EPITA – Ing 2
Big Data Analytics – BDA

TP 3 : Map-Reduce (Part 2)

idir.benouaret@epita.fr

Consignes

- **Travail à réaliser en groupe de 2 ou 3 étudiants**
- Pour chaque question/exercice, passez d'abord du temps pour concevoir votre solution avant de vous lancer dans l'implémentation.
- Prenez l'habitude de lancer/tester vos jobs d'abord sur un sous ensemble du dataset (`head -n nb dataset > dataset-small`) puis sur le dataset original (une fois qu'on est sur que notre code marche "bien")
- Certains datasets sont volontairement volumineux, pour tester le passage à l'échelle de votre solution (*sur un tout petit jeu de données l'impact des conneries qu'on peut faire n'est pas toujours visible*)
- Il faut produire une archive zip contenant :
  1. votre code commenté, de préférence un fichier par exercice.
  2. un rapport **pdf** qui doit contenir vos explications, description de votre solution, copies d'écran de vos résultats, et tout ce qui vous semble utile
- **Deadline : 8 Mars 23h42**

Exercice I : **Données textuelles**

Téléchargez le jeu de données gutenberg, ici. Ces données sont issues du projet Gutenberg (voir `https://www.gutenberg.org/`). Il s'agit d'une libraire en ligne regroupant plusieurs milliers de e-livres gratuits. Assurez vous de le mettre sur votre répertoire HDFS.

Q1 – Reprenez votre programme qui calcule la fréquence d'apparition de chaque mot dans le texte.

Q2 – Exécutez le en mode local, puis sur hadoop avec l'option -r hadoop : python3 wordcount.py -r hadoop hdfs :///user/root/data/fichier.txt

Activité du Cluster

Cette partie est à lire attentivement après avoir lancé une application MapReduce (un job) sur Yarn. Une fois que votre programme est lancé, il faut pouvoir examiner les informations retournées par Hadoop. Il y a deux types de compte-rendus à analyser : les affichages lors du lancement du job MapReduce et les logs Yarn.

Juste après le lancement de votre job, yarn affiche beaucoup de lignes.

- Repérez les lignes suivantes :

```
Total input files to process : 1
number of splits:8
```

La première ligne indique le nombre total de fichiers à traiter, ici un seul. La deuxième ligne le nombre de splits qui ont été effectués, il s'agit souvent du nombre de blocs (partitions) de votre fichier, dans cet exemple c'est 8.

- La ligne suivante indique l'identifiant de l'application

```
Submitted application application_1676814375656_0007
```

Cette identifiant sert à retrouver les détails (logs), l'état (running, finished, etc), la progression de votre job.

- `Job job_1676814375656_0007 running in uber mode : false`

Le mode 'uber' (True/False) : c'est une optimisation faite par yarn quand les données sont de petite taille. True quand c'est le cas, false sinon.

- Ensuite, vous aurez un état d'avancement des tâches map et reduce

Par exemple :

```
map 0% reduce 0%
map 10% reduce 0%
map 15% reduce 0%
map 20% reduce 0%
map 25% reduce 0%
map 30% reduce 0%
map 35% reduce 0%
map 40% reduce 0%
map 45% reduce 0%
map 54% reduce 0%
map 58% reduce 0%
map 70% reduce 0%
map 71% reduce 0%
map 75% reduce 0%
```

```
map 77% reduce 0%
map 80% reduce 0%
map 81% reduce 25%
map 82% reduce 25%
map 84% reduce 25%
map 85% reduce 25%
map 86% reduce 25%
map 88% reduce 25%
map 89% reduce 25%
map 90% reduce 25%
map 93% reduce 25%
map 95% reduce 25%
map 100% reduce 25%
map 100% reduce 46%
map 100% reduce 67%
map 100% reduce 78%
map 100% reduce 89%
map 100% reduce 100%
```

Plus votre job est "costaud" plus il y aura de lignes.

- La prochaine ligne indique si votre job a réussi ou non

```
Job job_1676814375656_0007 completed successfully
```

- À la fin du job, vous allez voir des informations très importantes : le nombre de paires (clé,valeur) en entrée et en sortie du map

```
Map input records=21228574
Map output records=63685722
```

Par exemple, ici, il y a eu 21228574 lignes traitées et 63685722 paires à destination du reducer. Quand Map output records vaut zéro, c'est sûrement que votre mapper ne parvient pas à travailler du tout ; vous aurez à analyser les logs sur Yarn, voir plus bas

- Il y a ensuite le nombre de paires (clé, valeur) traitées par le combiner, ici aucune car pas de combiner

```
Combine input records=0 Combine output records=0
```

Ici, nous n'avons pas défini de combiner donc cela affiche simplement 0. Essayez, de modifier votre programme pour ajouter un combiner et de voir la différence. Pour rappel, un combiner joue le rôle d'un mini reducer. Son job est de faciliter le travail du reducer en faisant un pré-calcul et en envoyant le résultat du pré-calcul aux reducers. Par exemple, sur le Wordcount un combiner peut effectuer la tache de calculer le nombre d'apparition de chaque mot de manière locale (sur une machine) et d'envoyer au reducer des résultats préliminaires.

- Pour finir, il y a le nombre de clés différentes ainsi que les paires traitées par le reducer :

```
Reduce input groups=3 Reduce input records=63685722 Reduce output records=3
```

Ici, il y a trois clés distinctes parmi les (clé, valeur) produites par les mappers, 63685722 paires en tout en entrée et seulement 3 en sortie (après le reduce).

Interface Web

Lorsque votre job a fini, qu'il ait marché ou non, vous devez pouvoir voir les résultats. Pour le cas d'un job qui termine sans succès, il faut pouvoir trouver la cause en analysant les logs pour regarder ou ça a foiré (NullPointerException, ClassCastException, NumberFormatException...)

Ouvrez votre navigateur, puis, allez sur l'url `http://localhost:8088`. C'est la page d'accueil du cluster.

1. Ouvrez le premier lien de la page d'accueil : applications . Cette page affiche toutes les applications MapReduce lancées sur YARN. Vous avez des informations sur l'activité de cluster : le nombre de jobs soumis, l'allocation de la mémoire, CPU...
2. Repérez l'une de vos derniers jobs que vous avez lancé à l'aide de son identifiant. Analysez, son état (state) (finished pour job terminé, et running pour les jobs en cours. FinalStatus (SUCCEEDED ou FAILED)
3. Cliquez sur son lien. La nouvelle page affiche les informations de votre application. En bas, vous verrez les tentatives (attempts) et sur quel(s) esclave(s) elle a tourné
4. Vous pouvez cliquer sur le lien Logs tout en bas à droite pour voir les différents logs qui sont émis.

Q3 – Adapter le programme comptant le nombre d'apparition de chaque mot en ajoutant un `Combiner`. Exécuter le programme et analyser/expliquer l'utilité d'ajouter un `Combiner`.

Q4 – Combien de mots commencent par chacune des lettres de l'alphabet.

Q5 – Cherchez tous les anagrammes du document. La question est volontairement vague. Faites toutes les suppositions et optimisations nécessaires. Par exemple : ne pas prendre en compte les mots contenant une seule lettre, ne pas prendre en compte les caractères de ponctuation. Un mot est forcément anagramme de lui même. Mais dans l'idéal extraire tous les mots s'écrivant avec les mêmes lettres mais dans un ordre différent. Par exemple : angered et enraged, break et brake, etc. *l'utilisation du package re peut être très utile.*

Q6 – Même question que la précédente, mais on veut maintenant garder que les palindromes (mots se lisant de la même façon dans les deux sens)

Exercice II : **Parcmètres (Paris)**

Le jeu de données suivant concerne les transactions sur les différents parcmètre à Paris. Téléchargez le en cliquant ici. Le fichier fait 24 448 029 enregistrements. Chaque ligne du dataset correspond aux champs explicités en table 1

TABLE 1: Dataset parcmètres

Champ	Exemple	Signification
0	4 110 201	Identifiant du parc-mètre
1	9 août 2014 15 :05	Date de la transaction
2	Rotatif	Type d'usager
3	CB	moyen de paiment
4	7,2	Montant en Euros
5	2	Nombres d'heures payées
6	9 août 2014 15 :05	Début du stationnement
7	9 août 2014 17 :05	Fin du stationnement

Ecrire les programmes MapReduce pour répondre aux questions suivantes :

Q1 – Le montant total des paiements par couple (année, mois)

Q2 – Le prix moyen par heure de stationnement

Q3 – La durée moyenne de stationnement par type d'usager

Q4 – La proportion de paiements avec « Paris Carte » par rapport aux paiements « CB »

Q5 – Les 10 parcmètres ayant généré le plus d'argent et les 10 ayant généré le moins d'argent

Q6 – Le parcmètre ou l'on stationne le moins, i.e., le moins utilisé en terme de temps de stationnement

EXERCICE III : **Données météo NCDC**

Vous venez d'être recruté par la société NCDC en tant que data engineer. Le but est de pouvoir les aider à analyser leurs données concernant la météo.

Nous allons commencer à travailler avec le fichier *isd-history.txt*. Il vient de cette URL `https://www1.ncdc.noaa.gov/pub/data/noaa/isd-history.txt`. C'est un fichier compact qui contient comme dans un fichier csv des champs, mais sans séparateur. Les champs sont de taille fixe. Mettez le sur votre répertoire HDFS puis Tapez la commande `hdfs dfs -tail data/isd-history.txt` pour voir à quoi peuvent ressembler les dernières lignes. La description de tous les champs est donnée en table 2

Q1 – Écrire un programme MapReduce qui calcule le nombres de stations dans l'hémisphère nord ($latitude \geq 0$) et dans l'hémisphère sud ($latitude \leq 0$). Il faut faire attention aux données manquantes.

Q2 – Quelle est la station Française (code pays FR) qui se trouve le plus au sud ? le plus au nord ? qui a la plus grande altitude ? Afficher à chaque fois le nom de la station.

Q3 – Pour chaque pays, afficher la station pour laquelle l'écart entre la date de début et de fin

Table 2: Descriptif des champs

Offset	taille	exemple	Signification
0	6	070930	USAF = Air Force Datsav3 station number
7	5	99999	WBAN = NCDC WBAN number
13	29	METZ NANCY LORRAINE	Station name
43	2	FR	FIPS country ID (pays)
48	2		ST : state of US stations
51	4	LFJL	CALL = ICAO call sign (code aéroport)
57	7	+48.982	LAT = Latitude in decimal degrees
65	8	+006.251	LON = Longitude in decimal degrees
74	7	+0265.2	ELEV = Elevation in meters (altitude)
82	8	19920408	BEGIN = Beginning Period Of Record (YYYYMMDD)
91	8	20230216	END = Ending Period Of Record (YYYYMMDD)

d'enregistrement est à la plus grande, si le résultat n'est pas unique afficher le nom de station dont la longueur est la plus grande.

Maintenant, on va travailler avec les données des relevés météos. Téléchargez le fichier *ncdc-2013-sorted.csv.gz* qui se trouve ici

Il s'agit des relevés météo sur toute l'année 2013. Chaque ligne du jeu de données contient :

- Le code de la station
- la date, en format ISO-8601
- Le type de valeur enregistrée. toutes les valeurs sont entières. Tmin(resp Tmax) correspond à la température minimale (resp. maximale) enregistrée.
- AWND désigne (average wind Speed)
- PRCP pour les précipitations
- D'autres type de données (TOBS, SNOW, etc)
- Le champs suivant contient la valeur correspondante (température, vitesse du vent, précipitations, etc)
- Chaque ligne contient 4 autres champs que nous n'utiliserons pas

La compagnie souhaite connaître la différence entre la température maximale et minimale à la station de Central Park pour chaque jour de 2013. La station correspondante à Central Park est celle avec le code USW00094728. Si nous jetons un coup d'œil aux valeurs TMIN et TMAX enregistrées pour cette station, ça ressemble à ça :

```
USW00094728,20130101,TMAX,44,,,X,2400
USW00094728,20130101,TMIN,-33,,,X,2400
USW00094728,20130102,TMAX,6,,,X,2400
USW00094728,20130102,TMIN,-56,,,X,2400
USW00094728,20130103,TMAX,0,,,X,2400
USW00094728,20130103,TMIN,-44,,,X,2400
```

Afin de permettre cette analyse, vous devez générer un fichier csv avec deux colonnes. Une colonne pour le jour (eg., 20130103) et la deuxième colonne doit correspondre à la variation de la température en degrés Celsius.

Q4 – Un stagiaire de votre équipe propose d'implémenter un job Mapreduce qui fait ceci :

- l'opération Map récupère des paires (tmin,tmax), les températures sont d'abord converties en degrés Celsius (division par 10)
- Pour chacune des paires, le reducer fait la soustraction entre la valeur maximale et la valeur minimale et afficher les résultats

L'idée à l'air pas mal, mais cela ne peut pas marcher. Quel est le problème ?

Q5 – Proposer un programme mapreduce qui ne prend en compte que les lignes qui nous intéressent. Il faut calculer la variation en température pour chaque jour. Quelle sont les clés et les valeurs que votre mapper doit sortir ? Quels sont leurs types ?

Q6 – Est ce que le mapper peut produire une paire (clé,valeur) depuis une seule ligne du dataset ? Comment résoudre ce problème ? Quelle serait la tâche de votre reducer ?

Q7 – Exécuter votre programme, d'abord en local puis sur hadoop

Exercice IV : **Amis et baseball**

Ici, nous allons travailler sur le fichier baseball_friends. Chaque ligne du jeu de données contient les infos suivantes :

- Le nom d'une personne
- L'équipe de Baseball dont il est fan (une seule)
- Suivi d'une liste des ses amis (liste de prénoms séparée par des virgules)

la première ligne par exemple dit que Aaden est fan des Red Sox et a 65 amis (Vérifiez cela avec un script). On suppose que tous les noms sont uniques et que la relation d'amitié est symétrique (supposition qui n'est pas forcément vraie sur le jeu de données)

Q1 – Écrire un programme map-reduce qui liste pour chaque personne son équipe favorite, ainsi que tous les amis qui supportent la même équipe favorite.