

Минобрнауки России  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Кафедра Электронно-вычислительные машины и системы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
**к курсовой работе (проекту)**

по дисциплине Системы обработки больших данных

на тему разработка программы обработки больших данных по теме «Записи  
выдачи книг в библиотеке Сиэтла»

Студент Бобунов Артем Владимирович  
(фамилия, имя, отчество)

Группа САПР-1.1

Руководитель работы (проекта) \_\_\_\_\_ П.Д. Кравченя  
(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

\_\_\_\_\_  
(подпись и дата подписания) (инициалы и фамилия)

\_\_\_\_\_  
(подпись и дата подписания) (инициалы и фамилия)

\_\_\_\_\_  
(подпись и дата подписания) (инициалы и фамилия)

Нормоконтролер \_\_\_\_\_  
(подпись, дата подписания) (инициалы и фамилия)

Волгоград 2021 г.

Минобрнауки России  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Направление (специальность) 09.04.01 — Информатика и вычислительная техника

Кафедра Электронно-вычислительные машины и системы

Дисциплина Системы обработки больших данных

Утверждаю

Зав. кафедрой \_\_\_\_\_

« \_\_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**  
**на курсовую работу (проект)**

Студент Бобунов Артем Владимирович

(фамилия, имя, отчество)

Группа САПР-1.1

1. Тема: разработка программы обработки больших данных по теме «Записи выдачи книг в библиотеке Сиэтла»

Утверждена приказом от « \_\_\_\_\_ » \_\_\_\_\_ 20\_\_ г. № \_\_\_\_\_

2. Срок представления работы (проекта) к защите « \_\_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

3. Содержание расчетно-пояснительной записки: Обзор методов и алгоритмов обработки больших данных, Проектирование и реализация системы анализа данных, Получение и интерпретация результатов

4. Перечень графического материала: \_\_\_\_\_

5. Дата выдачи задания « \_\_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

Руководитель работы (проекта) \_\_\_\_\_ П.Д. Кравченя

подпись, дата

инициалы и фамилия

Задание принял к исполнению \_\_\_\_\_

подпись, дата

инициалы и фамилия

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ..	5
1.1 Обзор методов анализа больших данных .....	5
1.2 Описание алгоритмов обработки данных.....	6
1.3 Выводы .....	8
2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИСТЕМЫ АНАЛИЗА ДАННЫХ...	9
2.1 Постановка задачи.....	9
2.2 Описание датасета.....	9
2.3 Алгоритмы MapReduce .....	10
2.4. Реализация алгоритма с использованием Фреймворка MapReduce .....	11
3 ПОЛУЧЕНИЕ И ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ .....	21
3.1 Описание компиляции и запуска разработанного приложения .....	21
3.2 Интерпретация результатов .....	23
3.3 Исследование масштабируемости приложения анализа данных .....	24
3.4 Выводы .....	25
ЗАКЛЮЧЕНИЕ .....	26
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	27

## ВВЕДЕНИЕ

В настоящее время системы обработки и анализа больших данных широко используются для доступа к крупномасштабным данным для извлечения полезной информации для поддержки и принятия решений

Целью данной работы является провести исследования по алгоритмам больших данных, и на основе выбранного алгоритма разработать программу по обработке больших данных с использованием выбранных технологий. Для достижения поставленной цели требуется решение следующих задач:

- найти количество повторений для каждого элемента атрибуты ItemType входного массива данных «Записи выдачи книг в библиотеке Сиэтла» и вывести в выходной файл используя технологии MapReduce и HDFS.

- найти для каждого читательского номера(читателя), в течении какого часа у читателя возникает больше всего потребностей к приобретению книги (результат вывести со смещением часового пояса) используя технологии MapReduce и HDFS.

- найти количество повторяющихся значений BibNumber входного массива данных «Записи выдачи книг в библиотеке Сиэтла» и вывести в выходной файл используя технологии MapReduce и HDFS.

В первом разделе работы приведен обзор методов и алгоритмов обработки больших данных MapReduce и Spark.

Во втором разделе описывается проектирование и реализация разработанного приложения по обработке больших данных с использованием технологий Hadoop MapReduce и распределенной файловой системой HDFS.

В третьем разделе представлено получение результатов обработки больших на основе выделенных задач.

# 1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ

## 1.1 Обзор методов анализа больших данных

Под анализом больших данных понимается как анализ массивов данных в рамках возможностей персонального компьютера, так и в рамках возможностей систем управления реляционными базами данных, при этом как в первом, так и во втором случае при формировании и статистики, и визуализации возникают определенные трудности, которые заключаются в необходимости обеспечения скоординированной работы компьютерных программ на десятках, сотнях или даже тысячах серверов. Анализ больших данных может быть охарактеризован по следующим параметрам:

- Объем - количество генерируемых данных. От этого показателя зависит, может ли определенный массив данных считаться большими данными или нет. Данные хранятся SQL-серверах в облачной среде.
- Многообразие - категория, к которой принадлежат большие данные. Знание такой принадлежности позволяет аналитикам наиболее эффективно работать с информацией.
- Скорость - скорость генерирования или обработки данных с целью осуществления поставленных целей.
- Изменчивость - нестабильность данных во времени.
- Достоверность - качество собранных данных, от которого зависит точность анализа.
- Сложность - трудоемкость процесса корреляции и построения взаимосвязей между данными.

Рассмотрим основные методы анализа больших данных. Методы анализа больших данных, применяемых в современных технологиях, можно отобразить с помощью следующей диаграммы:

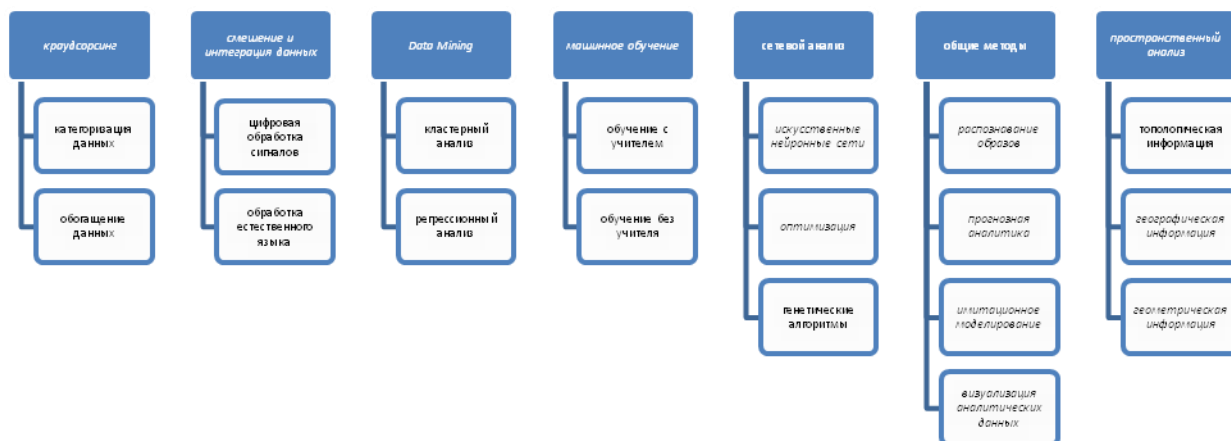


Рисунок 1 – Методы анализа больших данных

## 1.2 Описание алгоритмов обработки данных

Информация становится достаточно большой, чтобы осознать необходимость масштабирования традиционных алгоритмов. Поскольку данные не помещаются в память и распределены по машинам, алгоритмы также должны соответствовать распределенному хранилищу. Алгоритмы, называемые алгоритмами обработки больших данных, включают в себя случайные блуждания, распределенные хеш-таблицы, потоковую передачу, массовую синхронную обработку (BSP) и парадигмы MapReduce, платформы Spark. Каждый из этих алгоритмов уникален в своем подходе и подходит для определенных задач [1].

Целью алгоритмов является сокращение сетевых коммуникаций в распределенной сети, минимизация перемещений данных, снижение синхронных задержек и оптимизация вычислительных ресурсов. Далее кратко описаны основные алгоритмы обработки больших данных:

- Классический MapReduce, Apache компонент Hadoop для обработки данных, проводит вычисления в два этапа:

- а) Map, когда главный узел кластера (master) распределяет задачи по рабочим узлам (node).

б) Reduce, когда данные сворачиваются и передаются обратно на главный узел, формируя окончательный результат вычислений.

Пока все процессы этапа Map не закончатся, процессы Reduce не начнутся. При этом все операции проходят по циклу чтение-запись с жесткого диска. Это обуславливает задержки в обработке информации.

Таким образом, технология MapReduce хорошо подходит для задач распределенных вычислений в пакетном режиме, но из-за задержек (latency) не может использоваться для потоковой обработки в режиме реального времени. Для решения этой проблемы был создан Apache Spark и другие Big Data фреймворки распределенной потоковой обработки.

– В отличие от классического обработчика ядра Apache Hadoop с двухуровневой концепцией MapReduce на базе дискового хранилища, Spark использует специализированные примитивы для рекуррентной обработки в оперативной памяти. Благодаря этому многие вычислительные задачи реализуются в Спарк значительно быстрее.

Spark может работать как в среде кластера Hadoop под управлением YARN, так и без компонентов ядра хадуп, например, на базе системы управления кластером Mesos. Спарк поддерживает несколько популярных распределённых систем хранения данных (HDFS, OpenStack Swift, Cassandra, Amazon S3) и языков программирования (Java, Scala, Python, R), предоставляя для них API-интерфейсы.

Абстракция Spark для потока называется DStream (discretized stream, дискретизированный поток) и представляет собой микро-пакет, содержащий несколько отказоустойчивых распределенных датасетов, RDD (resilient distributed dataset).

Именно RDD является основным вычислительным примитивом Спарк, над которым можно делать параллельные вычисления и преобразования с помощью встроенных и произвольных функций, в том числе с помощью временных окон (window-based operations) . Подробнее про временные окна мы рассказывали здесь на примере Apache Kafka Streams.

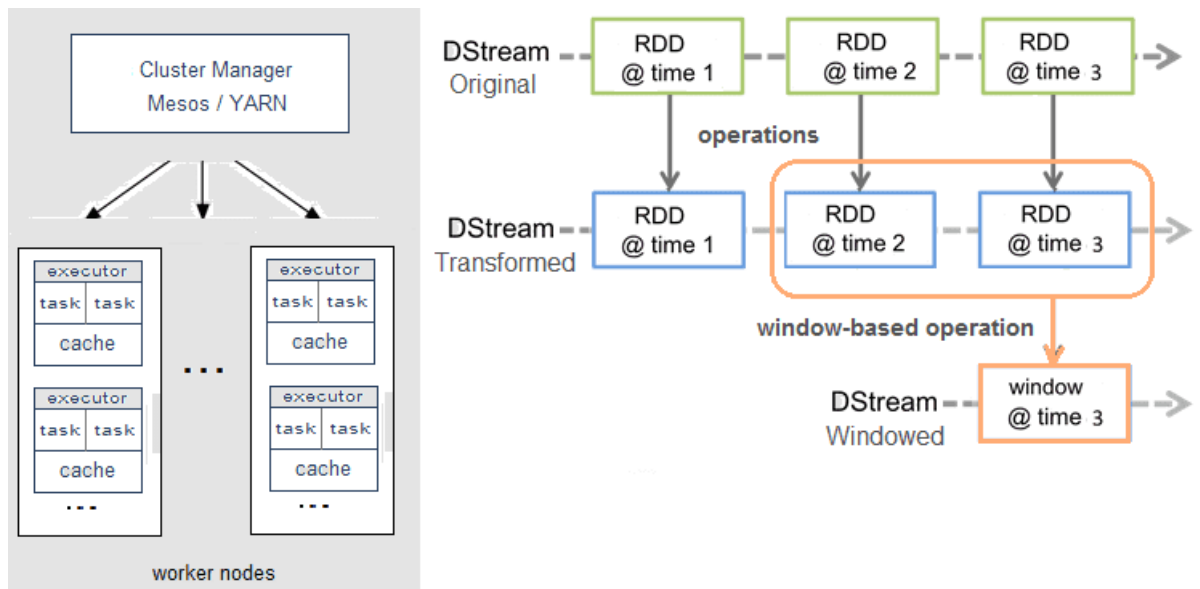


Рисунок 2 – Принцип работы Spark

### 1.3 Выводы

Проведенный обзор позволяет сформулировать следующие выводы, что MapReduce используется для реализации сложных пакетных задач. Spark способен выполнять пакетную обработку заданий в разы быстрее за счет уменьшения количества операций чтения/записи.



## 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИСТЕМЫ АНАЛИЗА ДАННЫХ

### 2.1 Постановка задачи

В курсовой работе, требуется программно реализовать три задачи для обработки записей по выдачи книг в библиотеке Сиэтла. В качестве исходных данных используется набор данных включающий журнал всех проверок физических предметов из публичной библиотеки Сиэтла. Реализацию программы анализа данных необходимо осуществить с использованием Hadoop MapReduce, HDFS и потоковой передачей данных Hadoop Streaming.

### 2.2 Описание датасета

Этот набор данных включает журнал всех проверок физических предметов из публичной библиотеки Сиэтла. Он расположен в открытом доступе по адресу [4]. Набор данных начинается с проверок, совершенных в период с апреля 2005 г. по сентябрь 2017 г. Продления не включены.

Набор данных содержит шесть атрибут:

- BibNumber – атрибут, отражающий стартовый номер читателя.
- ItemBarcode – атрибут, отражающий товарный штрих-код.
- ItemType – атрибут, отражающий тип элемента.
- Collcetion – атрибут, отражающий коллекцию.
- CallNumber – атрибут, отражающий уникальный номер.
- CheckoutDateTime – атрибут, отражающий дату выдачи товара.

## 2.3 Алгоритмы MapReduce

Алгоритм нахождения количества повторений значений атрибута ItemType в рамках модели MapReduce может быть представлен следующей последовательностью MapReduce этапа



Рисунок 3 – Первый этап MapReduce

Алгоритм нахождения для каждого читателя, в течении какого часа у читателя возникает больше всего потребностей к приобретению книги (результат вывести со смещением часового пояса) в рамках модели MapReduce может быть представлен следующей последовательностью MapReduce этапа:

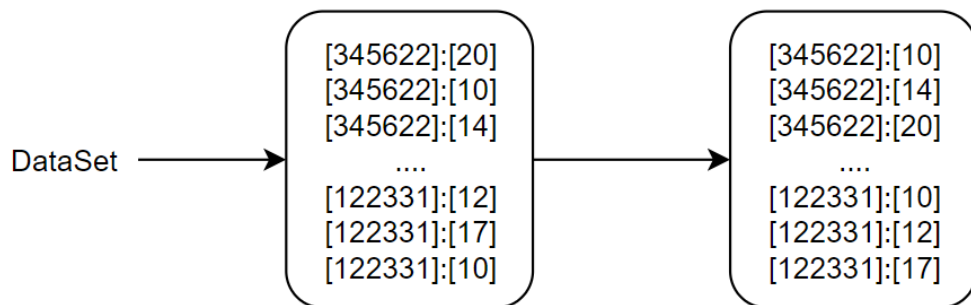


Рисунок 4 – Второй этап MapReduce

Алгоритм поиска количества повторяющихся значений BibNumber входного массива данных «Записи выдачи книг в библиотеке Сиэтла» и вывести в выходной файл используя технологии MapReduce и HDFS.

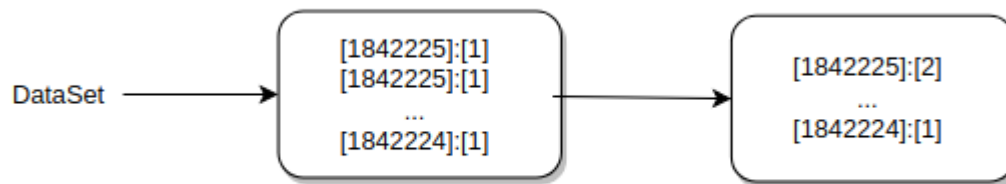


Рисунок 5 – Третий этап MapReduce

## 2.4. Реализация алгоритма с использованием Фреймворка MapReduce

В соответствии с приведенным алгоритмом MapReduce для нахождения количества повторений значений атрибута ItemType, был разработан программный код на языке Python. Первый процесс Map должен вывести значения атрибута ItemType, и каждому значению задать 1. Реализация первого этапа Map выполнена следующим образом:

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
    line = line.strip()
    columns = line.split(',')
    if len(columns) == 6:
        try:
            print("{}\t{}".format(columns[2],1))
        except ValueError:
            pass
```

Первый процесс Reduce должен пройти по входному массиву и найти количество вхождения определённого ключа и вывести их количество. Реализация первого этапа Reduce выполнена следующим образом:

```
#!/usr/bin/env python
import sys
ItemTypeCount = {}
```

for line in sys.stdin:

```
    line = line.strip()
```

```
    word, count = line.split('\t', 1)
```

```
    try:
```

```
        count = int(count)
```

```
    except ValueError:
```

```
        continue
```

```
    try:
```

```
        ItemTypeCount[word] = ItemTypeCount[word]+count
```

```
    except:
```

```
        ItemTypeCount[word] = count
```

```
for word in ItemTypeCount.keys():
```

```
    print ('%s\t%s' % ( word, ItemTypeCount[word] ))
```

Входные значения mapper1.py задаются в командной строке и имеют следующий вид: `-input /sema/ Checkouts_By_Title_Data_Lens_2005.csv`

Данные были добавлены в распределенную файловую систему HDFS с помощью команды: `hadoop fs -copyFromLocal /home/Hadoop/Checkouts_By_Title_Data_Lens_2005.csv /sema`

В приведенной ниже команде используется потоковая передача Hadoop Streaming. Mapper и Reducer являются исполняемыми файлами, которые считывают входные данные и выдают выходные данные. Команда создаст задание MapReduce и отправит его на кластер и будет отслеживать ход выполнения задания до завершения [2]. Для запуска программы необходимо ввести следующую команду в терминале:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -input  
/sema/Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output -mapper  
/home/hadoop/map.py -reducer /home/hadoop/reducer.py -file home/hadoop/map.py  
-file /home/hadoop/reducer.py
```

```

hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -input /sema/
Checkouts By Title Data Lens 2005.csv -output /sout/output18 -mapper /home/hadoop/map.py -reducer /home/hadoop/reducer.py -file /home/h
hadoop/map.py -file /home/hadoop/reducer.py
2022-01-19 20:47:44,697 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/hadoop/map.py, /home/hadoop/reducer.py, /tmp/hadoop-unjar14955659657540461026/] [] /tmp/streamjob1687789617126558
954.jar tmpDir=null
2022-01-19 20:47:45,392 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-01-19 20:47:45,510 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-01-19 20:47:46,100 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging
/job_1642587955476_0006
2022-01-19 20:47:50,664 INFO mapred.FileInputFormat: Total input files to process : 1
2022-01-19 20:47:50,673 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2022-01-19 20:47:53,465 INFO mapreduce.JobSubmitter: number of splits:2
2022-01-19 20:47:54,417 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642587955476_0006
2022-01-19 20:47:54,417 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-01-19 20:47:54,552 INFO conf.Configuration: resource-types.xml not found
2022-01-19 20:47:54,553 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-01-19 20:47:54,590 INFO Impl.YarnClientImpl: Submitted application application_1642587955476_0006
2022-01-19 20:47:54,612 INFO mapreduce.Job: The url to track the job: http://artemUbuntu:8088/proxy/application_1642587955476_0006/
2022-01-19 20:47:54,613 INFO mapreduce.Job: Running job: job_1642587955476_0006
2022-01-19 20:47:59,661 INFO mapreduce.Job: Job job_1642587955476_0006 running in uber mode : false
2022-01-19 20:47:59,662 INFO mapreduce.Job: map 0% reduce 0%
2022-01-19 20:48:15,762 INFO mapreduce.Job: map 17% reduce 0%
2022-01-19 20:48:17,773 INFO mapreduce.Job: map 33% reduce 0%
2022-01-19 20:48:21,786 INFO mapreduce.Job: map 42% reduce 0%
2022-01-19 20:48:23,794 INFO mapreduce.Job: map 51% reduce 0%
2022-01-19 20:48:27,808 INFO mapreduce.Job: map 59% reduce 0%
2022-01-19 20:48:29,825 INFO mapreduce.Job: map 83% reduce 0%
2022-01-19 20:48:31,836 INFO mapreduce.Job: map 100% reduce 0%
2022-01-19 20:48:42,886 INFO mapreduce.Job: map 100% reduce 100%

```


Рисунок 6 – Запуск программы

```

CPU time spent (ms)=53890
Physical memory (bytes) snapshot=1109966848
Virtual memory (bytes) snapshot=8243187712
Total committed heap usage (bytes)=792723456
Peak Map Physical memory (bytes)=401678336
Peak Map Virtual memory (bytes)=2771173376
Peak Reduce Physical memory (bytes)=309985280
Peak Reduce Virtual memory (bytes)=2750210048
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=273090246
File Output Format Counters
Bytes Written=5915734
2022-01-19 20:48:50,042 INFO streaming.StreamJob: Output directory: /sout/output18

```

Рисунок 7 – Запуск программы


MapReduce Application application\_1642587955476\_0006
Logged in as: dr:who

Cluster
Application
About Jobs
Tools

Active Jobs
Show 20 entries
Search:

Job ID	Name	State	Map Progress	Maps Total	Maps Completed	Reduce Progress	Reduces Total	Reduces Completed
job_1642587955476_0006	streamjob1687789617126558954.jar	RUNNING		2	0		1	0

Showing 1 to 1 of 1 entries
First Previous 1 Next Last

Рисунок 8 – Проверка запуска MapReduce

Выходные значения метода reducer1.py представляет собой массив данных. Для того чтобы отобразить данные, необходимо выполнить следующую команду:

```

2022-01-19 20:44:13,649 INFO streaming.StreamJob: Output directory: /sout/output17
hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop fs -getmerge /sout/output17/part-00000 /home/hadoop/fileOutput.txt
hadoop@artemUbuntu: /usr/local/hadoop/sbin$

```

Рисунок 9 – Выгрузка данных

Результат выполнения первого MapReduce этапа:

Открыть			24 ardv	2	
			25 arkit	1	
			26 armfc	8	
			27 armfm	5	
			28 armus	6	
			29 arnp	1	
			30 arper	47	
			31 arunkn	7	
			32 arvhs	7	
			33 arvid	1	
			34 bcbk	4052	
			35 bccas	1578	
			36 bccd	238	
			37 bccdrom	180	
			38 bckit	496	
			39 bcvhs	1547	
			40 blvhs	164	
			41 dcillb	17509	
			42 dcilll	26	
			43 jcbk	742786	
			44 jccas	21369	
			45 jccd	43901	
			46 jccdrom	1942	
			47 jcdvd	109408	
			48 jckit	5879	
			49 jcmus	276	
			50 jcrec	3	
			51 jcvhs	102217	
			52 jlvhs	116	
			53 jrbk	45	
			54 jrkit	2	
			55 jrvhs	2	
			56 ucflpdr	18	
			57 ucfold	12012	
			58 ucunkn	62	
			59 ucunknj	1287	
			60 unk	1	

Рисунок 10 – Результат выполнения первой задачи

В соответствии с приведенным алгоритмом MapReduce для каждого читательского номера(читателя), в течении какого часа у читателя возникает больше всего потребностей к приобретению книги (результат вывести со смещением часового пояса) используя технологии MapReduce и HDFS. Второй процесс Map должен вывести количество часов атрибута CheckoutDateTime для каждого значения атрибута BibNumber. Реализация второго этапа Map выполнена следующим образом:

```
#!/usr/bin/env python
```

```
import sys
```

```
from datetime import datetime
```

```
for line in sys.stdin:
```

```
    line = line.strip()
```

```
    columns = line.split(',')
```

```
    if len(columns) == 6:
```

```
        try:
```

```
            countUpVotes = datetime.strptime(columns[5], "%m/%d/%Y %I:%M:%S %p").hour
```

```
            print ("%s\t%s" % (columns[0],countUpVotes))
```

```
except ValueError:
```

```
    pass
```

Второй процесс Reduce должен сделать сортировку по возрастанию для каждой пары ключ значение с одинаковым ключем. Реализация второго этапа Reduce выполнена следующим образом:

```
#!/usr/bin/python
```

```
import sys
```

```
import operator
```

```
oldKey = None
```

```
dicHour = { }
```

```
print ("ItemBarcode | \tHour")
```

```
for line in sys.stdin:
```

```
    data_mapped = line.strip().split("\t")
```

```
    if len(data_mapped) != 2: continue
```

```
    thisKey, thisHour = data_mapped
```

```
    if oldKey and oldKey != thisKey:
```

```
        sorted_hours = sorted(dicHour.items(), key = operator.itemgetter(1), reverse =
True)
```

```
        topCount = sorted_hours[0][1]
```

```
        for tuple in sorted_hours:
```

```
            if tuple[1]==topCount: print ("{}{}\t{}".format(oldKey,tuple[0]))
```

```
        oldKey = thisKey;
```

```
        dicHour = { }
```

```
        oldKey = thisKey
```

```
        if thisHour in dicHour:
```

```
            dicHour[thisHour] += 1
```

```
        else:
```

```
            dicHour[thisHour] = 1
```

```
if oldKey != None:
```

```
sorted_hours = sorted(dicHour.items(), key = operator.itemgetter(1), reverse =
True)
topCount = sorted_hours[0][1]
for tuple in sorted_hours:
    if tuple[1]==topCount: print ("{0}\t{1}".format(oldKey,tuple[0]))
```

Входные значения mapper1.py задаются в командной строке и имеют следующий вид: -input /sema/ Checkouts\_By\_Title\_Data\_Lens\_2005.csv

Для запуска программы необходимо ввести следующую команду в терминале:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -input
/sema/Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output -mapper
/home/hadoop/mapper1.py -reducer /home/hadoop/reducer1.py -file
/home/hadoop/mapper1.py -file /home/hadoop/reducer1.py
```

```
hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -input /sema/
Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output17 -mapper /home/hadoop/mapper1.py -reducer /home/hadoop/reducer1.py -file /h
ome/hadoop/mapper1.py -file /home/hadoop/reducer1.py
2022-01-19 20:43:33,191 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/hadoop/mapper1.py, /home/hadoop/reducer1.py, /tmp/hadoop-unjar14228013752627699850/] [] /tmp/streamjob12637824013
058085699.jar tmpDir=null
2022-01-19 20:43:34,095 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-01-19 20:43:34,224 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-01-19 20:43:34,837 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging
/job_1642587955476_0005
2022-01-19 20:43:38,858 INFO mapred.FileInputFormat: Total input files to process : 1
2022-01-19 20:43:38,871 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2022-01-19 20:43:40,739 INFO mapreduce.JobSubmitter: number of splits:2
2022-01-19 20:43:42,052 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642587955476_0005
2022-01-19 20:43:42,052 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-01-19 20:43:42,183 INFO conf.Configuration: resource-types.xml not found
2022-01-19 20:43:42,183 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-01-19 20:43:42,228 INFO impl.YarnClientImpl: Submitted application application_1642587955476_0005
2022-01-19 20:43:42,254 INFO mapreduce.Job: The url to track the job: http://artemUbuntu:8088/proxy/application_1642587955476_0005/
2022-01-19 20:43:42,255 INFO mapreduce.Job: Running job: job_1642587955476_0005
2022-01-19 20:43:47,365 INFO mapreduce.Job: Job job_1642587955476_0005 running in uber mode : false
2022-01-19 20:43:47,369 INFO mapreduce.Job: map 0% reduce 0%
2022-01-19 20:43:55,455 INFO mapreduce.Job: map 50% reduce 0%
2022-01-19 20:43:59,475 INFO mapreduce.Job: map 100% reduce 0%
2022-01-19 20:44:07,516 INFO mapreduce.Job: map 100% reduce 100%
2022-01-19 20:44:13,567 INFO mapreduce.Job: Job job_1642587955476_0005 completed successfully
2022-01-19 20:44:13,649 INFO mapreduce.Job: Counters: 56
```

Рисунок 11 – Запуск второго задачи MapReduce

```
Merged Map outputs=2
GC time elapsed (ms)=72
CPU time spent (ms)=12940
Physical memory (bytes) snapshot=1051758592
Virtual memory (bytes) snapshot=8241270784
Total committed heap usage (bytes)=780140544
Peak Map Physical memory (bytes)=388100096
Peak Map Virtual memory (bytes)=2738700288
Peak Reduce Physical memory (bytes)=285065216
Peak Reduce Virtual memory (bytes)=2767204352
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=273090246
File Output Format Counters
Bytes Written=609
2022-01-19 20:44:13,649 INFO streaming.StreamJob: Output directory: /sout/output17
```

Рисунок 12 – Запуск второго задачи MapReduce





Cluster	Active Jobs									
Application	Show 20 entries Search:									
About Jobs	Job ID	Name	State	Map Progress	Maps Total	Maps Completed	Reduce Progress	Reduces Total	Reduces Completed	
Tools	job_1642587955476_0005	streamjob12637824013058085699.jar	RUNNING		2	2		1	0	
Showing 1 to 1 of 1 entries										
First Previous 1 Next Last										

Рисунок 13 - Проверка запуска MapReduce

Выходные значения метода `reducer1.py` представляет собой массив данных. Для того чтобы отобразить данные, необходимо выполнить следующую команду (аналогичная команда, как и для первой задачи, только `output` необходимо указать другой):

```
2022-01-19 20:44:13,649 INFO streaming.StreamJob: Output directory: /sout/output17
hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop fs -getmerge /sout/output17/part-00000 /home/hadoop/fileOutput.txt
hadoop@artemUbuntu: /usr/local/hadoop/sbin$
```

Рисунок 14 – Выгрузка данных

Результаты выполнения второго MapReduce

```
1 ItemBarcode | Hour
2 1000 15
3 1000 19
4 1000 16
5 1000 13
6 1000 12
7 10001 19
8 100012 15
9 100014 17
10 100014 14
11 100027 16
12 100027 17
13 100029 11
14 1000338 13
15 100034 15
16 100042 15
17 100042 13
18 100042 9
19 100042 14
20 100047 11
21 100048 14
22 100052 16
23 100059 16
24 100064 11
25 100064 14
26 100064 17
27 100084 12
28 100086 14
29 100104 14
30 100109 14
31 100117 8
32 100117 12
33 100117 14
34 100119 14
35 1001287 15
36 100133 12
37 100133 17
38 100134 15
```

Рисунок 15 – Результат работы второго MapReduce

В соответствии с приведенным алгоритм поиска количества повторяющихся значений BibNumber входного массива данных «Записи выдачи книг в библиотеке Сиэтла» и вывести в выходной файл используя технологии MapReduce и HDFS. Третий процесс Map должен вывести значения атрибуты BibNumber и добавить к каждому значению 1. Реализация третьего этапа Map выполнена следующим образом:

```
#!/usr/bin/python
import sys
import csv
def mapper():
    reader = csv.reader(sys.stdin, delimiter=',')
    writer = csv.writer(sys.stdout, delimiter=',', quotechar='"',
quoting=csv.QUOTE_ALL)
    for line in reader:
        if len(line)!=6:
            continue
        # check bibnumber
        if str.isdigit(line[0]) == False:
            continue
        keyValPair = []
        keyValPair.append(line[0])
        keyValPair.append(1)
        writer.writerow(keyValPair)
def main():
    mapper()
    sys.stdin = sys.__stdin__
if __name__ == "__main__":
    main()
```

Третий процесс Reduce должен сделать поиск количества вхождений каждого ключа в атрибут BibNumber. Реализация второго этапа Reduce выполнена следующим образом:

```
#!/usr/bin/python
import sys
import csv
reader = csv.reader(sys.stdin, delimiter=',')
writer = csv.writer(sys.stdout, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL)
def outputCount(oldKey,countTotal):
    output = []
    output.append(oldKey)
    output.append(countTotal)
    writer.writerow(output)
countTotal = 0
oldKey = None
for data in reader:
    if len(data) != 2:
        continue
    thisKey = data[0]
    if oldKey and oldKey != thisKey:
        outputCount(oldKey,countTotal)
        oldKey = thisKey
        countTotal = 0
    oldKey = thisKey
    countTotal += 1
if oldKey != None:
    outputCount(oldKey,countTotal)
```

Входные значения mapper1.py задаются в командной строке и имеют следующий вид: -input /sema/ Checkouts\_By\_Title\_Data\_Lens\_2005.csv

Для запуска программы необходимо ввести следующую команду в терминале:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar  
-input /sema/Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output  
-mapper /home/hadoop/mapper2.py -reducer /home/hadoop/reducer2.py -file  
/home/hadoop/mapper2.py -file /home/hadoop/reducer2.py
```

```
hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -input /sema/  
Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output26 -mapper /home/hadoop/mapper2.py -reducer /home/hadoop/reducer2.py -file /h  
ome/hadoop/mapper2.py -file /home/hadoop/reducer2.py  
2022-01-20 03:44:26,852 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.  
packageJobJar: [/home/hadoop/mapper2.py, /home/hadoop/reducer2.py, /tmp/hadoop-unjar1778388107638864300/] [] /tmp/streamjob886339097189  
8225545.jar tmpDir=null  
2022-01-20 03:44:27,778 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032  
2022-01-20 03:44:27,920 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032  
2022-01-20 03:44:28,504 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging  
/job_1642630219480_0009  
2022-01-20 03:44:32,643 INFO mapred.FileInputFormat: Total input files to process : 1  
2022-01-20 03:44:32,653 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866  
2022-01-20 03:44:34,382 INFO mapreduce.JobSubmitter: number of splits:2  
2022-01-20 03:44:35,084 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642630219480_0009  
2022-01-20 03:44:35,084 INFO mapreduce.JobSubmitter: Executing with tokens: []  
2022-01-20 03:44:35,216 INFO conf.Configuration: resource-types.xml not found  
2022-01-20 03:44:35,216 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.  
2022-01-20 03:44:35,265 INFO impl.YarnClientImpl: Submitted application application_1642630219480_0009  
2022-01-20 03:44:35,294 INFO mapreduce.Job: The url to track the job: http://artemUbuntu:8088/proxy/application_1642630219480_0009/  
2022-01-20 03:44:35,295 INFO mapreduce.Job: Running job: job_1642630219480_0009  
2022-01-20 03:44:41,383 INFO mapreduce.Job: Job job_1642630219480_0009 running in uber mode : false  
2022-01-20 03:44:41,387 INFO mapreduce.Job: map 0% reduce 0%  
2022-01-20 03:44:54,514 INFO mapreduce.Job: map 50% reduce 0%  
2022-01-20 03:44:55,523 INFO mapreduce.Job: map 100% reduce 0%  
2022-01-20 03:45:03,580 INFO mapreduce.Job: map 100% reduce 100%  
2022-01-20 03:45:04,597 INFO mapreduce.Job: Job job_1642630219480_0009 completed successfully  
2022-01-20 03:45:04,686 INFO mapreduce.Job: Counters: 56
```

Рисунок 16 – Запуск третьей задачи MapReduce

```
Map output bytes=56569452  
Map output materialized bytes=64158916  
Input split bytes=232  
Combine input records=0  
Combine output records=0  
Reduce input groups=347711  
Reduce shuffle bytes=64158916  
Reduce input records=3794726  
Reduce output records=347711  
Spilled Records=7589452  
Shuffled Maps =2  
Failed Shuffles=0  
Merged Map outputs=2  
GC time elapsed (ms)=97  
CPU time spent (ms)=19770  
Physical memory (bytes) snapshot=1152811008  
Virtual memory (bytes) snapshot=8221089792  
Total committed heap usage (bytes)=792723456  
Peak Map Physical memory (bytes)=394457088  
Peak Map Virtual memory (bytes)=2738909184  
Peak Reduce Physical memory (bytes)=367288320  
Peak Reduce Virtual memory (bytes)=2744905728  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
File Input Format Counters  
Bytes Read=273090246  
File Output Format Counters  
Bytes Written=5215693  
2022-01-20 03:45:04,686 INFO streaming.StreamJob: Output directory: /sout/output26
```

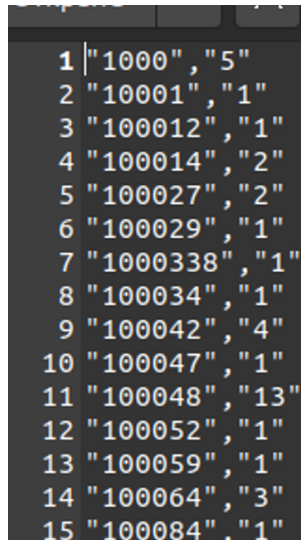
Рисунок 17 – Запуск третьей задачи MapReduce

Выходные значения метода reducer2.py представляет собой массив данных. Для того чтобы отобразить данные, необходимо выполнить следующую команду (аналогичная команда, как и для первой задачи, только output необходимо указать другой):

```
2022-01-20 03:45:04,686 INFO streaming.StreamJob: Output directory: /sout/output26
hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop fs -getmerge /sout/output26/part-00000 /home/hadoop/fileOutput
hadoop@artemUbuntu: /usr/local/hadoop/sbin$
```

Рисунок 18 – Выгрузка данных

Результаты выполнения третьего MapReduce:



```
1 |"1000", "5"
2 "10001", "1"
3 "100012", "1"
4 "100014", "2"
5 "100027", "2"
6 "100029", "1"
7 "1000338", "1"
8 "100034", "1"
9 "100042", "4"
10 "100047", "1"
11 "100048", "13"
12 "100052", "1"
13 "100059", "1"
14 "100064", "3"
15 "100084", "1"
```

Рисунок 19 – Результат работы третьего MapReduce

### 3 ПОЛУЧЕНИЕ И ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ

#### 3.1 Описание компиляции и запуска разработанного приложения

Разработанная программа представлена в виде файлов формата .py. Для компиляции приложения используется инструмент Hadoop Streaming [2]. Процесс компиляции осуществляется следующим образом:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar
-input /sema/Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output -mapper
/home/hadoop/map.py -reducer /home/hadoop/reducer.py -file home/hadoop/map.py
-file /home/hadoop/reducer.py
```

На рисунке ниже представлен результат успешной компиляции программы.

```

hadoop@artemUbuntu: /usr/local/hadoop/sbin$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -input /sema/
Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output18 -mapper /home/hadoop/map.py -reducer /home/hadoop/reducer.py -file /home/h
2022-01-19 20:47:44,697 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/hadoop/map.py, /home/hadoop/reducer.py, /tmp/hadoop-unjar14955659657540461026/] [] /tmp/streamjob1687789617126558
954.jar tmpDir=null
2022-01-19 20:47:45,392 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-01-19 20:47:45,510 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-01-19 20:47:46,100 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging
/job_1642587955476_0006
2022-01-19 20:47:50,664 INFO mapred.FileInputFormat: Total input files to process : 1
2022-01-19 20:47:50,673 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2022-01-19 20:47:53,465 INFO mapreduce.JobSubmitter: number of splits:2
2022-01-19 20:47:54,417 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642587955476_0006
2022-01-19 20:47:54,417 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-01-19 20:47:54,552 INFO conf.Configuration: resource-types.xml not found
2022-01-19 20:47:54,553 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-01-19 20:47:54,590 INFO impl.YarnClientImpl: Submitted application application_1642587955476_0006
2022-01-19 20:47:54,612 INFO mapreduce.Job: The url to track the job: http://artemUbuntu:8088/proxy/application_1642587955476_0006/
2022-01-19 20:47:54,613 INFO mapreduce.Job: Running job: job_1642587955476_0006
2022-01-19 20:47:59,661 INFO mapreduce.Job: Job job_1642587955476_0006 running in uber mode : false
2022-01-19 20:47:59,662 INFO mapreduce.Job: map 0% reduce 0%
2022-01-19 20:48:15,762 INFO mapreduce.Job: map 17% reduce 0%
2022-01-19 20:48:17,773 INFO mapreduce.Job: map 33% reduce 0%
2022-01-19 20:48:21,786 INFO mapreduce.Job: map 42% reduce 0%
2022-01-19 20:48:23,794 INFO mapreduce.Job: map 51% reduce 0%
2022-01-19 20:48:27,808 INFO mapreduce.Job: map 59% reduce 0%
2022-01-19 20:48:29,825 INFO mapreduce.Job: map 83% reduce 0%
2022-01-19 20:48:31,836 INFO mapreduce.Job: map 100% reduce 0%
2022-01-19 20:48:42,886 INFO mapreduce.Job: map 100% reduce 100%
2022-01-19 20:48:49,956 INFO mapreduce.Job: Job job_1642587955476_0006 completed successfully

```

Рисунок 20 – Успешный запуск программы

Для запуск программы так же использовались средства командной строки Ubuntu 20.04, представленные ниже:

```

hadoop@artemUbuntu:~$ cat Checkouts_By_Title_Data_Lens_2005.csv | python mapper1
.py | python reducer1.py

```

Рисунок 21 – Запуск программы без mapreduce

```

hadoop@artemUbuntu:~$ cat Checkouts_By_Title_Data_Lens_2005.csv | python map.py | python reducer.py

```

Рисунок 22 – Запуск программы без mapreduce

Результаты запуска программы с использованием средств командной строки Ubuntu 20.04:

```

hadoop@artemUbuntu:~$ cat Checkouts_By_Title_Data_Lens_2005.csv | python mapper1.py | python reducer1.py
ItemType      1
acbk          1376400
jcbk           742786
jcvhs         102217
accd           527068
jcdvd         109408
acdvd         439384
acvhs         303958
jccas         21369
jccd          43901
accas         61764
ucfold        12012
dcillb        17509
bcbk           4052
acfold        2439
acmus          8520
arb           1795
alvhs         3309
jckit         5879
bckit          496
ucunknj       1287
acdrom         845
ackit          211
jccdrum       1942
bccas         1578
arper          47
acvid         105
bcvhs         1547
bccd          238
arcd           47
acrec         547
ardvd          2
jcmus         276
ucfoldr       18

```

Рисунок 23 – Отображение результата выполнения команды

Для успешного запуска этапов MapReduce необходимо запустить сам Hadoop, как показано на рисунке ниже [3]:

```

hadoop@artemUbuntu:~$ cd /usr
hadoop@artemUbuntu:/usr$ cd local
hadoop@artemUbuntu:/usr/local$ ls
bin  etc  games  hadoop  include  lib  libexec  man  sbin  share  src
hadoop@artemUbuntu:/usr/local$ cd hadoop
hadoop@artemUbuntu:/usr/local/hadoop$ ls
bin  include  libexec  licenses-binary  logs  NOTICE.txt  sbin
etc  lib  LICENSE-binary  LICENSE.txt  NOTICE-binary  README.txt  share
hadoop@artemUbuntu:/usr/local/hadoop$ cd sbin
hadoop@artemUbuntu:/usr/local/hadoop/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [artemUbuntu]
hadoop@artemUbuntu:/usr/local/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@artemUbuntu:/usr/local/hadoop/sbin$ jps
4754 ResourceManager
4196 DataNode
4422 SecondaryNameNode
4042 NameNode
5292 Jps

```

Рисунок 24 – Запуск Hadoop

После завершения всех исследований необходимо остановить Hadoop, для этого в командной строке необходимо прописать следующие команды:

```

hadoop@artemUbuntu:/usr/local/hadoop/sbin$ ./stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [artemUbuntu]
hadoop@artemUbuntu:/usr/local/hadoop/sbin$ ./stop-yarn.sh
Stopping nodemanagers
localhost: WARNING: nodemanager did not stop gracefully after 5 seconds
Stopping resourcemanager
hadoop@artemUbuntu:/usr/local/hadoop/sbin$ █

```

Рисунок 25 – остановка работы Hadoop

### 3.2 Интерпретация результатов

Сравним результаты, полученные в ходе выполнения первого MapReduce и статистику наибольших повторяющихся значений атрибута ItemType в датасете

## A ItemType

acbk	36%	Valid <span style="color: green;">■</span>	3.79m	100%
		Mismatched <span style="color: orange;">■</span>	0	0%
jcbk	20%	Missing <span style="color: red;">■</span>	0	0%
Other (1675501)	44%	Unique	59	
		Most Common	acbk	36%

Рисунок 26 – атрибут ItemType

acbk	1376400
jcbk	742786
jcvhs	102217
accd	527068
jcdvd	109408
acdvd	439384
acvhs	303958
jccas	21369
jccd	43901
accas	61764
ucfold	12012
dcillb	17509
bcbk	4052
acfold	2439

Рисунок 27 – результаты первого mapreduce

Вывод: по статистике датасета на сайте kaggle наиболее встречающиеся значения атрибута ItemType является acbk, после выполнения первого mapreduce можно увидеть, что самое большое количество встречающихся значений в атрибуте является acbk равное 1376400.

### 3.3 Исследование масштабируемости приложения анализа данных

Для исследования масштабируемости приложения был проведен ряд его запусков с различным количеством процессов map и reduce с замером времени исполнения. Запуск приложения осуществлялся следующим образом:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar  
-input /sema/Checkouts_By_Title_Data_Lens_2005.csv -output /sout/output -mapper  
/home/hadoop/map.py -reducer /home/hadoop/reducer.py -file home/hadoop/map.py  
-file /home/hadoop/reducer.py
```



Время работы приложения измерялось средствами командной строки при запуске выше указанной команды. Отображение запуска и окончания можно посмотреть на первом рисунке 20, оно показано слева по горизонтали. Результаты исследования приведены в таблице 1.

Таблица 1 — Результаты исследования масштабируемости приложения

Количество процессов map	Количество процессов reduce	Время работы приложения $T$ , сек
1 (Map.py)	1 (Reducer.py)	65
1 (Mapper1.py)	1 (Reducer1.py)	40
1 (Mapper2.py)	1 (Reducer2.py)	38

### 3.4 Выводы

Приведенные результаты позволяют сделать следующие выводы:

- Программа успешно запускается,
- Показана интерпретация результатов,
- Проведено исследование масштабируемости приложения анализа данных.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было выполнено три этапа MapReduce с использованием технологии Hadoop MapReduce и распределенной файловой системой HDFS. Получены следующие результаты:

- Построен алгоритм mapreduce,
- Программно реализован алгоритм mapreduce,
- Результаты выполнения программы занесены в выходной файл, для проверки результативности выполненной работы,
- Алгоритм реализован согласно требованию работы, с использованием выше указанных технологий.

Результаты работы могут быть использованы для дальнейшей работы над проектом для построения иных задач с использованием технологий mapreduce.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Уайт Т. Hadoop: Подробное руководство. СПб. : Питер, 2013. 672 с.  
(дата обращения 19.01.2022)
- 2) Hadoop Streaming [Электронный ресурс] Режим доступа:  
<https://hadoop.apache.org/docs/r1.2.1/streaming.html> (дата обращения : 19.01.2022)
- 3) Hadoop Architecture Guide [Электронный ресурс] Режим доступа:  
[https://hadoop.apache.org/docs/r1.2.1/hdfs\\_design.html](https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html) (дата обращения :  
19.01.2022)
- 4) Seattle Library Checkout Records [Электронный ресурс] Режим  
доступа: <https://www.kaggle.com/seattle-public-library/seattle-library-checkout-records> (дата обращения : 19.01.2022)