Université de Technologie de Compiègne
NF26 : Datawarehouse et outils décisionnels

Rapport de projet
Analyse de données de vols

Auteur :
Emma Parois

Semestre :
Printemps 2020

16 juin 2020

Table des matières

1 Introduction

Ce rapport retrace la réflexion, les analyses et les choix qui ont été effectués dans le cadre du projet de NF26. Ce projet avait pour objectif de manipuler et traiter un grand jeu de données proposé par une section de l'ASA en 2009 et accessible à cette page : `http://stat-computing.org/dataexpo/2009/`. Ce jeu de données contient de nombreuses données à propos des vols effectués aux États-Unis entre 1987 et 2009. La manipulation et le traitement des données se sont faits autour de la problématique suivante : les retards de vols. Deux aspects autour des retards de vols ont été explorés. Le premier aspect : les retards lors des périodes de la journée, semaine et année, a été traité avec Cassandra. Le second aspect : les retards et l'âge des avions, a été traité avec Spark.

2 Modélisation

Pour mieux comprendre les choix de modélisation qui ont été effectué, il est nécessaire d'avoir connaissance des questions exactes que j'ai choisi parmi les cinq proposées :

1. Quelle est la meilleur période de la journée / de la semaine / de l'année pour prendre l'avion et réduire les retards ?
2. Les avions plus vieux subissent-ils plus de retards ?

Modélisation relative à la question 1. Parmi les données disponibles se trouvaient tous les fichiers de vols par années au format `csv` : `XXXX.csv`. Après avoir explorer ces fichiers, j'ai pu sélectionner les données qui m'ont paru être les plus importantes pour traiter cette question. Les voici (en français) :

— Année, mois, jour du mois, jour de la semaine, heure de départ CRS
— Retard à l'arrivée, retard au départ
— Annulé
— Numéro de la queue de l'avion

Modélisation relative à la question 2. Les données importantes pour répondre à cette question sont très similaires à celles de la question 1. à la différence que certains champs deviennent inutiles : mois, jour du mois, jour de la semaine, horaire de départ CRS. De plus, une données devient alors nécessaire : l'âge de l'avion le jour où il effectue le vol. Pour trouver cette donnée, le champ année du fichier `plane-data.csv` a été utilisé. Pour croiser cette donnée avec les autres champs sélectionnés pour la question 1. le numéro de la queue a été utilisé.

Pour répondre à ces deux questions, le calcul des moyennes, des écarts type et proportion de vols annulés ont été nécessaires. Voici la modélisation finalement adoptée :

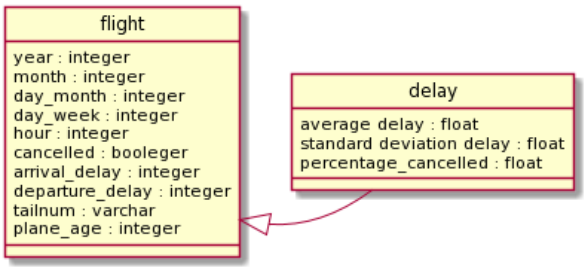

FIGURE 1 – Diagramme UML de la modélisation adoptée.

3 Acquisition des données

Cette étape consistait en la lecture des différents fichiers `csv` utilisés pour récupérer les données correspondant à la modélisation. Cette étape est commune aux deux questions et aux implémentations

avec Cassandra et Spark. Plusieurs données nécessitaient un pré-traitement avant d'être manipulées. En effet, l'heure de départ CRS était présente dans les fichiers `XXXX.csv` sous la forme d'un horaire de départ CRS au format `hh:mm`. Il a donc fallu ne sélectionner que l'heure de la chaîne de caractères. De plus, l'âge est aussi une donnée qui a nécessité un traitement. En effet, seule l'année du vol était donnée par les `XXXX.csv` et l'année de livraison de l'avion par le fichier `plane-data.csv`. Il a donc fallu calculer l'âge de l'avion effectuant chacun des vols présents dans les fichiers `XXXX.csv`.

La lecture des fichiers `XXXX.csv` est faite par une méthode qui *yield* les données à manipuler sous forme de `namedtuple` Python : `Flight`. Cette méthode est aussi chargée de calculer l'âge de l'avion et de l'intégrer dans les données *yieldées*. Les années de livraison des avions sont associées avec les vols par un dictionnaire Python ayant pour clés les numéros de queue des avions et pour valeur les années de livraison.

Cette étape n'effectue aucun nettoyage des données. Il sera effectué plus tard au moment de l'insertion des données. Un formatage est tout de même effectué sur les données manquantes : les données nulles ou vides sont remplacées par les caractères `NA`, ce qui permettre de simplifier le nettoyage.

4 Cassandra

Pour rappel, la question choisie était : quelle est la meilleur période de la journée / de la semaine / de l'année pour prendre l'avion et réduire les retards ?

4.1 Choix du stockage

Le choix le plus important a été de choisir la clé de partitionnement et la clé de tri. Pour choisir la clé de partitionnement, j'ai pris en compte le fait de pouvoir requêter les vols selon leur date. La clé de partitionnement est donc la suivante :

(année, mois, jour du mois, jour de la semaine, heure de la journée du vol)

La clé de tri devait pouvoir permettre de différencier tous les vols ayant une clé de partitionnement identique. Comme un avion de peut pas faire deux vols simultanément, la clé de tri choisie est la suivante :

(numéro de la queue de l'avion)

Les autres de champs stockés dans Cassandra sont les champs restants, mentionnés dans la partie 2.

4.2 Insertion des données

J'ai choisi de n'insérer dans Cassandra que les données dont l'année, le mois, le jour du mois, le jour de la semaine , l'heure et le statut d'annulation sont renseignés, c'est à dire différents de `NA`. Cela à l'avantage d'éviter une phase de nettoyage plus poussée lors de la réception ou l'analyse des données. Cela à aussi un inconvénient sur lequel je reviendrais dans la sous partie 4.6.

Pour insérer les données dans Cassandra, il faut prendre garde à ne pas tenter de stocker les données `NA` dans des colonnes dont le type n'est pas une chaîne de caractères. J'ai donc choisi d'insérer des *null* à la place des `NA`.

D'un point de vu architecture, j'ai choisi de gérer la connexion à Cassandra à l'aide d'une classe `ConnectionDB`. Les accès à la base de données se font tous à l'aide d'une classe qui hérite de la classe `ConnectionDB`. Les classes filles possèdent toutes une méthode qui permet d'être supprimer et de supprimer la connexion à la base en terminant la connexion au *cluster*. L'insertion des données se fait donc au sein d'un objet `InsertFlight` héritant de `ConnectionDB`.

4.3 Réception des données

Les données utiles pour traiter la question étaient les vols : d'une heure, d'un jour de la semaine et d'une période l'année (j'ai choisi les saisons). J'ai donc créé trois méthodes chargées de récupérer les données au grain souhaité. Ces méthodes nécessitant un accès à la base, elle appartiennent à un objet de la classe `GetFlight` héritant de la classe `ConnexionDB`. Ces trois méthodes font toutes appel à la même méthode qui requête directement la base : `get_flight_one_day_hour`. Cette méthode est la seule méthode à requêter la base. Elle requête les données identifiées par l'année, le jour du mois, le jour de la semaine, l'heure de la journée.

Pour ne pas requêter les données de toute la base, j'ai choisi de laisser l'utilisateur saisir des dates limites, entre lesquelles ils souhaitent faire les statistiques et les analyses. Ainsi, la méthode permettant de récupérer les vols associés à une heure de la journée consiste en une boucle sur les jours séparants les deux dates, en requêtant au sein de cette boucle les vols d'une journée à l'heure donnée. Pour toutes les heures de la journée, la méthode de requête est appelée 24 fois. Les autres méthodes, pour les jours de la semaine et les saisons, fonctionnent de la même manière. Cette stratégie pose, selon moi, des problèmes de performances. La sous partie 4.6 y reviendra.

Le requêtage des données devait aussi pouvoir permettre d'obtenir les vols annulés ou non, selon le choix de l'utilisateur. Une booléen a donc été utilisé. Cela a permis de traiter séparément les vols annulés et les vols maintenus.

4.4 Analyses des données

Pour répondre à la question, rappelée en début de partie 4, j'ai choisi d'utiliser les indicateurs suivants :

- La moyenne des retards à l'arrivée et au départ par heure, jour et saison.
- L'écart type des retards à l'arrivée et au départ par heure, jour et saison.
- La proportion des vols annulés par heure, jour et saison (en pourcentages).

Ces calculs reposaient sur deux méthodes :

- `_comp_mean_std_delay`. Cette méthode permet de calculer les moyennes et écart type, peu importe le grain, à l'aide d'un *map-reduce*.
- `_comp_cancelled_proportion`. Cette méthode permet de calculer les proportions de vols (en pourcentages), peu importe le grain, à l'aide d'un *map-reduce*.

Ces deux méthodes ont donc pu être réutilisées à tous les grains choisis.

4.5 Réponse à la question

Pour commencer, la figure 2 illustre les moyennes et écarts type des retards à l'arrivée et au départ par heure de la journée, obtenus sur les données d'une année (2007). On peut observer que les retards augmentent au fur et à mesure de la journée. Pour éviter les retards, il vaut donc mieux voyager le matin. Les moyennes affichées de 1h à 5h, ne sont pas très significatives car peu de données étaient disponibles sur ces heures là. Concernant les écarts type, on remarque que les données sont très dispersées autour de la moyenne, personne n'est donc à l'abri de tomber sur un vol très retardé ou au contraire très en avance. Enfin, on remarque que les retards pris au départ ne sont pas, ou peu, rattrapés pendant le vol en moyenne.

Ensuite, la figure 3 permet d'illustrer les retards moyens des vols à l'arrivée et au départ, en 2007, par jour de la semaine. On peut aussi observer, à droite, les proportions des vols annulés. On peut donc dire de manière assez confiante qu'il vaut mieux prendre l'avion le samedi si l'on veut éviter les retards, ou le mardi. De plus, le samedi semble être plus épargné par les annulations.

La figure 4 illustre les retards moyens des vols à l'arrivée et au départ, en 2007, par saison. On peut aussi observer, à droite, les proportions des vols annulés. On remarque que l'hiver est la saison

qui voit le plus de ses vols annulés (environ 1%), on peut imaginer que cela est lié à la météo. De plus, on observe que la saison qui subit les retards les plus importants est l'été. On peut imaginer que cela est lié aux vacances et à l'augmentation du trafic aérien.

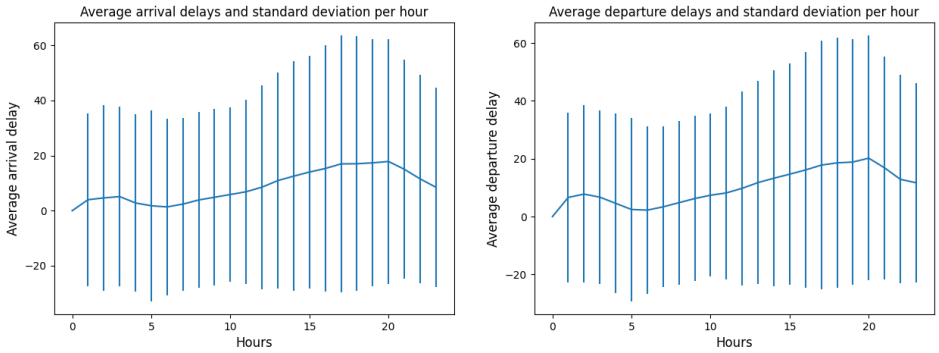

FIGURE 2 – Diagramme des retards moyens et écarts type des vols obtenus sur l'année 2007, heure par heure, à l'arrivée (à gauche) et au départ (à droite)

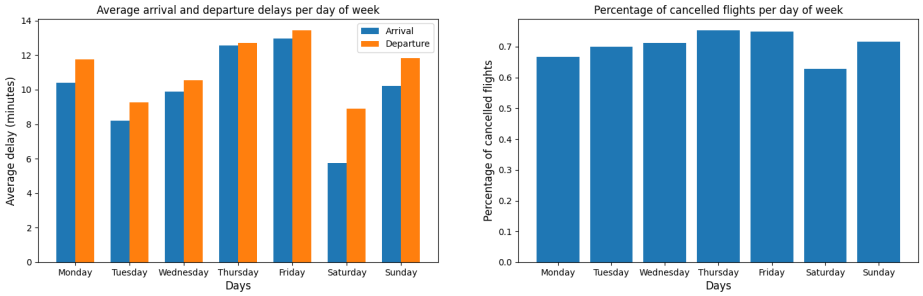

FIGURE 3 – Diagramme en barre des retards moyens des vols obtenus sur l'année 2007, jour par jour, ainsi que la proportion de vols annulés.

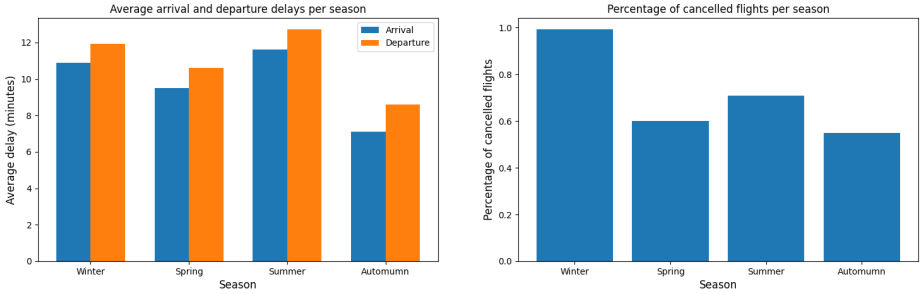

FIGURE 4 – Diagramme en barre des retards moyens des vols obtenus sur l'année 2007, par saison, ainsi que la proportion de vols annulés.

4.6 Critiques et améliorations

Comme je l'ai mentionné dans la sous partie 4.2, ne pas insérer les vols dont certaines données sont non conformes ou manquantes, ne permet pas de les quantifier. Si les quantifier était souhaité, il aurait été nécessaire de les insérer dans la base.

Un autre point critique est la stratégie adoptée pour récupérer les données associées aux différentes granularités. En effet, le stockage et l'implémentation choisis forcent à utiliser une requête minimale, du niveau de la granularité la plus faible : l'heure de la journée. Travailler avec des données de granularité

supérieur comme les saisons nécessitent donc de réaliser un grand nombre de requêtes, qui aurait pu être réduit si le stockage avait été différent. En effet, sept minutes d'attentes sont nécessaires pour calculer les statistiques des retards selon les heures de la journée, sur un an. Un stockage avec une date `timestamp` aurait permis de limiter le nombre de requêtes nécessaires pour récupérer les vols. Un nettoyage aurait du être fait ensuite pour ne *yielder* que les données associées à l'heure, jour ou saison souhaité.

Enfin, une dernière amélioration serait de pouvoir traiter les données de plusieurs années en les différenciant. En effet, dans ce qui a été fait pour l'instant, les années n'ont pas été prises en compte. Pour ce faire, il suffirait de créer une méthode où l'utilisateur saisirait des années, puis elle effectuerait les calculs pour chaque année.

5 Spark

Pour rappel, la question choisie était : les avions plus vieux subissent-ils plus de retards ?

5.1 Analyse des données

Pour répondre à la question susmentionnée, j'ai choisi d'utiliser les méthodes d'analyses suivantes :

1. Classer les différents vols effectués sur une année donnée dans des groupes d'âges d'avion : moins de 5 ans, entre 5 et 10 ans, entre 10 et 15 ans, entre 15 et 20 ans, entre 20 et 25 ans et plus que 25 ans. À l'issu de cette classification, calculer les retards moyens au départ et à l'arrivée, sur une année, ainsi que les écarts type.
2. Calculer la moyenne d'âge des avions ayant volé sur une année donnée, puis grouper les différents vols effectués sur la même année selon l'âge de l'avion : âge supérieur ou inférieur ou égal à la moyenne. Sur ces deux groupes ainsi créés, calculer la moyenne et l'écart type des retards à l'arrivée et au départ, sur la même année.

Les proportion de vols annulés n'a pas pu être calculé dans ce cas car la plus plupart des numéros de queue des n'étaient pas renseignés pour les vols annulés. Il était donc impossible de mettre en relation les avions avec leur âge, comme ces derniers n'étaient pas renseignés.

Avant tout calcul, une étape de nettoyage des données a été nécessaire, afin d'éliminer toutes les données contenant des `NA` qui auraient empêché de calculer les âges et de ne pas garder les vols annulés.

Ensuite, les deux méthodes ont nécessité le calcul des moyennes et écarts type par un *map-reduceByKey*. En voici le formalisme :

$map : flight \mapsto ((group(flight), year), [1, ArrDelay, DepDelay, ArrDelay^2, DepDelay^2])$

$reduceByKey : +_{\mathbb{R}^5}$ associatif et commutatif

$$map : ((group(flight), year), [\sum 1, \sum ArrDelay, \sum DepDelay, \sum ArrDelay^2, \sum DepDelay^2]) \mapsto$$

$$\left(\frac{\sum ArrDelay}{\sum 1}, \frac{\sum DepDelay}{\sum 1}, \sqrt{\frac{\sum ArrDelay^2}{\sum 1} \left(\frac{\sum ArrDelay}{\sum 1} \right)^2}, \sqrt{\frac{\sum DepDelay^2}{\sum 1} \left(\frac{\sum DepDelay}{\sum 1} \right)^2}, \sum 1 \right)$$

La première méthode nécessitait en plus le calcul de la moyenne d'âge de tous les avions ayant volé sur un an et sans redondance. Pour ce faire, deux *map-reduceByKey* ont exécutés à la suite. Le premier *map-reduceByKey* a été fait avec le numéro de queue et l'année du vol en guise de clé et a permis d'associer à chaque clé un unique âge. Le deuxième *map-reduceByKey* a permis de calculer les moyennes sur les années, en prenant les années comme clé.

5.2 Réponse à la question

La figure 5 illustre les retards moyens des vols à l'arrivée et au départ, en 2007, par catégorie d'âge des avions les ayant effectué. On peut supposer, en observant le diagramme, que les avions dont l'âge dépasse 25 ans semblent voler plus lentement. En effet, ils partent avec une avance moyenne inférieur aux autres catégories d'âge mais arrivent avec un retard nettement plus conséquent. Cette observation n'est pas surprenante car la durée de vie moyenne des avions est de 25 ans (source : `https://www.aeronewstv.com`).

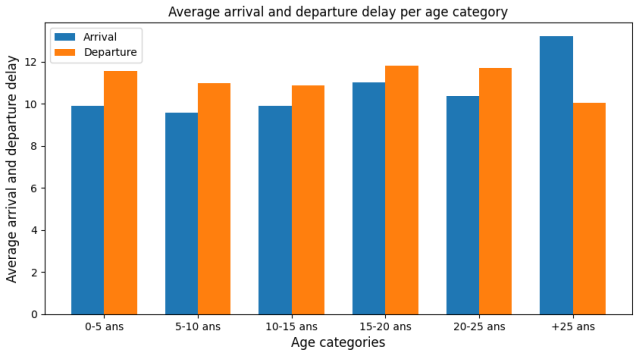

FIGURE 5 – Diagramme en barre des retards moyens des vols obtenus sur l'année 2007, par catégories d'âge des avions ayant volé.

J'ai choisi de ne pas présenter de figure concernant la deuxième méthode envisagée car elle n'apportait aucune information plus intéressante que celles fournies par la figure 5.

5.3 Critiques et améliorations

Une amélioration possible aurait été d'affiner les différentes catégories d'âge car la plus part des vols, en 2007, ont été effectués par des avions ayant entre 0 et 10 ans.

Je pense que la méthode d'observation des retards des avions plus ou moins vieux que la moyenne n'était pas une bonne méthode. En effet, la moyenne a eu pour effet de lisser les résultats et aucune différence visible n'était possible.

De plus, un lien significatif ou non aurait pu être mis en évidence entre l'âge et les retards en utilisant un coefficient de Pearson et en pratiquant un test d'indépendance par exemple.

Enfin, afficher les données sur plusieurs années auraient été intéressant mais n'a pas été implémenté, bien que les résultats des *map-reduceByKey* étaient fournis par année dans le but de ne pas mélanger les données des années.

6 Instructions d'exécution

Pour exécuter les analyses relatives à Cassandra et à Spark, il suffit de lancer un `%run analyse_cassandra.py` et un `%run analyse_spark.py` dans `ipython`. Il faut s'assurer que les fichiers `csv` sources soient bien situés dans le dossier à la racine `/project_data/`. Lancer ces deux codes crée les différentes figures utiles pour répondre aux questions.

7 Conclusion

En conclusion, la meilleure combinaison pour éviter les retards à l'arrivée est de prendre un avion de moins de 25 ans un samedi matin avant 10 h en automne.