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 _____ (Сафронов Н.С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Голубева С.Е.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2023

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

Постановка задачи

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

Вариант 4

Реализовать алгоритм PageRank. Входные данные – ориентированный граф (V,E), где вершины V представляют web-страницу, а ребра E – ссылки на страницы. Граф представлен в виде списка смежности – каждой странице соответсвует список, состоящий из страниц, на которые есть ссылки с данной страницы. Рассчитать значение PageRank для каждой страницы, для рассчета использовать как минимум 10 итераций.

$$PR(v) = \frac{1-d}{N} + d\sum_{i=1}^{n}\frac{PR(u_i)}{C(u_i)}$$

Входной файл: http://lintool.github.io/Cloud9/docs/exercises/sample-medium.txt

Ход выполнения работы

Листинг программы

pre_mapper.py:

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

import json
import sys


def process(line: str, separator: str = "\t") -> str:
    _line = line.strip()
    pages = _line.split(separator)
```

```
    if len(pages) == 0:
        return ""

    _node = {
        "adjacent": pages[1:],
        "pr": 1
    }
    yield f"{pages[0]}{separator}node{separator}{json.dumps(_node)}"

    pagerank = 1 / (len(pages[1:]) if len(pages[1:]) != 0 else 1)
    for i in range(1, len(pages)):
        yield f"{pages[i]}{separator}pagerank{separator}{pagerank}"


if __name__ == "__main__":
    for line in sys.stdin:
        for node in process(line):
            print(node)
```

mapper.py:

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

import json
import sys


def process(line: str, separator: str = "\t") -> str:
    _line = line.strip()
    node_id, info = _line.split(separator)
    _node = json.loads(info)

    yield f"{node_id}{separator}node{separator}{json.dumps(_node)}"

    adjacent = _node["adjacent"]
    pagerank = _node["pr"] / (len(adjacent) if len(adjacent) != 0 else 1)

    for i in range(len(adjacent)):
        yield f"{adjacent[i]}{separator}pagerank{separator}{pagerank}"


if __name__ == "__main__":
    for line in sys.stdin:
        for node in process(line):
            print(node)
```

reducer.py:

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

import json
import sys
```

```
class Reducer:

    def __init__(self, graph_length: int):
        self._current_node = {}
        self._current_node_id = None
        self._current_sum = 0
        self._graph_length = graph_length

    def _reduce_line(self, line: str, separator: str = "\t"):
        node_id, label, info = line.split(separator)

        if node_id == self._current_node_id:
            if label == "pagerank":
                self._current_sum += float(info)
            elif label == "node":
                self._current_node = json.loads(info)
        else:
            if self._current_node_id is not None and (
                    self._current_node is not None):
                self._current_node["pr"] = (0.15 /
float(self._graph_length) +
                                            0.85 * self._current_sum)
                result = json.dumps(self._current_node)
                print(f"{self._current_node_id}{separator}{result}")
                self._current_sum = 0
                self._current_node = None
            if label == "node":
                self._current_sum = 0
                self._current_node = json.loads(info)
                self._current_node_id = node_id
            elif label == "pagerank":
                self._current_sum += float(info)

    def reduce(self, stream):
        for line in stream:
            self._reduce_line(line, "\t")


if __name__ == "__main__":
    reducer = Reducer(graph_length=316)
    reducer.reduce(sys.stdin)
```

post_mapper.py:

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

import json
import sys
```

```
def process(line: str, separator: str = "\t") -> str:
    _line = line.strip()
    node_id, info = _line.split(separator)
    _node = json.loads(info)

    yield f"{_node['pr']}{separator}{node_id}"


if __name__ == "__main__":
    for line in sys.stdin:
        for node in process(line):
            print(node)
```

mapred.sh:

```
#!/bin/bash
PRE_MAPPER="pre_mapper.py"
MAPPER="mapper.py"
POST_MAPPER="post_mapper.py"
REDUCER="reducer.py"
INPUT=$1
OUTPUT="/user/hduser/lab3_output"
iteration=0
MAX_ITERATIONS=10
echo "[SCRIPT] Removing Output $OUTPUT..."
/usr/local/hadoop/bin/hdfs dfs -rm -r -f $OUTPUT
iteration=$((iteration + 1))
echo "[SCRIPT] Starting MapReduce Job For $INPUT:
$iteration/$MAX_ITERATIONS"
/usr/local/hadoop/bin/hdfs dfs -rm -r -f "$OUTPUT.$iteration"
/usr/local/hadoop/bin/mapred streaming -input "$INPUT" -output
"$OUTPUT.$iteration" -mapper $PRE_MAPPER -reducer $REDUCER

for i in $(seq $((iteration + 1)) $MAX_ITERATIONS)
do
  echo "[SCRIPT] Starting MapReduce Job For $OUTPUT.$i: $i/$MAX_ITERATIONS"
  /usr/local/hadoop/bin/hdfs dfs -rm -r -f "$OUTPUT.$i"
  /usr/local/hadoop/bin/mapred streaming -input "$OUTPUT.$((i-1))/part-
00000" -output "$OUTPUT.$i" -mapper $MAPPER -reducer $REDUCER
done

echo "[SCRIPT] Starting Final MapReduce Job For $OUTPUT.$iteration..."
/usr/local/hadoop/bin/hdfs dfs -rm -r -f "$OUTPUT"
/usr/local/hadoop/bin/mapred streaming -D
mapred.text.key.comparator.options=-nr -input
"$OUTPUT.$MAX_ITERATIONS/part-00000" -output "$OUTPUT" -mapper $POST_MAPPER

/usr/local/hadoop/bin/hdfs dfs -cat "$OUTPUT/part-00000" | sort -r
```

Результаты выполнения программы

```
hadoop@keyone-laptop:~/lab3$ hdfs dfs -head /user/hduser/lab3/input.txt
8833657 14520693        9322955 12525256        11262195        8188767
12019176        10232609        11102895        10851068        12168088        12581018
11005099        11260861        12530085        10917054        9231155 9591538
12890810        10611187        7593132 11223541        11020056        11102895
8626706 8675593 8646598 10545285        8188767 10974650
12414625        11704861        9815941 14534541        12960057        12776186
12188907        9831246 12556535        14528284        12360481        12379772
7624149 7959292 7959284 7977648 12112314        7509629
9815544 8609661 7525978 9842972 12150454        14534541
7529048 8732665 7529049 11774034        10232609        11223541
9545345 10674396        10816436        9852124 11559546        9832424
11500939        11559546        10207053        12138094        11594774        11274216
12767520        10699960        9883987 11704861        8626686 11498020
9699666 11559546        10766344        11704861        9056422 10642432
10669726        11546773        9545345 10674396        10847581        9852124
9342439 9342440 9488037 11879554        10430101        12897145
11376126        10967123        12867429        11559546        10602507        11337496
14534541        10319328        11489838        9545345 10077005        11479214
11788654        10229198        11376121        10207053        12447687        12138094
12746329        11713209        10846175        9633824 hadoop@keyone-laptop:~/lab3$
```

Рисунок 1. Входной файл

```
hadoop@keyone-laptop:~/lab3$ ./mapred.sh /user/hduser/lab3/input.txt
```

Рисунок 2. Запуск скрипта с задачами MapReduce

```
0.002223079997824967    11559546
0.0016124051616485985   9205064
0.001051462333865202    12543789
0.0010513418303313767   12566306
0.0009659874006926151   12138094
0.0009569272733693515   11245436
0.0009058041049259045   9056422
0.0009014733203627191   12937141
0.0008520700501507543   11872675
0.0008464307615360035   11102895
0.0008436305523340016   10319328
0.000832522358847675    12379772
0.000766396850791641    11788654
0.0007663766908526366   12447687
0.0007450739916498965   10791772
0.000740080915299428    10777488
0.0007306110098629392   12592384
0.0007304905063291139   8188767
0.0007304905063291139   11145562
0.000726755875809273 9  9516080
0.0007133096509094491   9815941
0.0007105665976814511   12374697
0.000702694180526052    12438247
0.0006994190026903753   11500939
0.0006663224301357677   9545345
0.0006635126582278481   12068000
0.000657873777142508    10766344
0.0006497943037974683   8732665
0.0006497943037974683   11278691
0.0006360759493670886   9539310
0.0006360759493670886   8620490
0.0006360759493670886   8609661
0.0006360759493670886   7593132
0.0006360759493670886   12767520
0.0006360759493670886   10383152
0.0006298703951498056   12593846
0.0006298703951498056   12533667
0.0006049842857893455   11376121
0.0005970972661720044   9699666
0.0005970972661720044   7664648
0.0005966393054700213   10931447
0.0005879785077962912   10816436
0.0005874938835182802   11896447
0.0005828164556962025   11498020
0.0005828164556962025   10699960
0.0005771775746108625   8174129
0.0005771775746108625   10642432
0.0005719280331127469   9815813
```

Рисунок 3. Демонстрация файла с результатом

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