|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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.

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)

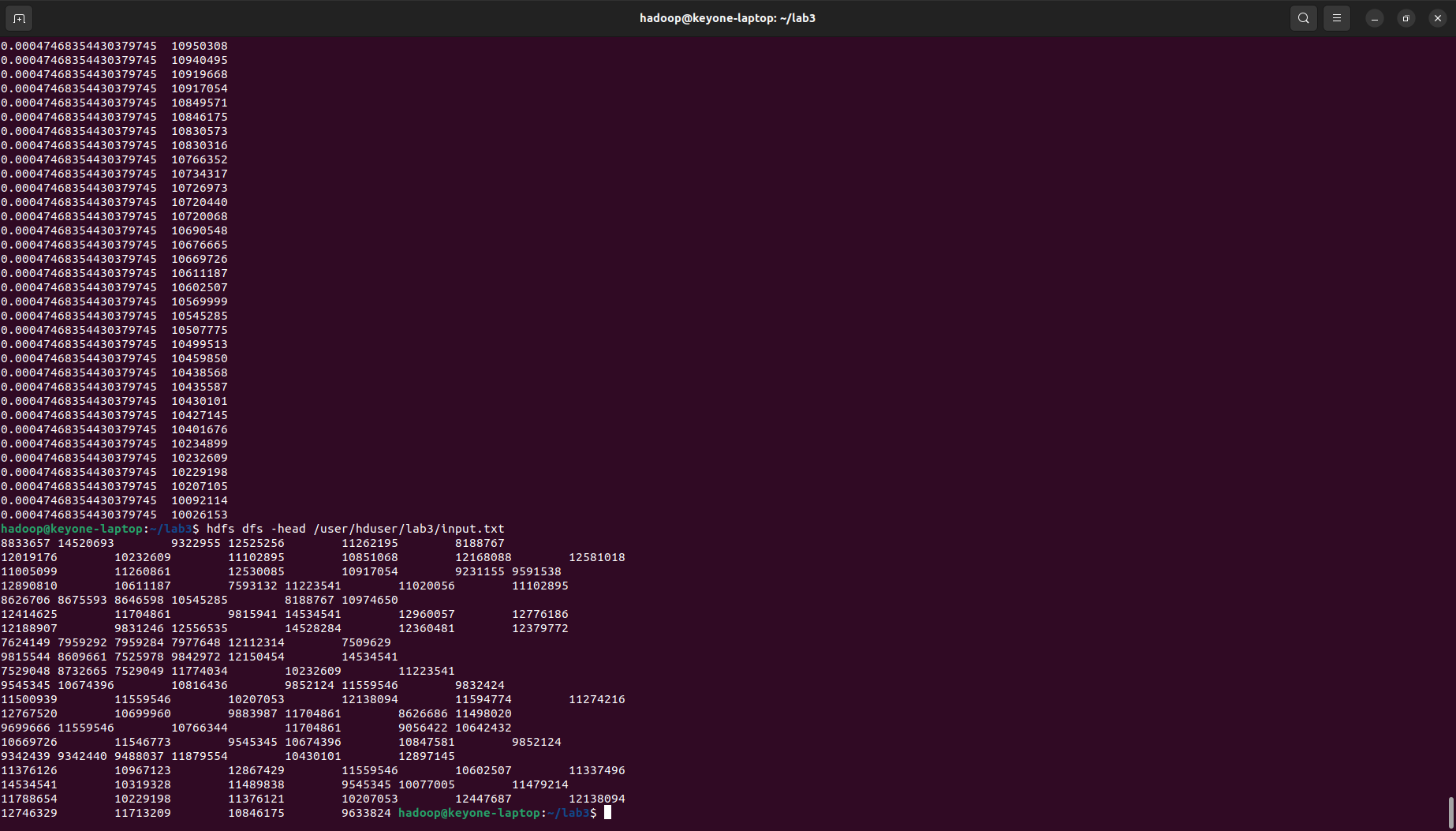
**post\_mapper.py:**

#!/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)

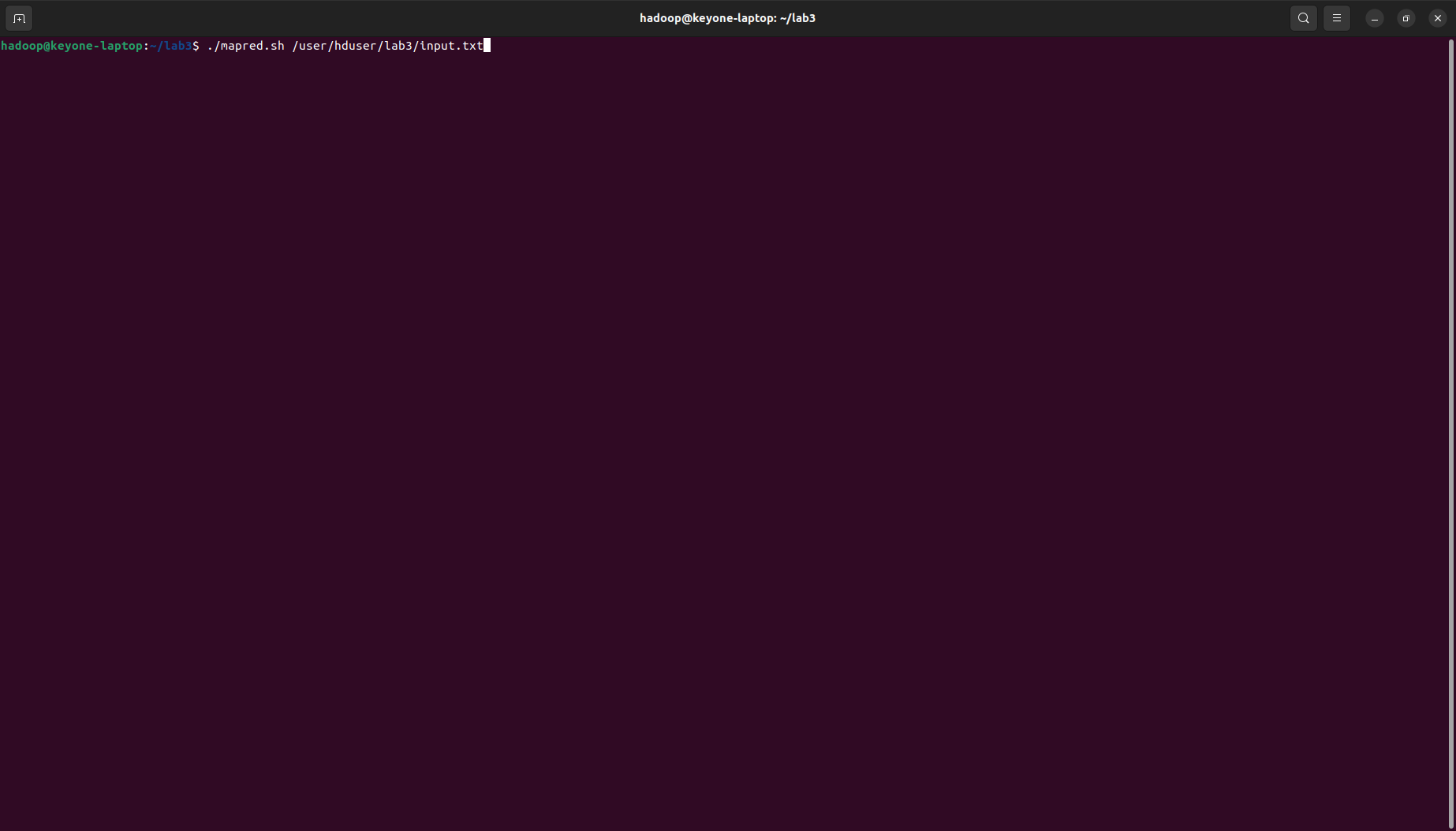
**mapred.sh:**

**#!/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

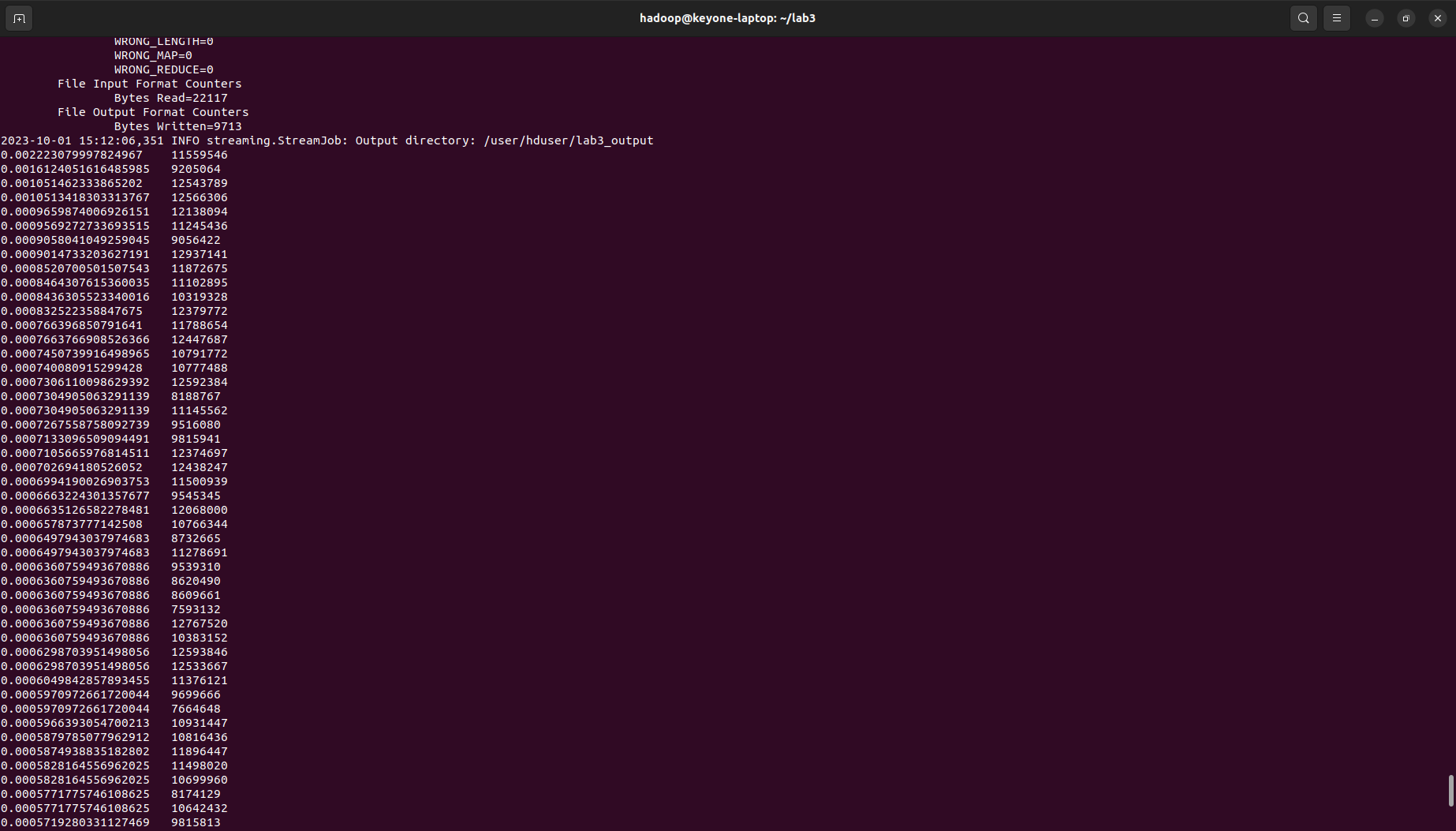
**Результаты выполнения программы**



**Рисунок 1.** Входной файл



**Рисунок 2.** Запуск скрипта с задачами MapReduce



**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.