

Bases de données NOSQL et Big data

TP3 : Hadoop MAP/REDUCE

Diplôme National d'Ingénieur en Informatique

Spécialité :

Génie Logiciel

Réalisée par :

Oussama Ben Slama

Année Universitaire 2024/2025

Chapitre 1

Hadoop MapReduce

Nous avons parlé la semaine dernière de l'utilité de Hadoop MapReduce, un framework permettant d'écrire facilement des applications pour traiter d'énormes quantités de données. En effet, un programme MapReduce est composé d'une procédure de **map**, qui effectue le filtrage et le tri, et d'une méthode **reduce**, qui réalise une opération de synthèse.

1.1 Anagrammes

Dans cet exercice, nous allons simuler le mécanisme de Hadoop MapReduce pour déterminer les mots anagrammes. On commence par écrire le script **mapper** qui prend un fichier texte de l'entrée standard et trie chaque mot alphabétiquement.

```

mapper.py > ...
1
2  import sys
3
4  for line in sys.stdin:
5      word = line.strip()
6      sorted_word = ''.join(sorted(word))
7      print(f"{sorted_word}\t{word}")
8

```

FIGURE 1.1 – mapper.py

Ensuite, nous implémentons le *reducer*, qui calcule l'occurrence des mots anagrammes.

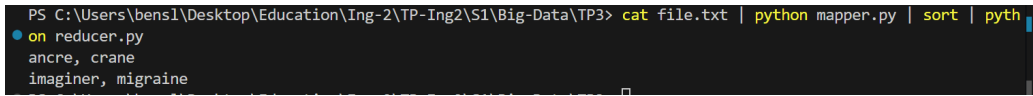
```

reducer.py > ...
1  import sys
2  from collections import defaultdict
3
4  anagrams = defaultdict(list)
5
6  for line in sys.stdin:
7      sorted_word, word = line.strip().split("\t")
8      anagrams[sorted_word].append(word)
9
10 for sorted_word, words in anagrams.items():
11     if len(words) > 1:
12         print(f'{', '.join(words)}")
13

```

FIGURE 1.2 – reducer.py

L'exécution de ces scripts pour un exemple donné produit le résultat suivant :




```
PS C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP3> cat file.txt | python mapper.py | sort | python reducer.py
ancre, crane
imaginer, migraine
```

FIGURE 1.3 – Résultat

1.2 Analyse des sentiments des clients sur Twitter

Dans ce travail, nous implémentons un script **mapper** qui, pour chaque tweet, calcule l'occurrence de certains mots, puis détermine si le tweet est classé comme **satisfait** ou **insatisfait**. Ensuite, un script **reducer** calcule le nombre de tweets dans chaque classe.



```
mapper_tweet.py > ...
1
2 import sys
3
4 bad_words = ["nul" , "insatisfait" , "bof" ,"incompétents"]
5 good_words = ["super" , "satisfait" , "excellent","merci"]
6 for line in sys.stdin:
7     bad = 0
8     good = 0
9     words = line.strip().split(" ")
10    for word in words :
11        if word in bad_words:
12            bad += 1
13        if word in good_words:
14            good += 1
15    if bad == good :
16        print(f"inconcluant"\t{1}")
17    elif bad > good :
18        print(f"insatisfait"\t{1}")
19    else :
20        print(f"satisfait"\t{1}")
21
```

FIGURE 1.4 – *mapper_{tweet}.py*

```

reducer_tweet.py > ...
1  import sys
2  from collections import defaultdict
3
4  occ = defaultdict(int)
5
6  for line in sys.stdin:
7      tweet, _ = line.strip().split("\t")
8      occ[tweet] += 1
9
10 for tweet, count in occ.items():
11     print(f"{tweet}\t{count}")

```

FIGURE 1.5 – `reducer_tweet.py`

```

PS C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP3> cat file1.txt | python mapper_tweet.py | sort | python reducer_tweet.py
inconcluant      1
insatisfait      2
satisfait        2
PS C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP3> 

```

FIGURE 1.6 – Résultat tweet

1.3 Index web inversé

Dans cet exercice, le but est de déterminer pour chaque page web les autres pages qui font référence à elle. Voici les scripts :

```

mapper_web.py > ...
1  import sys
2
3  for line in sys.stdin:
4      line = line.strip()
5      if not line:
6          continue
7      pi, references = line.split(" : ")
8      pi = pi.strip()
9      references = references.split(", ")
10     for pj in references:
11         print(f"{pj.strip()}\t{pi}")
12

```

FIGURE 1.7 – `mapperweb.py`

```

reducer_web.py ×
reducer_web.py > ...
1  import sys
2  from collections import defaultdict
3
4  pj_references = defaultdict(list)
5  for line in sys.stdin:
6      pj , pi = line.split('\t')
7      pj_references[pj].append(pi)
8
9  for pj, pis in pj_references.items():
10     print(f"{pj}:{', '.join(pis)}")
11

```

FIGURE 1.8 – `reducerweb.py`

```
PS C:\Users\bens1\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP3> cat file3.txt | python mapper_web.py | sort | python reducer_web.py
P1:P2
, P3
, P4

P2:P1
, P4

P3:P1
, P4

P4:P2
, P3

PS C:\Users\bens1\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP3> 
```

FIGURE 1.9 – Résultat web