Projet de session

Cluster Hadoop & Administration / MapReduce & Hive

Auteurs: Charles Brisson, Mario Nadon, Yadong Wang

AEC Spécialiste Intelligence d'affaires & Big Data - Automne 2015

Collège Bois-de-Boulogne

Introduction

Le client, M. Brisson, est un promoteur immobilier spécialisé dans la construction/rénovation de locaux afin de les rendre attrayants pour des commerçants à la recherche des meilleurs endroits pour s'installer.

Grâce à son approche « big data », le promoteur cible des endroits précis pour l'établissement, par exemple, de restaurants d'un type ou d'un autre (*fast food,* familial, exotique, etc.) - en utilisant les données récupérées dans d'autres villes, en établissant des modèles et prédictions.

L'approche est complexe et dépasse largement l'étendue du présent travail. Celui-ci demeure toutefois un élément important de la démarche.

Caractéristiques des données utilisées (Dataset)

Les données utilisées proviennent de YELP « The Challenge Dataset »

- **2.2 millions d'évaluations** et **591,000 commentaires** par **552,000 utilisateurs** pour **77,000 entreprises**
- **566,000 caractéristiques** d'entreprises, ex., heures d'ouverture, stationnement disponible, ambiance.
- Réseau social de **552,000** utilisateurs pour un total de **3.5 millions d'arrêtes de graphe** social.
- Enregistrements (check-in) agrégés à travers le temps pour chacune des **77,000** entreprises
- **200,000 images** des entreprises

Villes :

- **Angleterre:** Edinburgh
- **Allemagne:** Karlsruhe
- **Canada:** Montréal, Waterloo
- **États-Unis:** Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison

Source: https://www.yelp.com/dataset_challenge

Caractéristiques des données utilisées (Dataset)

Le dataset est composé de 6 tables: (les données que nous avons utilisées sont identifiées en bleu)

business

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM), ]
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}
```

tip

```
{
    'type': 'tip',
    'text': (tip text),
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'date': (date, formatted like '2012-03-14'),
    'likes': (count),
}
```

review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

user

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
```

photos (from the photos auxiliary file)

This file is formatted as a JSON list of objects.

```
[
    {
        "photo_id": (encrypted photo id),
        "business_id" : (encrypted business id),
        "caption" : (the photo caption, if any),
        "label" : (the category the photo belongs to, if any)
    },
    {...}
]
```

check-in

```
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it will not be in the dict
}
```

Note: Les autres données nous serviront assurément pour de futurs travaux de formation

Le besoin d'affaires

Classer les entreprises avec **MapReduce Java** :

- selon le **score**[1]
- selon **quartile**
- par **ville**
- par **catégorie**

Question à laquelle on pourra répondre avec HIVE :
*Quelle est le **nombre moyen d'évaluations** par **Ville** et **Nombre d'étoiles**?*

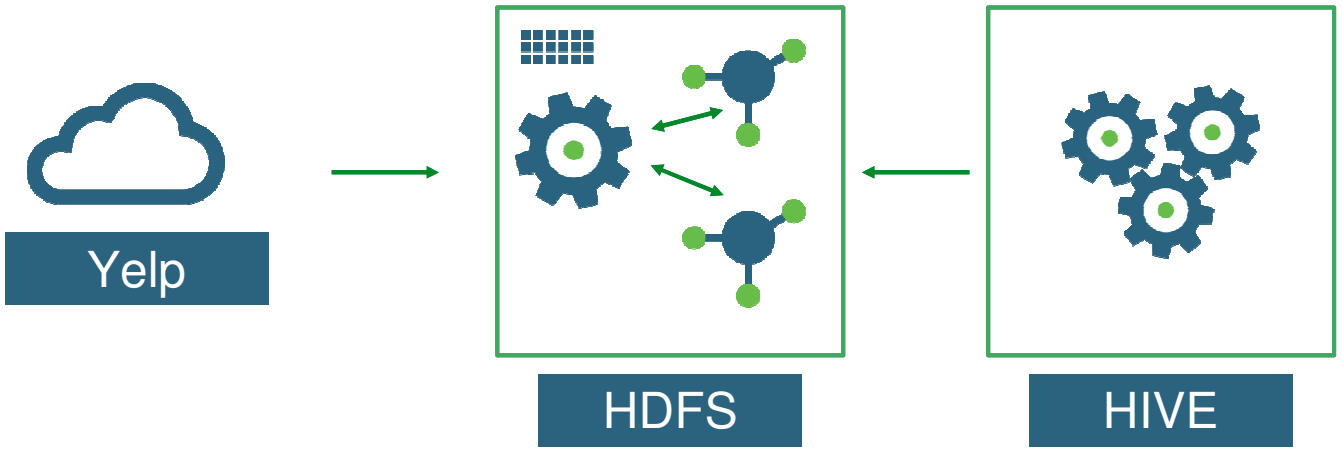
[1] : **Score:** Le score est une formule mise en place seulement dans un but pédagogique et ne représente aucunement une façon de bien classer les entreprises.
Formule utilisé pour le calcul du score = (([*stars*] * 10) * 1,000,000) + nombre d'évaluations [nb_*review*]
notes: * 10 pour éliminer les décimales
* 1,000,000 pour ne pas perdre le poids du champs [*stars*]

Caractéristiques du Cluster Hadoop

Système

Caractéristiques	Namenode	Datanode #1	Datanode #2
Mémoire	9 Gb ram	2.5 Gb ram	2.5 Gb ram
OS Linux	CentOS 6.4	CentOS 6.4	CentOS 6.4
Outils	cloudera manager 5.4.3		

Diagramme



Développement Hive

Question

Quelle est le nombre moyen d'évaluations par Ville et nombre d'étoiles pour la Ville de Montréal (par exemple)?

Créer la table

```
Create table yelpBusiness (json STRING);
```

Charger les données du dataset

```
Load data inpath '/user/root/testtp/inputyelp/business.json'
into   table yelpBusiness;
```

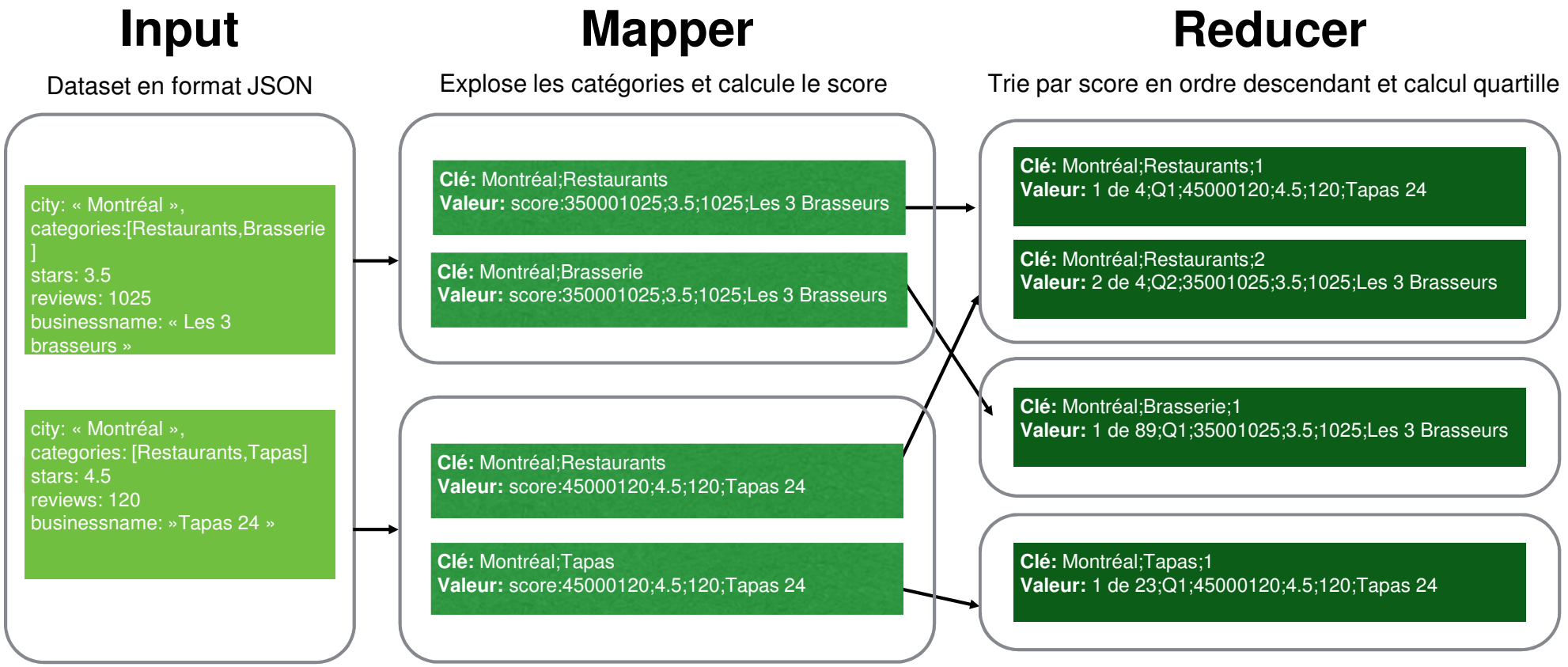
Requête pour obtenir le nombre moyen d'évaluations

```
Select      get_json_object(yelpBusiness.json,'$.city')                    as city,
            get_json_object(yelpBusiness.json,'$.stars')                   as stars,
            avg(get_json_object(yelpBusiness.json,'$.review_count')) as nbreviews
From        yelpBusiness
Where       get_json_object(yelpBusiness.json,'$.city') = 'Montréal'
Group by get_json_object(yelpBusiness.json,'$.city'),
            get_json_object(yelpBusiness.json,'$.stars')
Order by  city, stars;
```

Développement Map/Reduce (java)

Besoin

Pour chaque ville / catégorie nous voulons classer les entreprises en ordre de meilleures nombres d'étoiles et du plus grand nombre d'évaluations et leur déterminer dans quel Quartille elles se trouvent.



Développement MapReduce (java)

Code Mapper:

```
public static class YelpRankMapper extends Mapper<Object, Text, Text, Text>{

  public static final String  fieldSep       = ";";
  public static final char    stringDelim  = '"';
  public static final String  newLine      = "\n";
  public static final String  comma        = ",";
  private Text keyText    = new Text();
  private Text valueText = new Text();

public void map(Object key, Text value, Context context)  throws IOException, InterruptedException {

// Lire les informations qui sont en format JSON
  JsonReader reader = Json.createReader(new StringReader(value.toString()));
  JsonObject yelpBusiness = reader.readObject();
  reader.close();

// Récupérer le VILLE
  String ville = yelpBusiness.getString("city");
  if (!ville.isEmpty()) {
   // Récupérer les informations de la Business
    double nbStars                  = yelpBusiness.getJsonNumber("stars").doubleValue();
    int nbReviews                   = yelpBusiness.getInt("review_count");
    String businessName             = yelpBusiness.getString("name") .replace(newLine, comma);
    String businessAdresse          = yelpBusiness.getString("full_address") .replace(newLine, comma);
    String state                    = yelpBusiness.getString("state") .replace(newLine, comma);
    JsonNumber latitude             = yelpBusiness.getJsonNumber("latitude");
    JsonNumber longitude            = yelpBusiness.getJsonNumber("longitude");

   // Calculer le score
    int score = ((int)(nbStars*10) * 1000000) + nbReviews;

   // Bâtir la VALEUR du Tuple
    valueText.set(Integer.toString(score)                                  + fieldSep +
                  nbStars                                                  + fieldSep +
                  String.valueOf(nbReviews)                                + fieldSep +
                  stringDelim + businessName     + stringDelim             + fieldSep +
                  stringDelim + businessAdresse + stringDelim              + fieldSep +
                  stringDelim + state                 + stringDelim        + fieldSep +
                  latitude.toString()                                      + fieldSep +
                  longitude.toString());

   // Récupérer les catégories de la business et traiter toutes les catégories
        JsonArray categoriesArray = yelpBusiness.getJsonArray("categories");
        for (JsonValue categorie : categoriesArray)
        { keyText.set(stringDelim + ville                          + stringDelim + fieldSep
                                  + categorie.toString());
          context.write(keyText, valueText);
       }
  }}}
```

Développement MapReduce (java)

Code Reducer:

```
public static class YelpRankReducer extends Reducer<Text,Text,Text, Text>

{   private Text newKey   = new Text();
    private Text newValue = new Text();

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

 // Récupérer les VALEURS (Iterable) et les ajouter dans une structure List (triable)
          List<String> businessList = new ArrayList<String>();
          for (Text business : values) { businessList.add(business.toString()); }

 // Trier la liste des VALEURS par SCORE en ordre descendant
          Collections.sort(businessList, new Comparator<String>() {
                    public int compare(String s1, String s2)
                    { StringTokenizer tokenS1 = new StringTokenizer(s1,";");
                      StringTokenizer tokenS2 = new StringTokenizer(s2,";");
                      int scoreS1 = Integer.parseInt(tokenS1.nextToken());
                      int scoreS2 = Integer.parseInt(tokenS2.nextToken());
                      return new Integer(scoreS2).compareTo(new Integer(scoreS1));
                    }});

 // Caluler les limites des Quartiles
          int nbBusiness =  businessList.size();
          int q1 = (int) Math.round(nbBusiness * 25 / 100);
          int q2 = (int) Math.round(nbBusiness * 50 / 100);
          int q3 = (int) Math.round(nbBusiness * 75 / 100);
          String quartille;

 // Pour toutes les business de la CLÉ (VILLE / CATÉGORIE)
          for (int rang = 0; rang < nbBusiness; rang++)
          { // Déterminer le quartile
              if       (rang <= q1)   {quartile = "Q1";}
              else if (rang <= q2)   {quartile = "Q2";}
              else if (rang <= q3)   {quartile = "Q3";}
              else                          {quartile = "Q4";};

            // Nouvelle CLE et VALEUR
              newKey.set (key.toString() + fieldSep + (rang + 1));
              newValue.set (fieldSep + stringDelim  + (rang + 1) + " de " + nbBusiness + stringDelim +
                                fieldSep + stringDelim  + quartille                                + stringDelim +
                                fieldSep + businessList.get(rang));

             context.write(newKey , newValue);

        }
}}
```

Difficultés rencontrées

Installation du cluster

- Communication entre les machines:
  - Cartes réseaux (static, dhcp, nat, mac address)
  - SSH (Keygen)
  - Adresse IP, IPV6 (selinux)
  - Configuration dans les fichiers pour namenode, datanode, host, hostname, localhost
  - Network time protocol (NTP)
- Manque de RAM de la machine virtuelle utilisée

MapReduce

- Traiter le dataset sous format JSON
- Trier les données par score en ordre descendant
- Se rendre compte qu'il ne faut pas de Combiner pour la logique désiré

Hive

- Traiter le dataset sous format JSON
- Permission pour écriture dans les répertoires

Compétences acquises

Techniques

- Certaine habitude à installer des *clusters* Cloudera Manager (l'équipe a monté plus de 50 clusters en un mois)
- Déploiement réussi de taches MapReduce en Java
- Utilisation réussie de HIVE pour obtenir d'autres résultats

Conceptuelles et affaires

- Compréhension de l'ampleur du travail pour l'installation et le déploiement d'une solution *big data*

Conclusion

Développeur

- Prêt pour le prochain projet!

Client

- Prêt à confier la prochaine étape à l'équipe!