Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА №3

«Цепочки MapReduce задач. Сравнение документов»

ДИСЦИПЛИНА: «Технологии обработки больших данных»

Выполнил: студент гр. ИУК4-72Б _____ (Карельский М.К.)
(Подпись)

Проверил: _____ (Голубева С.Е.)
(Подпись)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель: формирование практических навыков использования цепочек MapReduce для решения сложных задач обработки больших данных.

Задачи:

1. Получить навыки реализации цепочки MapReduce задач.
2. Изучить интерфейс Hadoop MapReduce.
3. Изучить алгоритмы анализа, сравнение текстовых документов
4. Получить практические навыки обработки и анализа текстовых данных

Задание:

Выполнить задание с помощью цепочки MapReduce задач согласно варианту. В качестве входных текстовых файлов можно использовать книги в txt формате из библиотеки Project Gutenberg: https://www.gutenberg.org.

Вариант 7

Подсчитать средний рейтинг фильма. Входной файл рейтингов имеет формат:

userId, movieId,rating,timestamp

Выполнить операцию объединения с файлом, содержащим названия фильмов. Данный файл имеет формат:

movieId, title, genres

Результат должен быть сохранен в файле в формате:

movieId, title, av_rating

Объединение выполнять согласно подходу Job-side join. В результате должны быть представлены 20 фильмов с самым высоким средним рейтингом

Исходные файлы:

rating.csv

movies.csv

Листинг:

script.sh

```
#!/bin/bash
MAPPER1="mapper1.py"
REDUCER1="reducer1.py"
MAPPER2="mapper2.py"
REDUCER2="reducer2.py"
RATINGS="/user/hduser/input/ratings.csv"
MOVIES="/user/hduser/input/movies.csv"
OUTPUT1="/user/hduser/temp/output"
RESULT1="/user/hduser/temp/output/part-00000"
INPUT2="/user/hduser/temp/input"
OUTPUT="/user/hduser/output"
/usr/local/hadoop/bin/hdfs dfs -rm -R -f $OUTPUT1
/usr/local/hadoop/bin/hdfs dfs -rm -R -f $INPUT2
/usr/local/hadoop/bin/hdfs dfs -mkdir $INPUT2
/usr/local/hadoop/bin/hdfs dfs -rm -R -f $OUTPUT
```

```
/usr/local/hadoop/bin/mapred streaming -input $RATINGS -output $OUTPUT1 -mapper
$MAPPER1 -reducer $REDUCER1
/usr/local/hadoop/bin/hdfs dfs -cp $RESULT1 $INPUT2
/usr/local/hadoop/bin/hdfs dfs -cp $MOVIES $INPUT2
/usr/local/hadoop/bin/mapred streaming -input $INPUT2 -output $OUTPUT -mapper
$MAPPER2 -reducer $REDUCER2
/usr/local/hadoop/bin/hdfs dfs -head "$OUTPUT/part-00000"
```

mapper1.py

```
#!/usr/bin/python3.10
import sys

for line in sys.stdin:
    line = line.strip()
    if line != '':
        userId, movieId, rating, timestamp = line.split(',')
        if rating != 'rating':
            print(movieId, rating)
```

reducer1.py

```
#!/usr/bin/python3.10
import sys

ratings = {}
for line in sys.stdin:
    line = line.strip()
    movieId, rating = line.split()
    if movieId in ratings:
        ratings[movieId][0] += float(rating)
        ratings[movieId][1] += 1
    else:
        ratings[movieId] = [float(rating), 1]

for movieId in ratings:
    sum = ratings[movieId][0]
    count = ratings[movieId][1]
    ratings[movieId] = sum / count

ratings = dict(sorted(ratings.items(), key=lambda item: item[1], reverse=True))

i = 0
for movieId in ratings:
    print(f'{movieId},{ratings[movieId]},1')
    i += 1
    if i >= 20:
        break
```

mapper2.py

```
#!/usr/bin/python3.10
```

```
import sys

for line in sys.stdin:
    line = line.strip()
    if line != '':
        data = line.split(',')
        if len(data) == 3:
            if data[2] == '1':
                print(line)
            else:
                movieId, title, genres = data
                if title != 'title':
                    print(f'{movieId},{title}')
        if len(data) > 3:
            movieId = data[0]
            title = '#'.join(data[1:-1])
            print(f'{movieId},{title}')
```

reducer2.py

```
#!/usr/bin/python3.10
import sys

ratings = {}
titles = {}

for line in sys.stdin:
    line = line.strip()
    data = line.split(',')
    movieId = data[0]
    if len(data) == 3:
        ratings[movieId] = data[1]
    else:
        titles[movieId] = data[1].replace('#', ',')

i = 1
for movieId in ratings:
    title =  titles[movieId]
    rating = ratings[movieId]
    print(i, movieId, title, rating)
    i += 1
```

Результат:

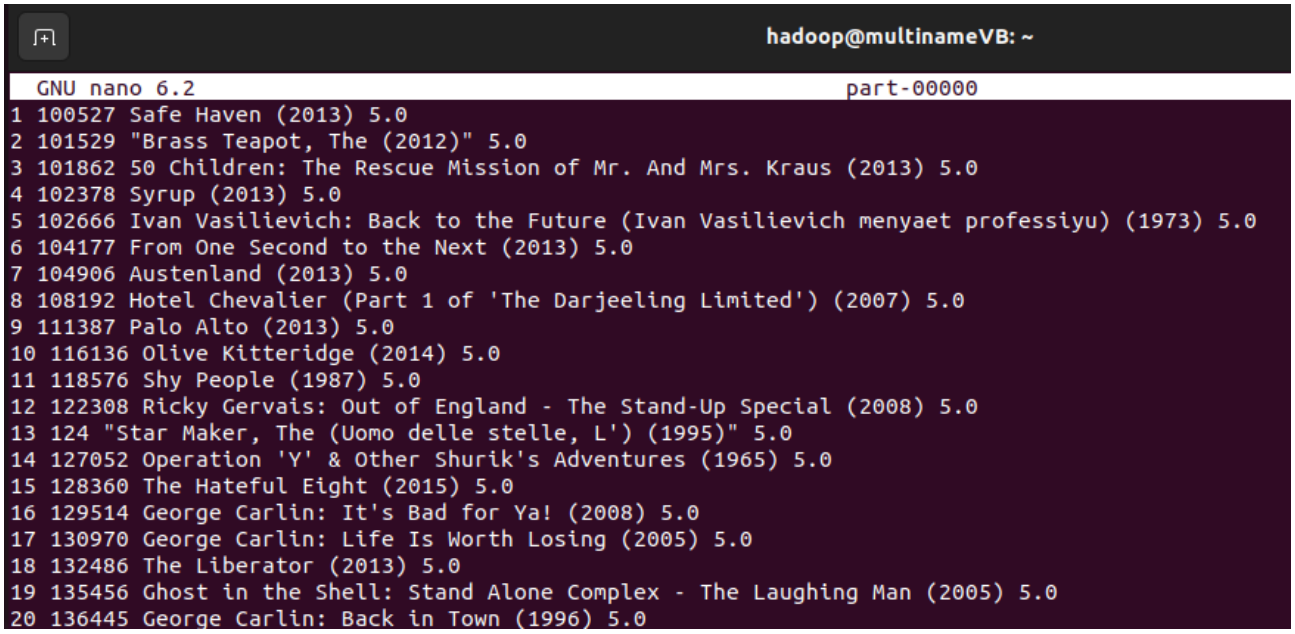

Рис. 1. Результат

Вывод: в ходе выполнения лабораторной работы были получены практические навыки использования цепочек MapReduce для решения сложных задач обработки больших данных.