Recherche & Développement autour de la problématique du Big Data

VIGNERON
Joris

Sommaire

1. Introduction
2. Présentation de l'entreprise
3. Sujet du stage
4. Etat de l'art
5. Découverte et expérimentation
6. Application
7. Conclusion

Introduction

- Choix de l'Observatoire astronomique

- 2 raisons ayant motivé ce choix

Présentation de l'entreprise

- Stage réalisé à l'Observatoire astronomique de Strasbourg

- Construit en 1881, possède la 3ème plus grande lunette de France par sa tailler

Observatoire astronomique de Strasbourg

- Il se doit de contribuer aux progrès de la connaissance

Présentation de l'entreprise – Les équipes de recherche

- Galaxies : réalise des recherches sur la formation des galaxies
- Hautes énergies : s'intéresse aux sources émettrice de rayon X
- Le CDS : développe des services d'accès aux données

CENTRE DE DONNÉES
ASTRONOMIQUES DE STRASBOURG

Présentation de l'entreprise - Les services du CDS

Base de données de référence pour la bibliographie des objets astronomiques

Plus de 40 ans de données, informations sur plus de 8 Millions d'objets

Base de données qui regroupe plus de 13 000 catalogues d'objets célestes. 300 000 requêtes par jour en moyenne.

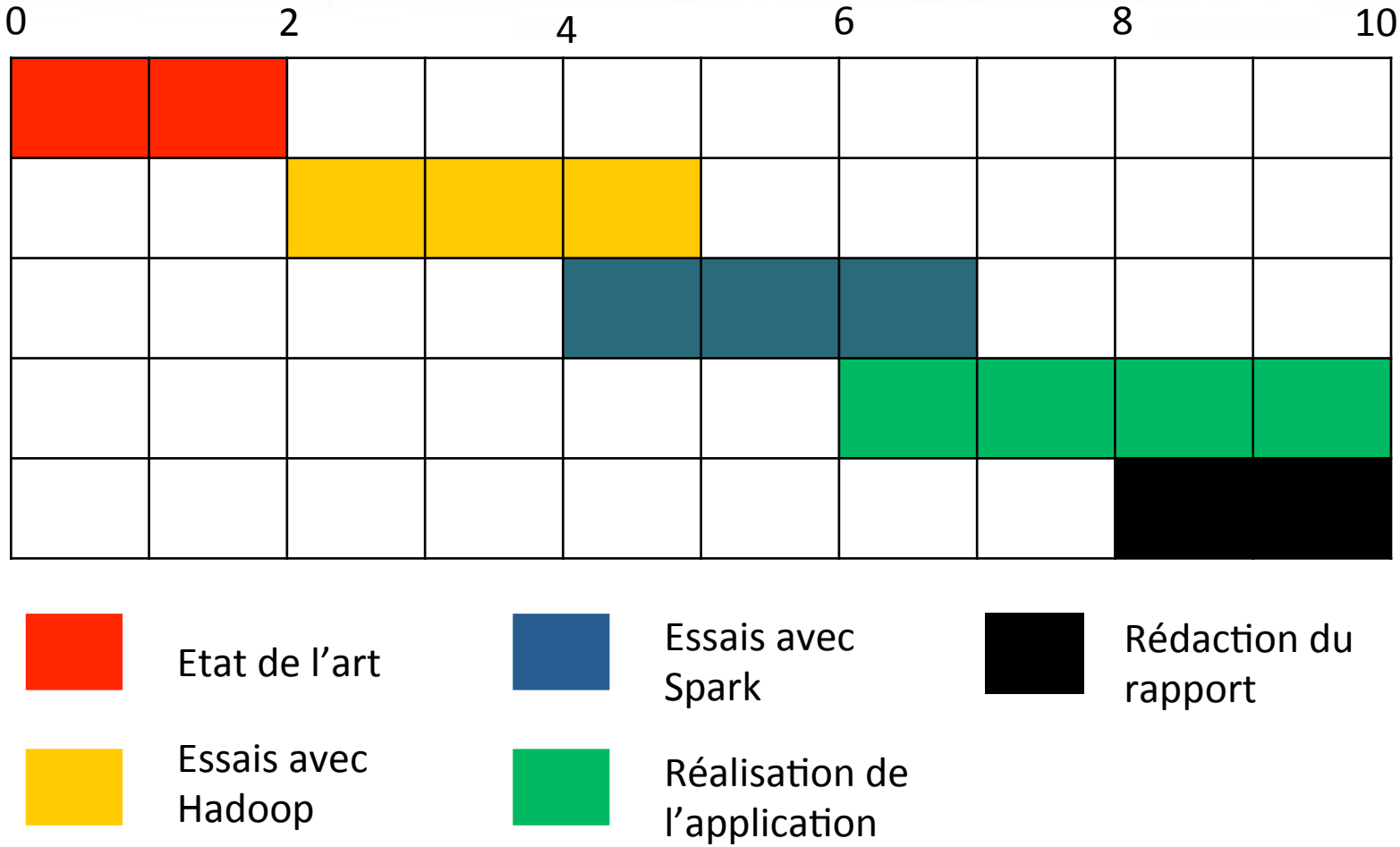
Atlas interactif du ciel permettant de visualiser des images astronomiques

Sujet du stage

- Problématique du Big Data et notamment les technologies qui gravitent autour

- Enjeux du stage :
  - Réalisation de diverses expérimentations dans le but d'éprouver les outils à disposition
  - Mise en œuvre d'un cas d'utilisation de l'Observatoire

Sujet du stage - Déroulement



Etat de l'art

- Début du stage : découvrir les technologies du Big Data

- Apprentissage des différentes facettes de cette technologie

Etat de l’art – Le Big Data

- Big Data ou « grosses données »
- L’humanité génère 2,5 trillions d’octets de données/ jours
- Les 3V : Volume, Variété, Vitesse
- Faiblesse des BD classiques

Etat de l'art - MapReduce

- Patron d'architecture de programmation
- Manipulation de grandes quantités de données
- Fonctionnement en 2 phases : phase Map et phase Reduce

Etat de l'art - Hadoop

- Framework Java Open Source développé par la fondation Apache

- Hadoop Distributed File System, HDFS est le système de fichier distribué d'Hadoop
- Basé sur le patron MapReduce
- Différentes distributions implémentant ce framework

Etat de l'art - Spark

- Framework développé par AMPLab de l'université de Berkeley
- Modèle de programmation plus simple et temps d'exécution 100 fois plus courts

- Le RDD, « Resilient Distributed Dataset », est une abstraction de collection sur laquelle est réalisée des opérations de manières distribuées
- Programmation différente d'Hadoop

Etat de l'art – Pig & Hive

- Infrastructure construite par-dessus Hadoop

- Utilise PigLatin, un langage de script

- Hive, permet de réaliser des requêtes en HiveQL, un langage proche du SQL

- Très bon outil pour développer de rapide job MapReduce

Découverte et expérimentation

- Objectif prendre en main le framework Hadoop
- Choix de la distribution HortonWorks
- Installation d'une distribution en local avec Eclipse

Hortonworks Sandbox with HDP 2.2

Leave Feedback

Component	Version	
Hue	2.6.1-2	
HDP	2.2.4	
Hadoop	2.6.0	
Pig	0.14.0	
Hive-HCatalog	0.14.0	
Oozie	4.1.0	
Ambari	2.0.1b1	Disable
HBase	0.98.4	
Knox	0.5.0	
Storm	0.9.3	
Falcon	0.6.0	
Sandbox Build	f14c3d8 09:23 03-04-15	

Découverte et expérimentation -Hive

- Simplification d'un job MapReduce
- Utilisation d'un langage HiveQL très proche du SQL

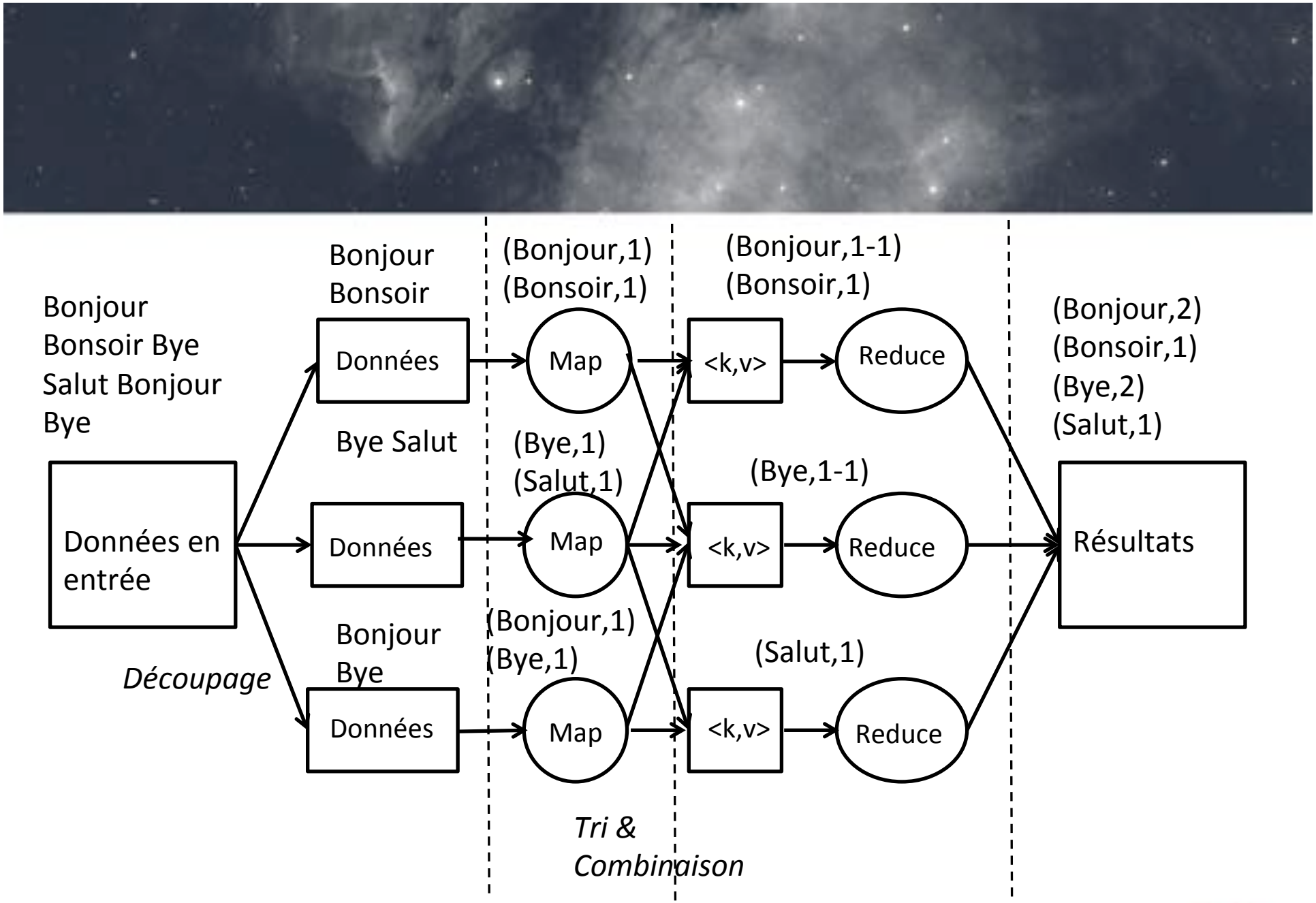
Bilan sur Hive et Pig

- Accessible pour des programmeurs non formés à Hadoop

- Inconvénients :
  - Presque inefficace pour des programmes trop complexes
  - Ne supporte pas tous les standards attendus

Découverte et expérimentation – Essais avec Hadoop

- Utilisation de Maven pour créer les projets
- Programmeur doit coder : une fonction *map()*, une fonction *reduce(),* et un driver
- Les données chargées sont découpées sous forme de bloc puis distribuées sur le cluster


Bonjour
Bonsoir Bye
Salut Bonjour
Bye
Données en entrée
Découpage
Bonjour
Bonsoir
Données
Bye Salut
Données
Bonjour
Bye
Données
(Bonjour,1)
(Bonsoir,1)
Map
(Bye,1)
(Salut,1)
Map
(Bonjour,1)
(Bye,1)
Map
Tri & Combinaison
(Bonjour,1-1)
(Bonsoir,1)
<k,v>
Reduce
(Bye,1-1)
<k,v>
Reduce
(Salut,1)
<k,v>
Reduce
(Bonjour,2)
(Bonsoir,1)
(Bye,2)
(Salut,1)
Résultats

Découverte et expérimentation – Essais avec Spark

- Chargement des données dans des javaRDD
- Utilisation de méthode emprunter au langage fonctionnel pour réaliser des traitements sur les JavaRDD
- Utilisation des classes Function, Function2,… permettant de définir les types d'entrées et de sorties

```
JavaPairRDD<String,String> cpiboPaire = cpiboInputFile.mapToPair(new PairFunction<String, String, String>(){
    public Tuple2<String,String> call(String s){
        //je découpe la ligne pour récupérer chacune des valeurs
        String[] t = s.split(",");
        //je stocke la valeur ra et de et je les convertit en double
        Double ra = Double.parseDouble(t[0]);
        Double de = Double.parseDouble(t[1]);
        //je calcule la valeur de l'id
        Long heald = nestIdx(ra,de);
        return new Tuple2<String,String>(heald.toString(),s);

    }
});
```

Application d'une fonction, dont le paramètre est une classe dont on peut définir les types d'entrées et de sorties, puis application sur chaque élément de notre JavaRDD d'un traitement correspondant à la fonction call contenu dans la classe interne.

Application

- Détermination d'une application à réaliser pour l'Observatoire

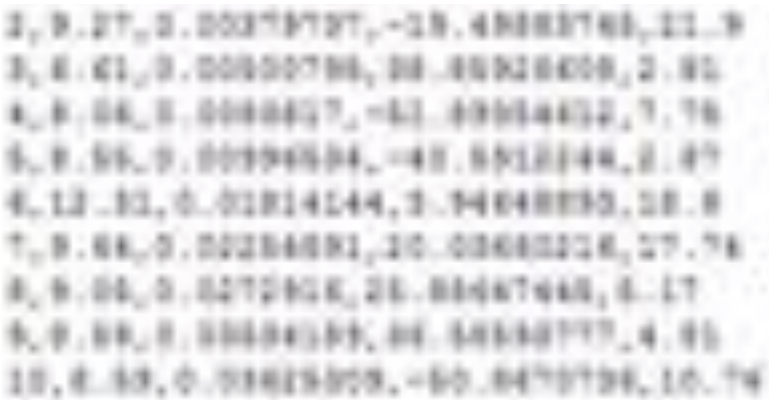
- Réalisation d'une application de CrossMatch

Application - CrossMatch

- Réalisation d'une jointure entre 2 catalogues
- Calcul d'un identifiant servant de clé de jointure
- Filtration des sources en calculant la distance entre elles

Application - Description des sources

- 2 Catalogues :
  - Tycho : 2 539 913 valeurs
  - HipMain : 119 218 valeurs



Application – Réalisation avec Hadoop

- Réalisation de recherches pour trouver une façon de faire une jointure

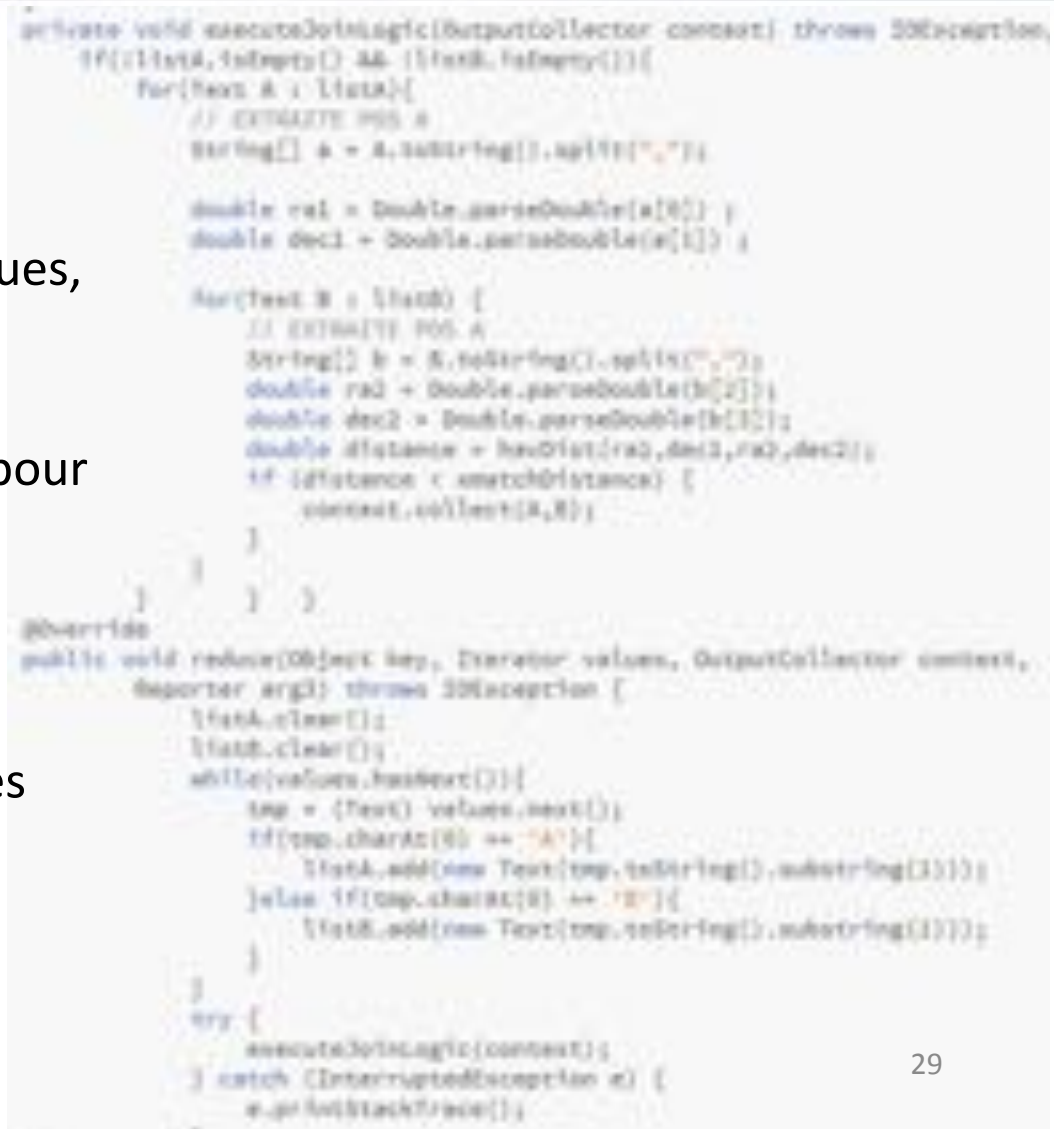
- Construction du code :
  - 2 Mappers, un par catalogue
  - 1 Reducer
  - 1 Driver

Application - HipMainJoinMapper

- Découpage de la ligne courante pour obtenir valeurs de ra et de
- Calcul de l'identifiant à partir de ces 2 valeurs
- Création du couple <clé,valeur> :
  - Clé -> l'identifiant
  - valeur -> la ligne courante + une lettre

Application - CrossMatchReducer

- En entrées : un couple <clé,Collection<valeur>>
- Ces valeurs viennent des 2 catalogues, identification par la lettre
- Parcours de la Collection<valeur> pour ranger chaque valeur dans la liste correspondante
- Réalisation de la jointure en parcourant les 2 listes, filtration des données avant écriture



Application - Résultat

- En entrée on a donc :
  - Tycho : 2,5 Millions valeurs
  - HipMain : 120 000 valeurs

- En sortie : 119 167 valeurs

Conclusion

- Une expérience enrichissante
- Difficultés rencontrées
- Conclusion sur Hadoop et Spark

Merci de m’avoir écouté, avez-vous des questions ?