# Big Data Project : Vessels Voyages

*By Chedly Zouche*

**National Engineering School of Carthage - Tunisia**

*2021 - 2022*

# Table of contents

01.

AIS

02.

About Apache Kylin

03.

Pipeline and project steps

04.

Conclusion

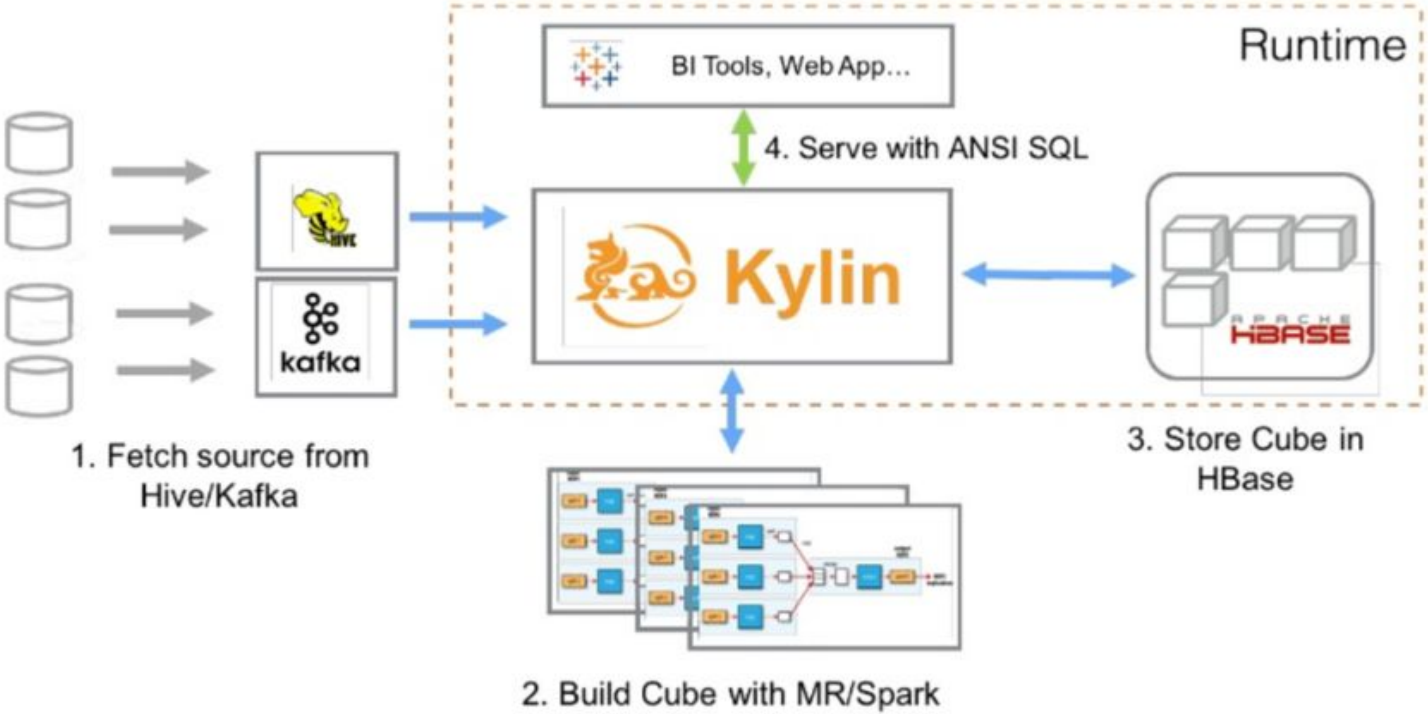
# AIS

- The automatic identification system (AIS) is an automatic tracking system that uses transceivers on ships and is used by vessel traffic services (VTS)
- The original purpose of AIS was solely collision avoidance but many other applications have since developed and continue to be developed. AIS is currently used for :
  - Collision avoidance
  - Fishing fleet monitoring and control
  - Maritime security
  - Accident investigation
  - Fleet and cargo tracking
  - Statistics and economics : Which is the case of this project, in fact historical data from AISs can help in mapping fishery activities, official Maritime statistics such as port visits and faster economic indicators like time in port and port traffic.

# About Apache Kylin

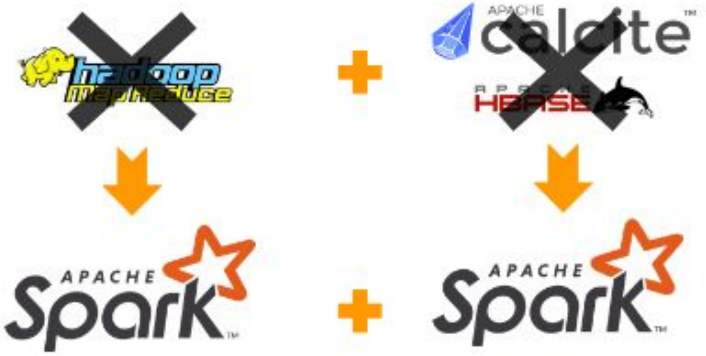
Apache Kylin is an open source online analytical processing (OLAP) engine for interactive analysis of big data. The platform was designed to provide a SQL interface and multidimensional analysis (MOLAP) on Hadoop/Spark.

# Architecture

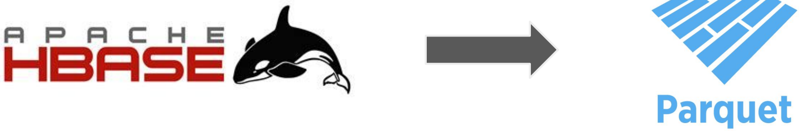


# Version 4 changes

## Build + Query



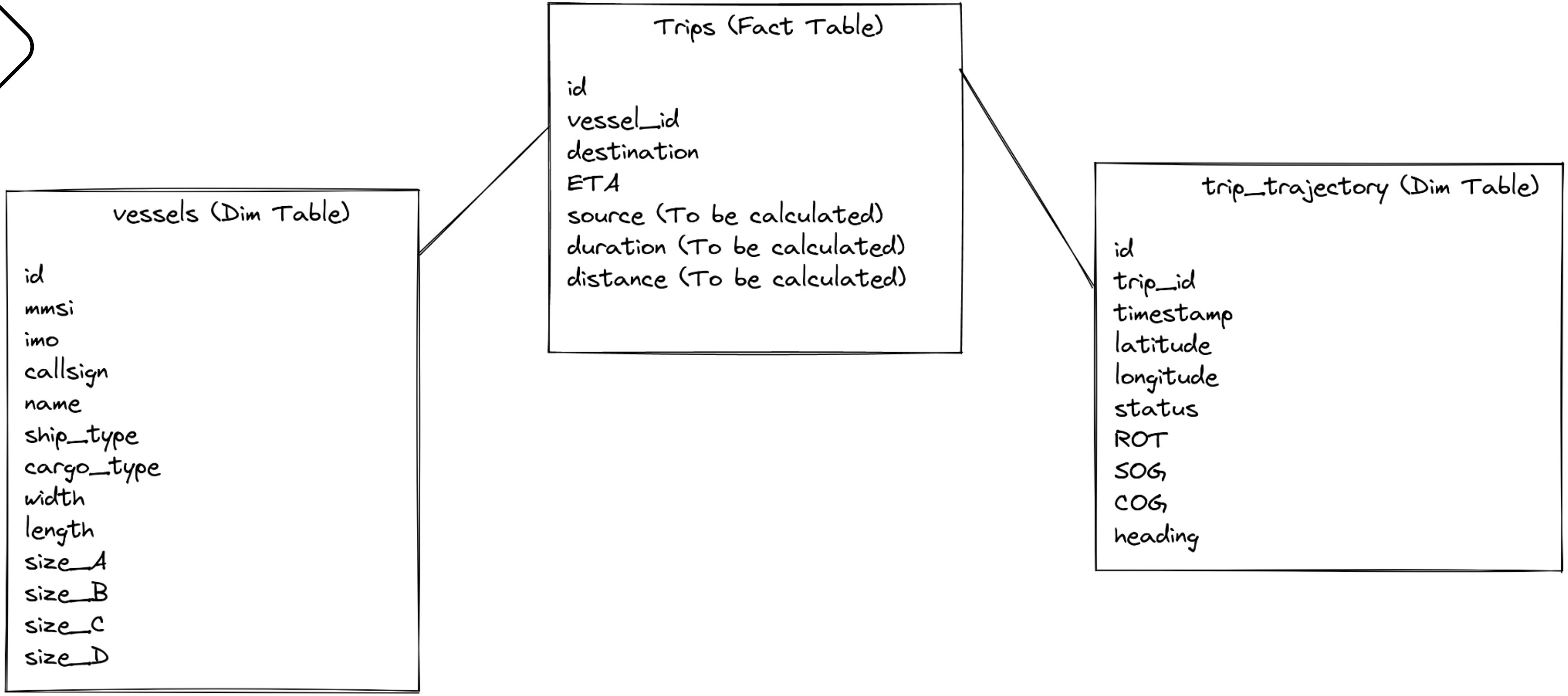
## Storage



# Pipeline

1. **Download the dataset of 1 day** from http://web.ais.dk/aisdata/ `=>` `/input/get-ais-data.sh`)
2. **Extracting the fact table and dimension tables into separate csv files and perform general cleanup (Remove missing data…)** `=>` `/input/generate_csvs.py`
3. **Create the database, tables and load data** `=>` `/input/hive-setup.sql`
4. **Import the database into Apache Kylin from the Web UI and create the models/cubes**.

# Data Warehouse structure



Trips (Fact Table)

id
vessel_id
destination
ETA
source (To be calculated)
duration (To be calculated)
distance (To be calculated)

vessels (Dim Table)

id
mmsi
imo
callsign
name
ship_type
cargo_type
width
length
size_A
size_B
size_C
size_D

trip_trajectory (Dim Table)

id
trip_id
timestamp
latitude
longitude
status
ROT
SOG
COG
heading

# Pipeline

5. **Calculate trip duration** => /queries/trip_duration.sql:

```
select trip_id, unix_timestamp(ts) - unix_timestamp(ts_first)
trip_duration
from (
 select trip_id,ts, first_value(ts) over (partition by trip_id
order by ts) ts_first, row_number() over (partition by trip_id
order by ts desc) lnr
 from trip_trajectory
) Sub
Where lnr=1;
```

# Pipeline

6. **Calculate trip start_date, average speed and distance** :

- We will need to add the ESRI functions to Hive first =>

  `/input/esri-setup.sh`
- Create the functions as listed in the `README` file
- Import the required jars from HDFS
  - `spatial-sdk-hive.jar`
  - `spatial-sdk-json.jar`
  - `esri-geometry-api.jar`

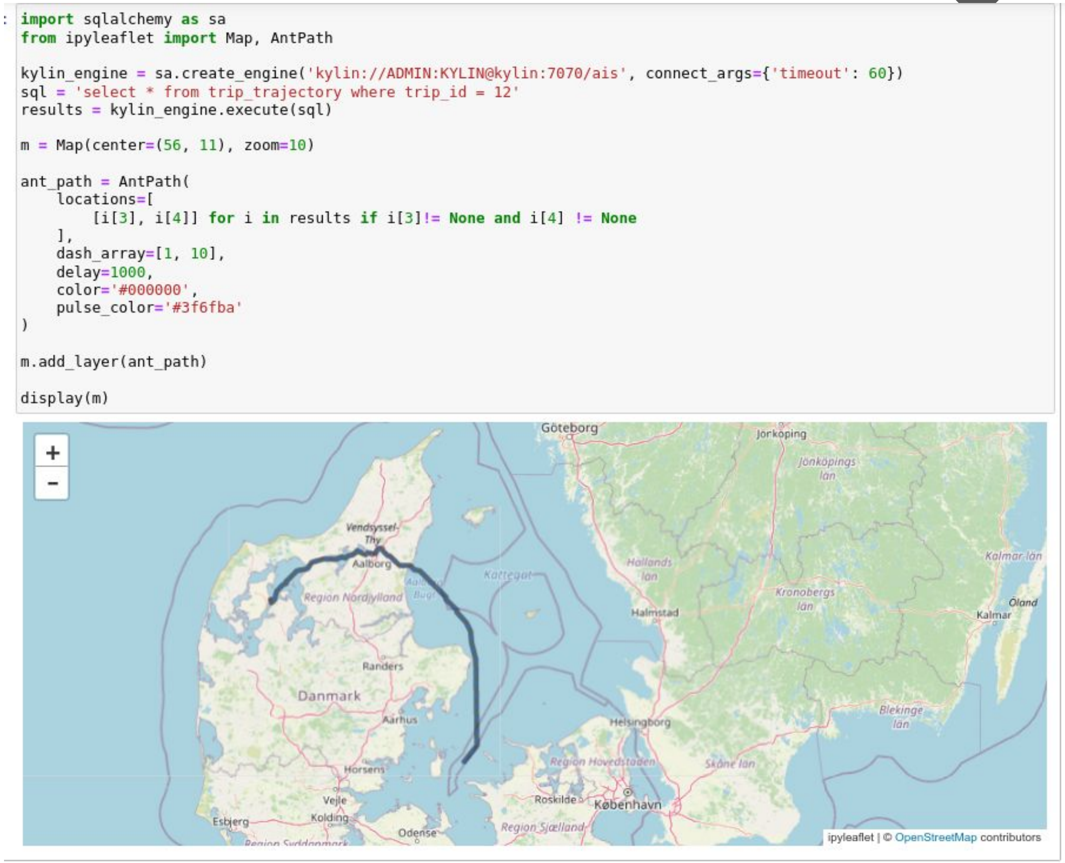
# Pipeline

6. **Calculate trip start_date, average speed and distance of each trip** => `/queries/trip-startdate_avgspeed_distance.sql`:

```
select trip_id, min(ts) start_time, max(ts) end_time, Sum(Distance) distance, Sum(Distance)/1000 *
(3600.0/Sum(time_passed)) avg_speed
from (
 select trip_id, ts, time_passed, ST_GeodesicLengthWGS84(ST_SetSRID(L,4326)) Distance
 from (
   select trip_id, ts, unix_timestamp(ts) - unix_timestamp(lag(ts,1) over (partition by trip_id order by ts))
time_passed, lat, lon, ST_LineString(prev_longitude, prev_latitude, lon, lat) L
   from (
     select trip_id, ts, lat, lon,
       lag(lat,1) over (partition by trip_id order by ts) prev_latitude,
       lag(lon,1) over (partition by trip_id order by ts) prev_longitude
     from trip_trajectory
   ) Sub
 ) Sub1
) Sub2
group by trip_id;
```

# Pipeline

7. **Draw trip trajectory on a map** (A jupyter notebook is provided)



```
import sqlalchemy as sa
from ipyleaflet import Map, AntPath

kylin_engine = sa.create_engine('kylin://ADMIN:KYLIN@kylin:7070/ais', connect_args={'timeout': 60})
sql = 'select * from trip_trajectory where trip_id = 12'
results = kylin_engine.execute(sql)

m = Map(center=(56, 11), zoom=10)

ant_path = AntPath(
    locations=[
        [i[3], i[4]] for i in results if i[3]!= None and i[4] != None
    ],
    dash_array=[1, 10],
    delay=1000,
    color='#000000',
    pulse_color='#3f6fba'
)

m.add_layer(ant_path)

display(m)
```

# To conclude

- Apache Kylin provides a better user experience of performing OLAP analysis using SQL compared to the traditional way of MapReduce with Hadoop
- The switch to Parquet for storage allowed for better disk space usage thanks to its encoding capabilities.
- Apache Kylin can also be easily used along with very popular data analysis libraries and tools such as Pandas thanks to its Python client.
- One drawback is that it doesn't offer a way to automate Model/Cube design via an API although it has support for ODBC, JDBC and Python as well as a REST API that can only be used to query data and build a Cube that was predefined.

# Thank you !

chedly.zouche@pm.me