POLITECNICO
MILANO 1863

Big Data Project
https://github.com/PaolaS/BigData

Middleware Technologies for Distributed Systems
Prof. Guinea Sam

Paola Sanfilippo 882892
Francesco Tinarelli 883738
Marco Wenzel 883732

Dataset

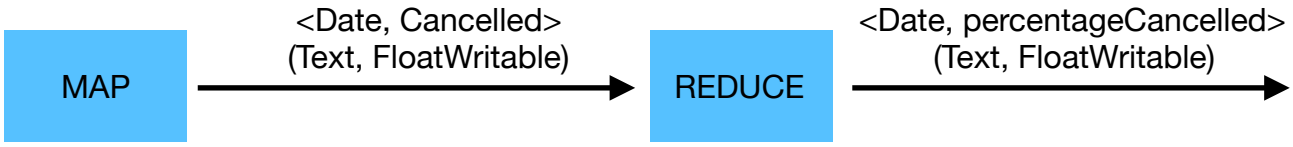
1 Year -> 1994-2008,
2 Month -> 1-12,
3 DayofMonth -> 1-31,
4 DayOfWeek -> 1 (Monday) - 7 (Sunday),
5 DepTime -> actual departure time (local, hhmm),
6 CRSDepTime -> scheduled departure time (local, hhmm),
7 ArrTime -> actual arrival time (local, hhmm),
8 CRSArrTime -> scheduled arrival time (local, hhmm),
9 UniqueCarrier -> unique carrier code,
10 FlightNum -> flight number,
11 TailNum -> plane tail number,
12 ActualElapsedTime -> in minutes,
13 CRSElapsedTime -> in minutes,
14 AirTime -> in minutes,
15 ArrDelay -> arrival delay, in minutes,
16 DepDelay -> departure delay, in minutes,
17 Origin -> origin IATA airport code,
18 Dest -> destination IATA airport code,
19 Distance -> in miles,
20 TaxiIn -> taxi in time, in minutes,
21 TaxiOut -> taxi out time in minutes,
22 Cancelled -> was the flight cancelled?,
23 CancellationCode -> reason for cancellation,
(A = carrier, B = weather, C = NAS, D = security),
24 Diverted -> 1 = yes, 0 = no,
25 CarrierDelay -> in minutes,
26 WeatherDelay -> in minutes,
27 NASDelay -> in minutes,
28 SecurityDelay -> in minutes,
29 LateAircraftDelay -> in minutes

Query 1

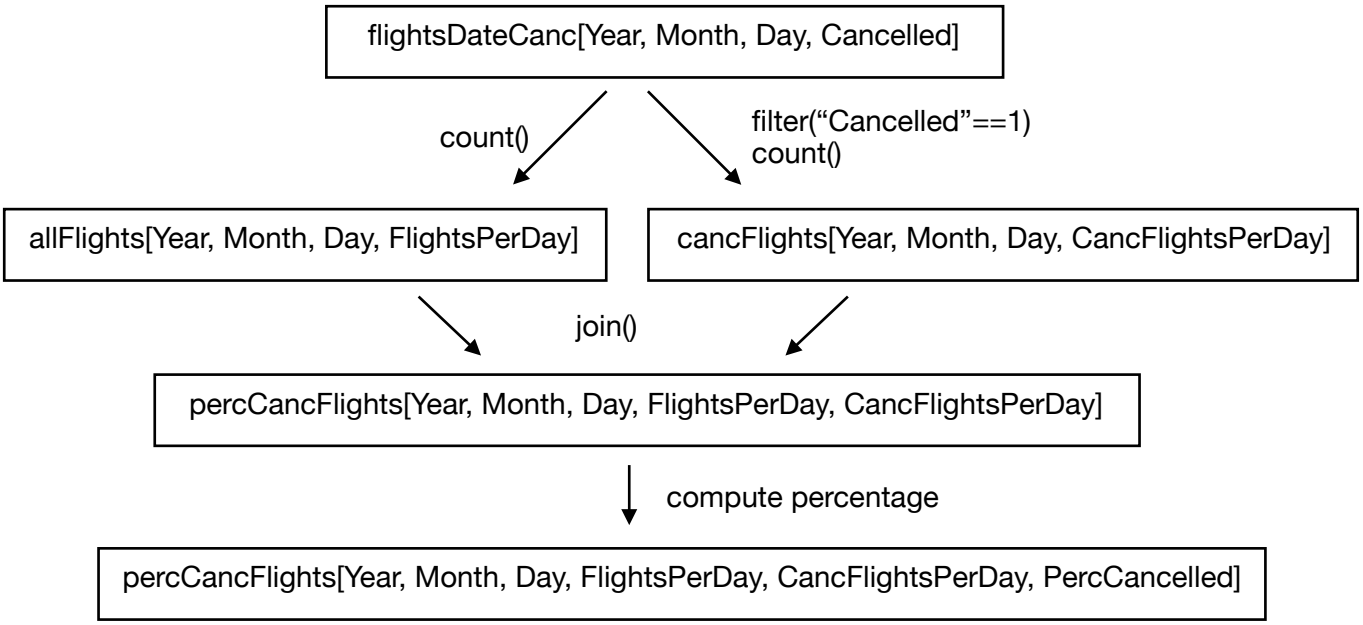
the percentage of canceled flights per day, throughout the entire data set

During the phase of data exploration, it was discovered that both information about the date and “cancelled” data don’t have any missing value.

Hadoop



Spark



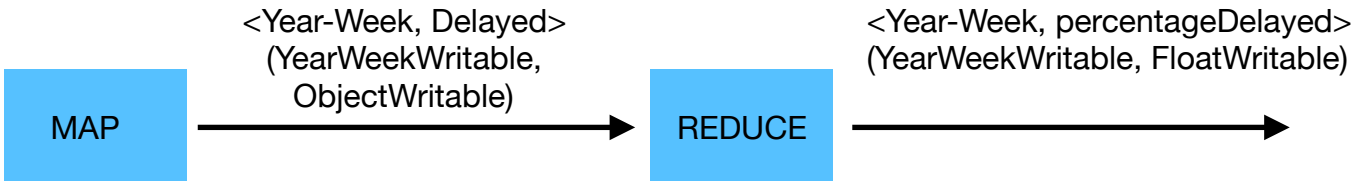
Query 2

weekly percentages of delays that are due to weather, throughout the entire data set

The percentage is computed as the ratio between the number of delays due to the weather and the total amount of flights.
In this case, for the second query we decided to include in the total amount also those flights that don't have any value for the WeatherDelay column, as it may mean that there's no delay. On the other side, obviously those data haven't been considered for the amount of delayed flights.

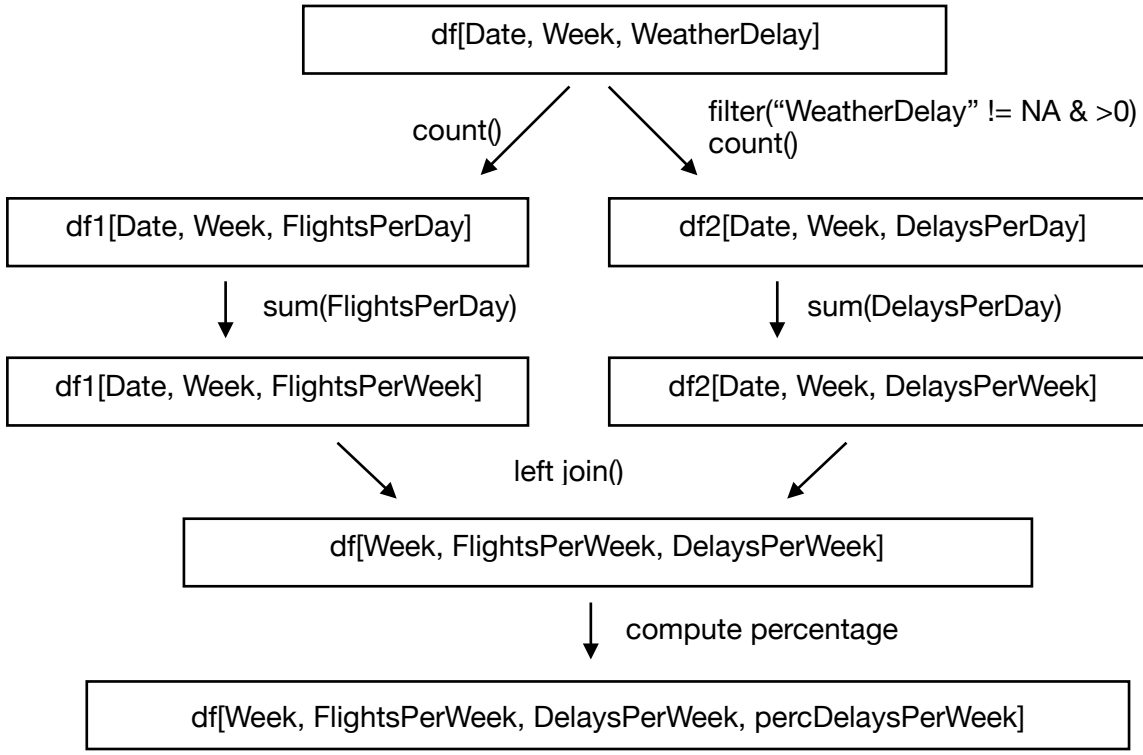
Hadoop

We introduced a new Writable called YearWeekWritable, built in this way: Year (Text) - Week (Text)



Spark

It has been chosen to work on dataframes with weeks from 1 to 52/53, instead of working on year based dataframes. The fixFirstRows, fixLastRows, fromMonToThu and fromThuToSun functions adjust the dataframe accordingly.
The dataframe d0 is used only to compute the first and last day for every week, so it is not included in the picture below. Moreover, in the illustration we also omitted the steps made by the loadYearCsv function to compute the "Date" column.



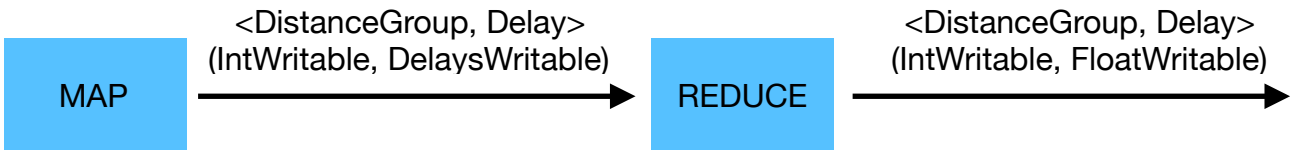
Query 3

the percentage of flights belonging to a given "distance group" that were able to halve their departure delays by the time they arrived at their destinations. Distance groups assort flights by their total distance in miles. Flights with distances that are less than 200 miles belong in group 1, flights with distances that are between 200 and 399 miles belong in group 2, flights with distances that are between 400 and 599 miles belong in group 3, and so on. The last group contains flights whose distances are between 2400 and 2599 miles

The percentage is computed as the ratio between the flights, whose DepartureDelay is greater or equal to the double of the ArrivalDelay, on the total amount of flights.
In this case we had to perform some data cleaning before manipulating the data: the flights that had missing values in the fields ArrivalDelay, DepartureDelay or Distance were discarded. In addition to this, also those rows with distance greater than 2599 miles.
We also decided to cut out also the entries with a negative DepartureDelay, as it isn't properly a "delay" (on the other side we kept those with a negative ArrivalDelay as a flight can take off late but make up for lost time and arrive a bit in advance of the scheduled time).

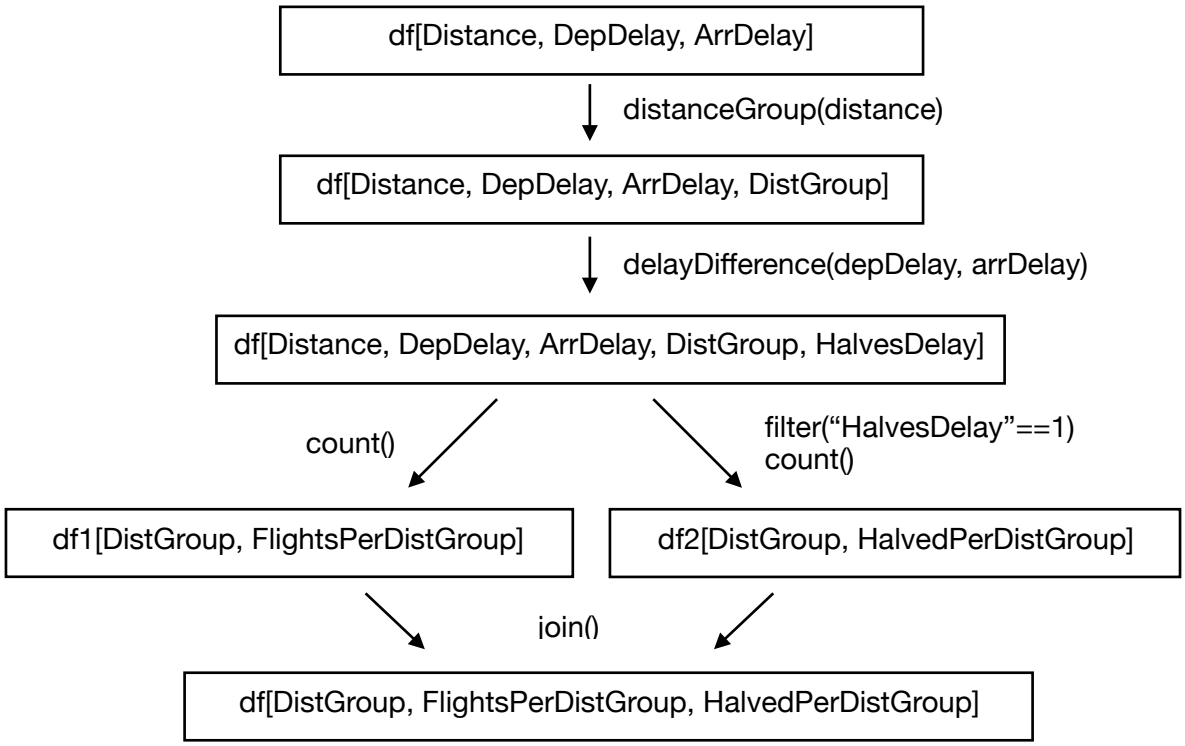
Hadoop

We introduced the DelaysWritable, that groups the DepartureDelay (FloatWritable) and the ArrivalDelay (FloatWritable).



Spark

We compute, for each year, the number of flights per distance group and the number of flights that managed to halve their delay.



In this way, we obtain the total number of flights and the halved ones for each distance group for every year.
Then we add all these entries to a single data frame, where we compute the percentages, through the whole dataset.

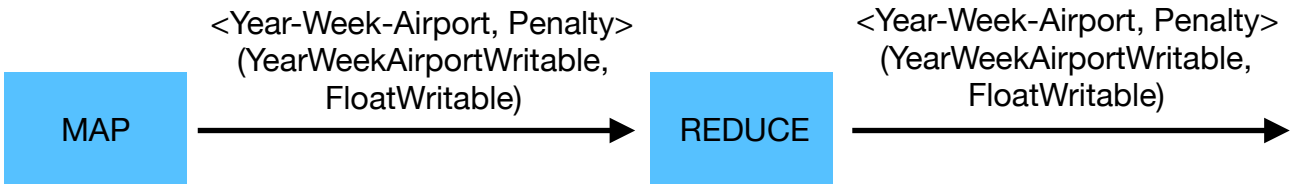
Query 4

a weekly "penalty" score for each airport that depends on both its incoming and outgoing flights. The score adds 0.5 for each incoming flight that is more than 15 minutes late, and 1 for each outgoing flight that is more than 15 minutes late

Hadoop

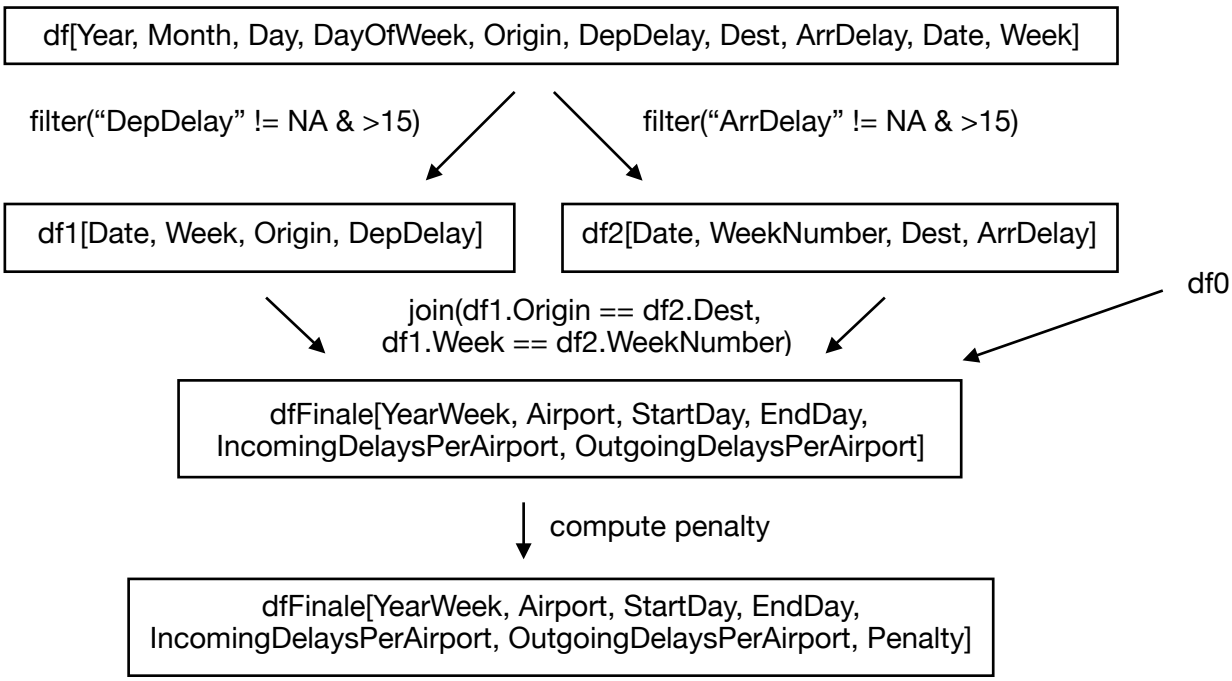
We created the YearWeekAirportWritable, made of 3 different Text: Year-Week-Airport. Each entry goes through two different checks:

- if neither the DepartureDelay nor the OriginAirport are missing and the DepartureDelay is greater than 15 minutes, then we create a pair with key YearWeekAirportWritable (Year-Week-OriginAirport) and value 1, that is the penalty for the departure delays, as a FloatWritable.
- if neither the ArrivalDelay nor the DestinationAirport are missing and the ArrivalDelay is greater than 15 minutes, then we create a pair with key YearWeekAirportWritable (Year-Week-DestinationAirport) and value 0.5, that is the penalty for the arrival delays, as a FloatWritable.



Spark

It has been chosen to work on dataframes with weeks from 1 to 52/53, instead of working on year based dataframes. The fixFirstRows, fixLastRows, fromMonToThu and fromThuToSun functions adjust the dataframe accordingly.
The function startEndWeek is used to add to each week the first and last day, saved in the d0 dataframe, so it is not included in the picture below. Moreover, in the illustration we also omitted the steps made by the loadYearCsv function to compute the “Date” column.



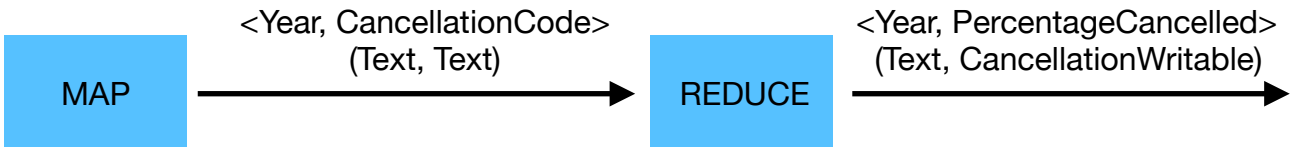
Query 5

the yearly percentage of flights cancelled for each reason (A = carrier, B = weather, C = NAS, D = security)

For every year, the percentage is computed as the ratio between the number of cancellation for a specific cancellation code and the total amount of flights that have been cancelled.
We decided to filter the cancelled flights keeping only those with a cancellation code (so we discarded those with missing values). Moreover, we choose to include in the count also those (few) flights that have a cancellation code and the “Cancelled” field equals to 0.

Hadoop

A new writable was created, in order to keep track of all the different percentages of cancellation.



Spark

We load the Year and the Cancellation Code. Then we create different dataframes to count the total amount of flights with a cancellation code (df0) and the amount for each cancellation code (dfA, dfB, dfC and dfD). Then all those data frames are joined together and the percentages are computed.