117,100</td>
<td>5</td>
</tr>
<tr>
<td>116,400</td>
<td>8</td>
</tr>
</tbody>
</table>
From Swansea to Switzerland
From Swansea to Switzerland
Installed in July 2017
58 Days: 1bn Messages, 12k aircraft, 200,000 KM² (*)

<table>
<thead>
<tr>
<th>MSG Type</th>
<th>First TS</th>
<th>Latest TS</th>
<th>Distinct Hex Ident</th>
<th>No of Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIR</td>
<td>[NULL]</td>
<td>[NULL]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ID</td>
<td>[NULL]</td>
<td>[NULL]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MSG_1</td>
<td>2017-07-17 22:01:12</td>
<td>2017-09-12 11:06:39</td>
<td>8,163</td>
<td>6,234,095</td>
</tr>
<tr>
<td>MSG_2</td>
<td>[NULL]</td>
<td>[NULL]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MSG_3</td>
<td>2017-07-17 22:01:47</td>
<td>2017-09-12 11:06:40</td>
<td>7,705</td>
<td>61,756,974</td>
</tr>
<tr>
<td>MSG_4</td>
<td>2017-07-17 22:01:47</td>
<td>2017-09-12 11:06:39</td>
<td>7,769</td>
<td>62,167,725</td>
</tr>
<tr>
<td>MSG_5</td>
<td>2017-07-17 22:01:47</td>
<td>2017-09-12 11:06:40</td>
<td>11,665</td>
<td>208,449,533</td>
</tr>
<tr>
<td>MSG_6</td>
<td>2017-07-17 22:01:47</td>
<td>2017-09-12 11:06:40</td>
<td>11,512</td>
<td>69,780,084</td>
</tr>
<tr>
<td>MSG_7</td>
<td>2017-07-17 22:01:48</td>
<td>2017-09-12 11:06:40</td>
<td>11,694</td>
<td>213,866,386</td>
</tr>
<tr>
<td>MSG_8</td>
<td>2017-07-17 22:01:48</td>
<td>2017-09-12 11:06:40</td>
<td>11,799</td>
<td>398,037,717</td>
</tr>
<tr>
<td>STA</td>
<td>[NULL]</td>
<td>[NULL]</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Let’s start with just one aircraft

- **Call Sign**  - G-EUPA
- **Mode-S**     - 400801
- **Aircraft type** - Airbus A319-131
- **Build date**  - September 1999 (18 years old)
- **Serial Number** - 1082
- **Airline**     - BA / BAW
- **Livery**      - Olympic Dove
How many Type-3 messages?

```sql
/*
 * Count the number of MSG_3 messages for this one aircraft.
 */
SELECT count(*) AS noof_messages
FROM dump1090_kafka.dump1090_msg_3 msg_3
WHERE msg_3.hex_ident = '400801'
```

<table>
<thead>
<tr>
<th>noof_messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>13,719</td>
</tr>
</tbody>
</table>
Take a closer look at these Type-3 messages

```sql
/*
 * Take a look at these messages in more detail
 */
SELECT
  msg_3.hex_ident AS msg_3_hex_ident,
  msg_3.msg_gen_ts AS msg_3_msg_gen_ts,
  msg_3.altitude AS msg_3_altitude,
  msg_3.latitude AS msg_3_latitude,
  msg_3.longitude AS msg_3_longitude
FROM
dump1090_kafka.dump1090_msg_3 msg_3
WHERE
  msg_3.hex_ident = '400801'
ORDER BY
  msg_3.msg_gen_ts
```
Take a closer look at these Type-3 messages

<table>
<thead>
<tr>
<th>msg_3_msg_gen_ts</th>
<th>msg_3_altitude</th>
<th>msg_3_latitude</th>
<th>msg_3_longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2017-07-21 13:38:00</td>
<td>16,100</td>
<td>47.44213</td>
<td>7.97107</td>
</tr>
<tr>
<td>2 2017-07-21 13:38:04</td>
<td>16,275</td>
<td>47.44249</td>
<td>7.96083</td>
</tr>
<tr>
<td>3 2017-07-21 13:38:13</td>
<td>16,675</td>
<td>47.44318</td>
<td>7.94133</td>
</tr>
<tr>
<td>4 2017-07-21 13:38:14</td>
<td>16,750</td>
<td>47.44331</td>
<td>7.93742</td>
</tr>
<tr>
<td>5 2017-07-21 13:38:16</td>
<td>16,850</td>
<td>47.44345</td>
<td>7.93247</td>
</tr>
<tr>
<td>6 2017-07-21 13:38:17</td>
<td>16,900</td>
<td>47.44353</td>
<td>7.93009</td>
</tr>
<tr>
<td>64 2017-07-21 13:47:30</td>
<td>34,250</td>
<td>47.65279</td>
<td>6.43016</td>
</tr>
<tr>
<td>65 2017-07-21 13:47:35</td>
<td>34,325</td>
<td>47.65557</td>
<td>6.41742</td>
</tr>
<tr>
<td>66 2017-07-21 13:47:38</td>
<td>34,400</td>
<td>47.65764</td>
<td>6.40798</td>
</tr>
<tr>
<td>68 2017-07-24 22:23:18</td>
<td>13,975</td>
<td>46.54559</td>
<td>5.83897</td>
</tr>
<tr>
<td>69 2017-07-24 22:23:26</td>
<td>13,800</td>
<td>46.53415</td>
<td>5.85055</td>
</tr>
<tr>
<td>70 2017-07-24 22:23:28</td>
<td>13,750</td>
<td>46.53186</td>
<td>5.8529</td>
</tr>
</tbody>
</table>
Visualising the flight tracks of aircraft “400801”
Visualising the flight tracks of aircraft “400801”
Vertica’s time series Gap Filling & Interpolation

```sql
SELECT
gfi_msg_3_msg_gen_ts AS msg_3_msg_gen_ts,
gfi_altitude_int AS altitude,
gfi_longitude AS longitude,
gfi_latitude AS latitude
FROM
(
    SELECT
        gfi_msg_3_msg_gen_ts AS gfi_msg_3_msg_gen_ts,
        TS_FIRST_VALUE(msg_3.altitude, 'LINEAR')::INT AS gfi_altitude_int,
        TS_FIRST_VALUE(msg_3.latitude, 'LINEAR') AS gfi_latitude,
        TS_FIRST_VALUE(msg_3.longitude, 'LINEAR') AS gfi_longitude
    FROM
dump1090_kafka.dump1090_msg_3_msg_3
    WHERE
        msg_3.hex_ident = '400801'
    TIMESERIES
        gfi_msg_3_msg_gen_ts AS '1 seconds'
    OVER
        (PARTITION BY
            msg_3.hex_ident
        ORDER BY
            msg_3.msg_gen_ts::TIMESTAMP(0))
) a
```
The result of applying time series GFI

<table>
<thead>
<tr>
<th>msg_3_msg_gen_ts</th>
<th>altitude</th>
<th>longitude</th>
<th>latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2017-07-21 13:38:01</td>
<td>16,100</td>
<td>7.97107</td>
<td>47.44213</td>
</tr>
<tr>
<td>2 2017-07-21 13:38:02</td>
<td>16,144</td>
<td>7.96851</td>
<td>47.44222</td>
</tr>
<tr>
<td>3 2017-07-21 13:38:03</td>
<td>16,188</td>
<td>7.96595</td>
<td>47.44231</td>
</tr>
<tr>
<td>4 2017-07-21 13:38:04</td>
<td>16,231</td>
<td>7.96339</td>
<td>47.4424</td>
</tr>
<tr>
<td>5 2017-07-21 13:38:05</td>
<td>16,275</td>
<td>7.96083</td>
<td>47.44249</td>
</tr>
<tr>
<td>634 2017-07-21 13:48:34</td>
<td>35,375</td>
<td>6.2561471237</td>
<td>47.690992103</td>
</tr>
<tr>
<td>635 2017-07-21 13:48:35</td>
<td>35,375</td>
<td>6.2561456856</td>
<td>47.6909881544</td>
</tr>
<tr>
<td>636 2017-07-21 13:48:36</td>
<td>35,375</td>
<td>6.2561442475</td>
<td>47.6909842059</td>
</tr>
<tr>
<td>637 2017-07-21 13:48:37</td>
<td>35,375</td>
<td>6.2561471237</td>
<td>47.690992103</td>
</tr>
</tbody>
</table>
Introducing Sessions

```sql
SELECT
    msg_3.hex_ident,  
    msg_3.msg_gen_ts,  
    msg_3.altitude,  
    msg_3.latitude,  
    msg_3.longitude  
AS msg_3_hex_ident,  
AS msg_3_msg_gen_ts,  
AS msg_3_altitude,  
AS msg_3_latitude,  
AS msg_3_longitude,

CONDITIONAL_TRUE_EVENT
    (msg_3.msg_gen_ts - LAG(msg_3.msg_gen_ts) > '1 hour')
OVER
    (PARTITION BY
        msg_3.hex_ident
    ORDER BY
        msg_3.msg_gen_ts)
AS msg_3_flight

FROM
dump1090_kafka.dump1090_msg_3 msg_3
WHERE
    msg_3.hex_ident = '400801'
) msg_3
```
Introducing Sessions

```
SELECT
    gfi_msg_3.msg_gen_ts AS msg_3_msg_gen_ts,
    gfi_altitude_int AS altitude,
    gfi_longitude AS longitude,
    gfi_latitude AS latitude,
    msg_3_flight AS flight
FROM
    (SELECT
        msg_3_hex_ident
    FROM
gf_int
    )
WHERE
dump1000_kafka.dump10000_msg_3 msg_3
```

```
SELECT
    msg_3_hex_ident AS msg_3_hex_ident,
    msg_3_msg_gen_ts AS msg_3_msg_gen_ts,
    msg_3_altitude AS msg_3_altitude,
    msg_3_latitude AS msg_3_latitude,
    msg_3_longitude AS msg_3_longitude,
    CONDITIONAL_TRUE_EVENT
FROM
gf_int
WHERE
dump1000_kafka.dump10000_msg_3 msg_3
ORDER BY
gf_int.msg_3_msg_gen_ts::TIMESTAMP(0)
```

```
SELECT
    gfi_msg_3.msg_gen_ts AS msg_3_msg_gen_ts
FROM
gf_int
```
Result of sessionising our GFI data

<table>
<thead>
<tr>
<th>msg_3_msg_gen_ts</th>
<th>altitude</th>
<th>longitude</th>
<th>latitude</th>
<th>flight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2017-07-21 13:38:01</td>
<td>16,100</td>
<td>7.97107</td>
<td>47.44213</td>
<td>0</td>
</tr>
<tr>
<td>2 2017-07-21 13:38:02</td>
<td>16,144</td>
<td>7.96851</td>
<td>47.44222</td>
<td>0</td>
</tr>
<tr>
<td>3 2017-07-21 13:38:03</td>
<td>16,188</td>
<td>7.96595</td>
<td>47.44231</td>
<td>0</td>
</tr>
<tr>
<td>4 2017-07-21 13:38:04</td>
<td>16,231</td>
<td>7.96339</td>
<td>47.4424</td>
<td>0</td>
</tr>
<tr>
<td>5 2017-07-21 13:38:05</td>
<td>16,275</td>
<td>7.96083</td>
<td>47.44249</td>
<td>0</td>
</tr>
<tr>
<td>6 2017-07-21 13:38:06</td>
<td>16,325</td>
<td>7.9583925</td>
<td>47.44257625</td>
<td>0</td>
</tr>
<tr>
<td>632 2017-07-21 13:48:32</td>
<td>35,357</td>
<td>6.2589616667</td>
<td>47.6903822222</td>
<td>0</td>
</tr>
<tr>
<td>633 2017-07-21 13:48:33</td>
<td>35,375</td>
<td>6.25615</td>
<td>47.691</td>
<td>0</td>
</tr>
<tr>
<td>634 2017-07-24 22:23:19</td>
<td>13,975</td>
<td>5.83897</td>
<td>46.54559</td>
<td>1</td>
</tr>
<tr>
<td>635 2017-07-24 22:23:20</td>
<td>13,953</td>
<td>5.8404175</td>
<td>46.54416</td>
<td>1</td>
</tr>
<tr>
<td>636 2017-07-24 22:23:21</td>
<td>13,931</td>
<td>5.841865</td>
<td>46.54273</td>
<td>1</td>
</tr>
<tr>
<td>27318 2017-09-07 15:24:47</td>
<td>26,225</td>
<td>5.55071</td>
<td>46.81439</td>
<td>44</td>
</tr>
<tr>
<td>27319 2017-09-07 15:24:48</td>
<td>26,250</td>
<td>5.54877</td>
<td>46.81508</td>
<td>44</td>
</tr>
</tbody>
</table>
What our flights tracks looked like...
... and what they look like with GFI and sessions
Agenda

▪ Background to “The Lab Series” and the Big Data & Machine Learning Meetups
▪ Covered so far on Project #1:
  - Introduction to Automatic Dependent Surveillance – Broadcast (ADS-B)
  - Using a Raspberry Pi to capture and decode ADS-B signals
  - DUMP1090 – Live tracking and streaming
  - Apache Kafka and Extract Transform & Load (ETL)
  - Introduction to Vertica
  - Kafka / Vertica integration and Management Console
  - Vertica integration tools and simple visualisations
  - Vertica data modelling tools – Capture & Enrich / Measure & Prepare / Model & Deploy
  - Measure and Prepare: Outlier detection, gap filling & interpolation and sessionization
▪ What’s next?
What’s Next?

The next Big Data & Machine Learning Meetups

- Big Data and Machine Learning (London) – Meetup #8
  Date, time and venue TBC

- Big Data and Machine Learning (Munich) – Meetup #2
  Thursday 23rd November 2017 @ CGI Munich

- Big Data and Machine Learning (Cambridge) – Meetup #2 & #3
  Wednesday 10th January 2018 @ Jagex, Cambridge
  Wednesday 4th April 2018 @ Jagex, Cambridge
Thank you

Mark.Whalley@microfocus.com

www.vertica.com
www.myvertica.com