Национальный исследовательский ядерный университет “МИФИ”

Институт кибернетических интеллектуальных систем

Лабораторная работа №1

“Hadoop”

По курсу: «Наука о данных и анализ больших данных»

Выполнила студентка группы М20-502:

Мохорт Елизавета

Преподаватель:

Ровнягин М.М.

Москва, 2021

**Задание:**

a) Бизнес логика:

2. Программа, которая агрегирует сырые метрики в выбранный диапазон:

Входной формат: metricId, timestamp (millis), value (справочник обозначений: metricId - metricName)

Выходной формат: metricId, timestamp, scale, value

b) Output Format:

3. SequenceFile со Snappy сжатием (плюс команда просмотра содержимого сжатого файла посредством распаковки). Приложить скриншот просмотра сжатого контента.

**Ход работы:**

После установки и настройки конфигураций hadoop для работы с проектами в распределенном режиме необходимо в директории hadoop:

* Запустить файловую систему:

/opt/hadoop-2.10.1/sbin/start-dfs.sh

* Запустить yarn:

/opt/hadoop-2.10.1/sbin/start-yarn.sh

* Проверить запущенные узлы командой jps, в случае, если узлы не запустились:

1. Остановить демоны

/opt/hadoop-2.10.1/sbin/stop-all.sh

1. Отформатировать файловую систему

hdfs namenode –format

(если не запускается DataNode, то перед форматированием необходимо очистить каталог /tmp/hadoop-<user> командой «sudo rm -rf tmp/hadoop-root»)

1. Запустить демоны снова

/opt/hadoop-2.10.1/sbin/start-all.sh

1. Создать домашние директории в hadoop

hdfs dfs -mkdir /user

* Cкачать папку проекта с репозитория GitHub:

git clone <https://github.com/Mokhort/DSBDA_HW1>

* Перейти в папку проекта и сгенерировать входной файл:

./generateInput.sh test.txt

* Если, тесты уже проводились, почистить файловую систему Hadoop от входных и выходных данных:

hdfs dfs -rm -r input

hdfs dfs -rm -r output

* Поместить сгенерированный входной файл в папку распределенной файловой системы:

hdfs dfs -put input input

* Запустить job’у

yarn jar target/lab1-1.0-SNAPSHOT-jar-with-dependencies.jar input output

* Забрать с файловой системы hadoop выходной SequenceFile со Snappy сжатием и посмотреть в консоли:

hadoop fs -text output/part-r-00000 > out.txt

cat out.txt

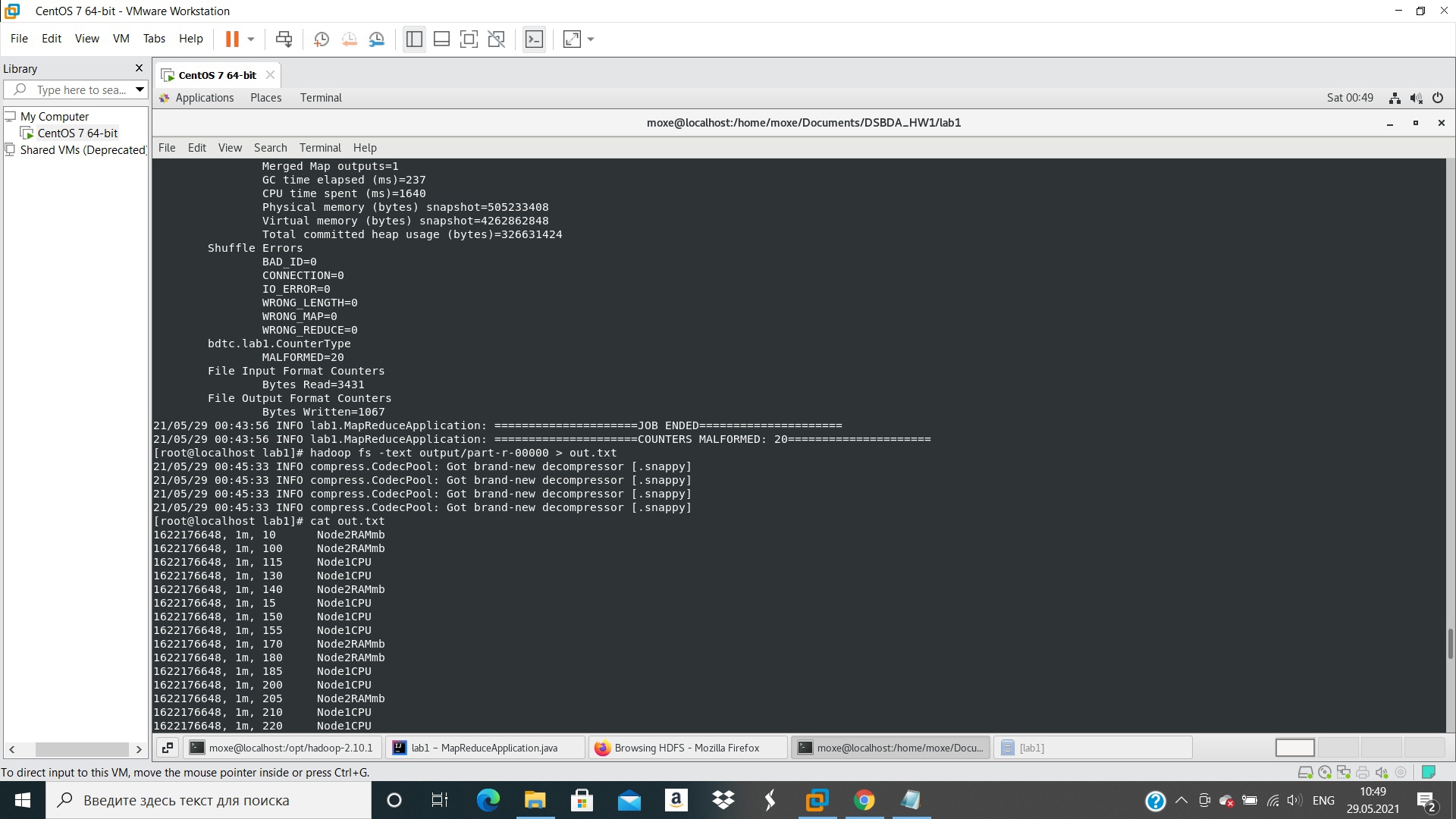


Рисунок 1 –Скриншот результата работы программы(использование счетчиков и просмотр выходного файла)

**Принцип работы программы:**

Основная работа разделяется между Mapper и Reducer:

* Mapper построчно обрабатывает входные файлы, с помощью счетчиков определяет количество ошибочных строк, отделяет ключ в каждой строке и приписывает каждому ключу значение;
* Reducer принимает пары ключ-значение, агрегирует по ключу метрики и формирует строку в выходной файл
* Для отладки работы программы были написаны Unit test’ы:

MapTest, ReduceTest, MapReduceTest, CounterTest

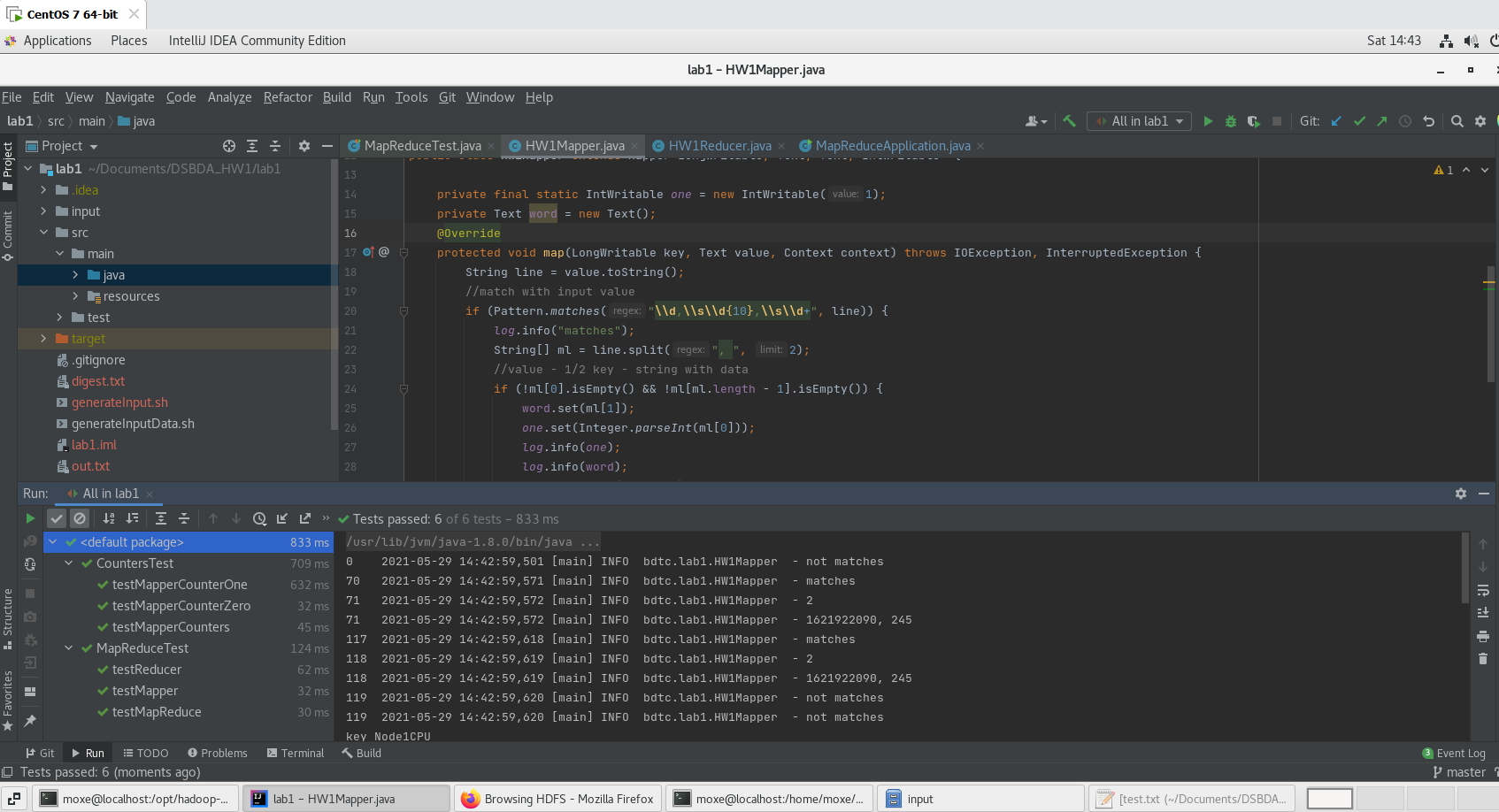


Рисунок 2 – Скриншот успешного выполнения Unit test’ов

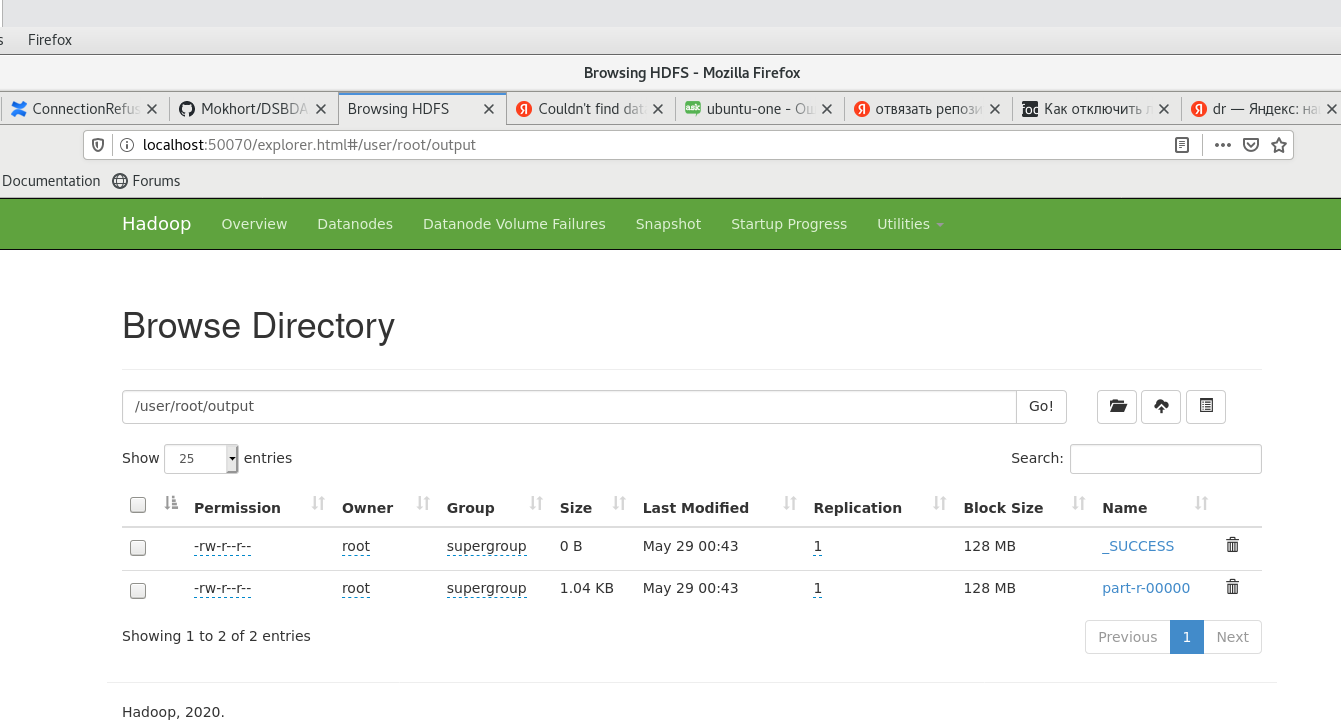


Рисунок 3 – Скриншот распределенной файловой системы после работы программы (выходной файл)

**Листинг файла генерации входных данных:**

#!/bin/bash

COUNTER=0

if [[ $# -eq 0 ]] ; then

echo 'You should specify output file!'

exit 1

fi

rm -rf input

mkdir input

for i in {1..200}

do

COUNTER=$(($COUNTER + 1))

COUNTER\_OST=$(($COUNTER %10))

ID=$((1 + RANDOM % 2))

CURRENTTIME=`date +"%s"`

RAND=$((1 + RANDOM % 150))

CURRENTTIME=$(($CURRENTTIME +$((1 + RANDOM % 4))))

VALUE=$((1+ RANDOM %300))

OST=$(($VALUE %5))

if [ $COUNTER\_OST -eq 0 ]

then

echo ${COUNTER} >> input/$1.1

elif [ $OST -eq 0 ]

then

echo ${ID}, ${CURRENTTIME}, ${VALUE} >> input/$1.1

else

VALUE=$(($VALUE +5 - $OST))

echo ${ID}, ${CURRENTTIME}, ${VALUE} >> input/$1.1

fi

done

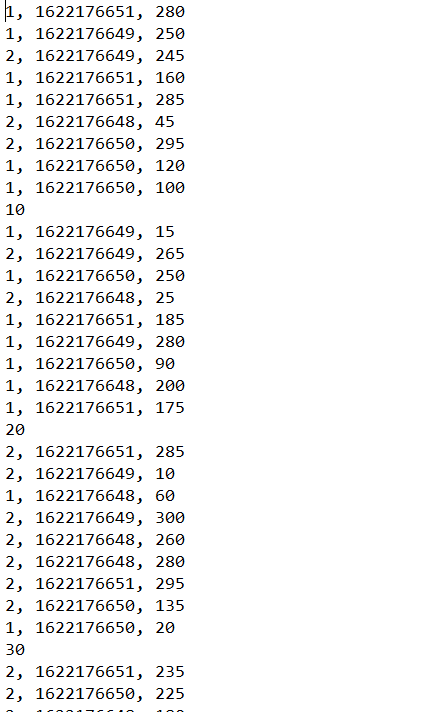


Рисунок 4 – Скриншот содержимого входного файла

Программный код всего проекта находится в репозитории GitHub <https://github.com/Mokhort/DSBDA_HW1>

**Вывод:** В ходе лабораторной работы была изучена система Hadoop для работы с данными в распределенном режиме и реализована программа в соответствии с вариантом задания.