|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторным работам №** | 7-8 |

**Название:** Работа с Hadoop и Spark

**Дисциплина:** Технология параллельных систем баз данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-12М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | |  |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2023

**Введение**

**1. Цель работы:** приобретение навыков инсталляции и работы с продуктами Apache Hadoop и Apache Spark, поддерживающих технологию MapReduce. Эти системы используются для обработки больших данных (Big Data).

**Ход выполнения**

**2. Установка Hadoop и Spark.**

Для упрощения процесса установки воспользуемся docker-контейнерами с hadoop и spark. Склонируем репозиторий с описанием конфигурации docker-compose “git clone git@github.com:Marcel-Jan/docker-hadoop-spark.git”. Конфигурация системы контенеров приведена в листинге 1 (некоторые контейнеры, такие как hive server, опущены для краткости, т.к. не используются в данной работе).

Листинг 1 — файл docker-compose.yml

|  |
| --- |
| version: "3"  services:  namenode:  image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8  container\_name: namenode  restart: always  ports:  - 9870:9870  - 9010:9000  volumes:  - hadoop\_namenode:/hadoop/dfs/name  environment:  - CLUSTER\_NAME=test  - CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000  env\_file:  - ./hadoop.env  datanode:  image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8  container\_name: datanode  restart: always  volumes:  - hadoop\_datanode:/hadoop/dfs/data  environment:  SERVICE\_PRECONDITION: "namenode:9870"  CORE\_CONF\_fs\_defaultFS: hdfs://namenode:9000  ports:  - "9864:9864"  env\_file:  - ./hadoop.env  resourcemanager:  image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8  container\_name: resourcemanager  restart: always  environment:  SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"  env\_file:  - ./hadoop.env  nodemanager1:  image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8  container\_name: nodemanager  restart: always  environment:  SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"  env\_file:  - ./hadoop.env  spark-master:  image: bde2020/spark-master:3.0.0-hadoop3.2  container\_name: spark-master  depends\_on:  - namenode  - datanode  ports:  - "8080:8080"  - "7077:7077"  environment:  - INIT\_DAEMON\_STEP=setup\_spark  - CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000  spark-worker-1:  image: bde2020/spark-worker:3.0.0-hadoop3.2  container\_name: spark-worker-1  depends\_on:  - spark-master  ports:  - "8081:8081"  environment:  - "SPARK\_MASTER=spark://spark-master:7077"  - CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000  volumes:  hadoop\_namenode:  hadoop\_datanode:  hadoop\_historyserver: |

Данная конфигурация описывает соответствие портов на хост-машине и внутри виртуальной сети docker, используемые носители, а так же дополнительные параметры для контейнеров. Назначение основных контенейров в конфигурации:

* namenode — хранит имена файлов и расположение их сегментов;
* datanode — хранит сегменты файлов;
* resource manager и node manager — управляют ресурсами всего кластера и отдельных узлов соответственно;
* spark-master — управляет задачами spark;
* spark-worker — выполняет задачи spark.

Перейдя в папку с файлом конфигурации, запустим контейнеры командой “docker-compose up” (рисунок 1). Примечание: некоторые контейнеры не уместились на одном скриншоте.

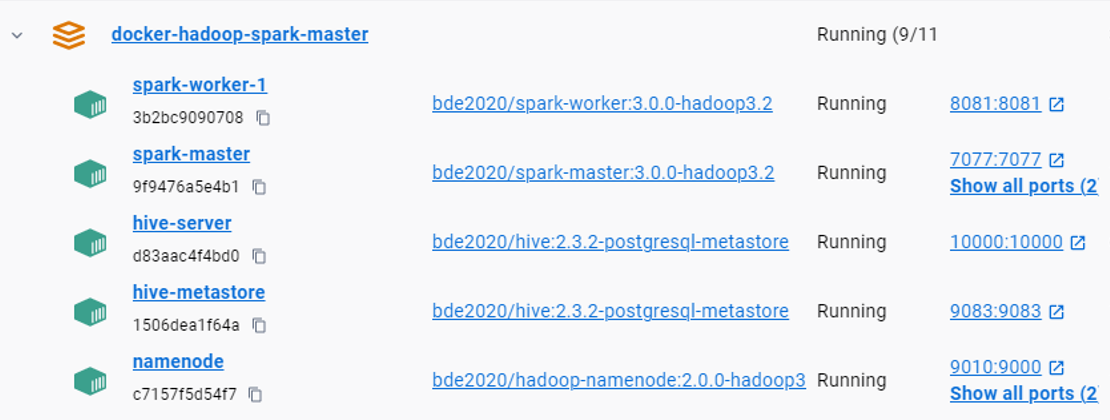


Рисунок 1 — запущенные контейнеры

Проверим работу hadoop с помощью команды jps (рисунок 2). Hadoop работает корректно.

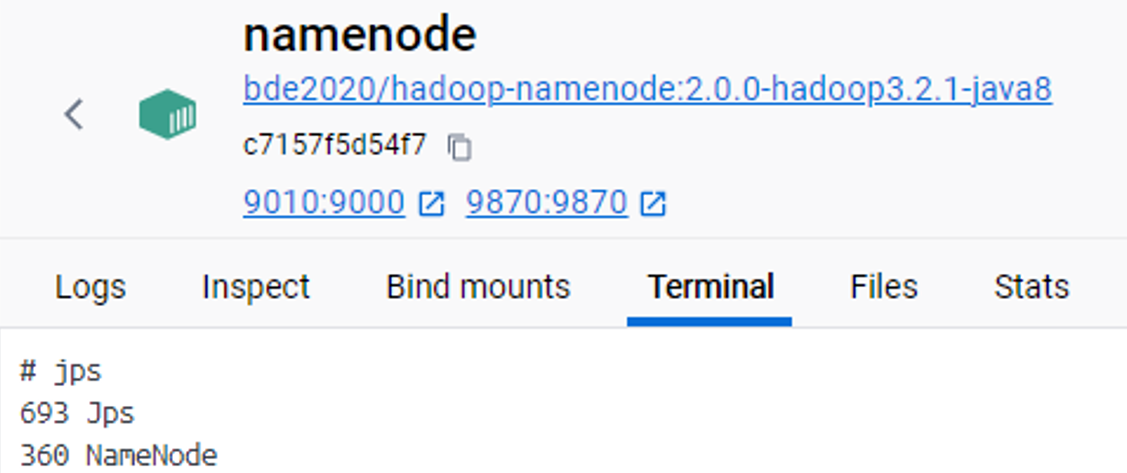


Рисунок 2 — проверка работы hadoop

Проверим работу веб-интерфейсов hadoop и spark (рисунки 3, 4).

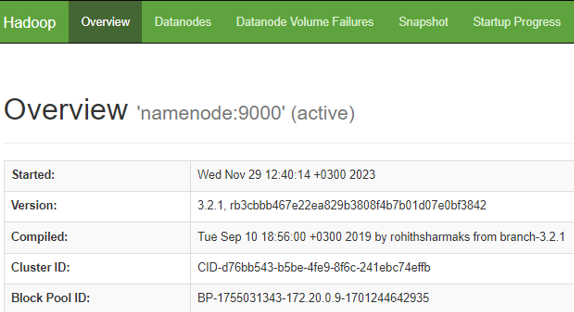


Рисунок 3 — веб-интерфейс hadoop

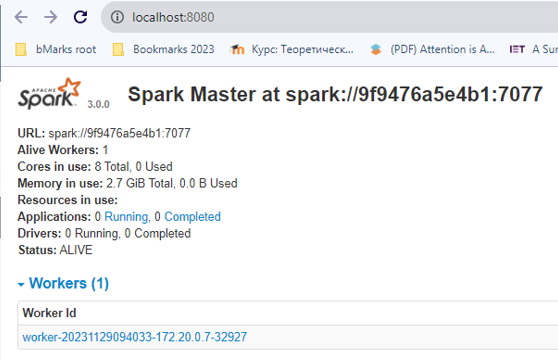


Рисунок 4 — веб-интерфейс spark

**3. Работа с hadoop и spark**

В консоли контейнера spark-master запустим pyspark и создадим датафрейм из пар чисел с их названиями (рисунок 5).

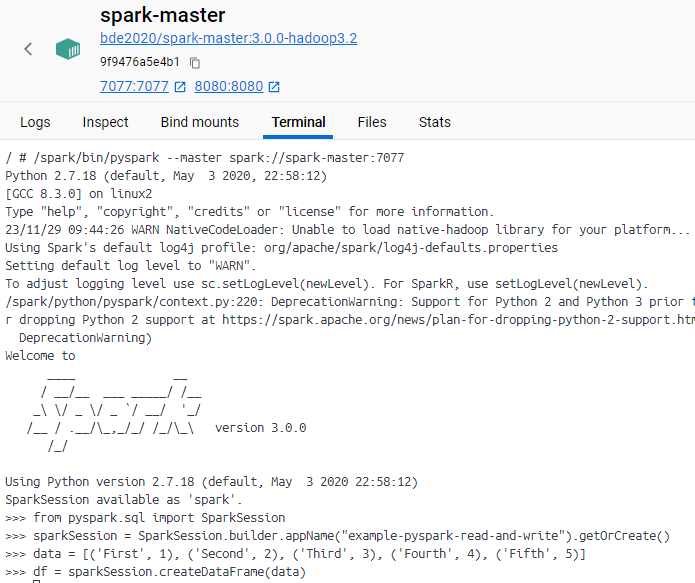


Рисунок 5 — создание датафрейма в pyspark

После этого сохраним датафрейм командой df.write.csv("hdfs://namenode:9000/chapter5/example.csv"). Прочитаем и отобразим датафрейм, убеждаясь в корректной работе pyspark (рисунок 6).

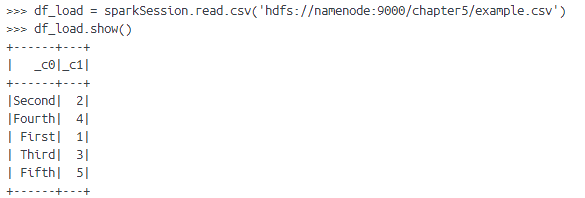


Рисунок 6 — чтение и отображение датафрейма

Выйдем из консоли pyspark, зайдем в консоль scala и выйдем из нее (рисунок 7).

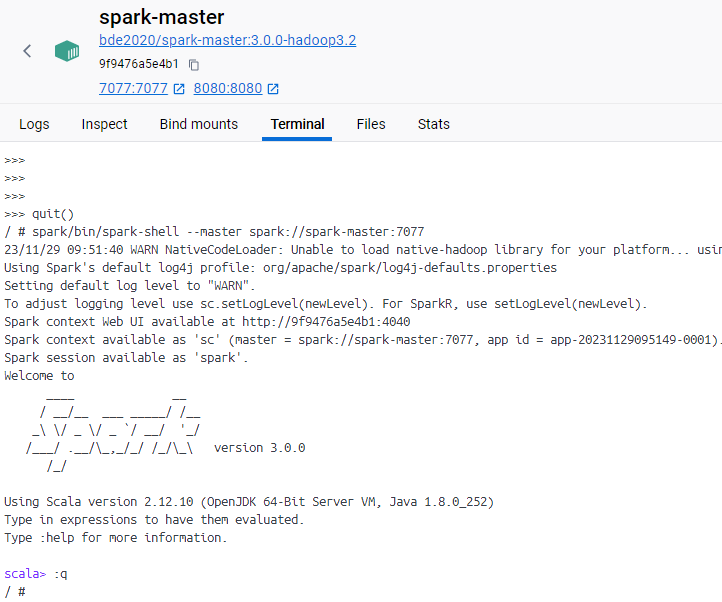


Рисунок 7 — проверка консоли scala

Скопируем файл с пьесой «Гамлет» с хост-машины в контейнер (рисунок 8).

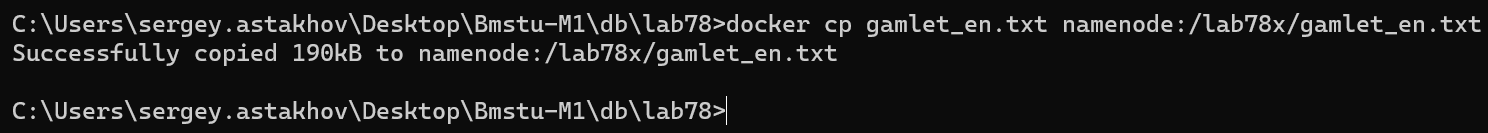


Рисунок 8 — копирование файла в контейнер

Внутри контейнера скопируем файл в hdfs и проверим его наличие в hdfs (рисунок 9).

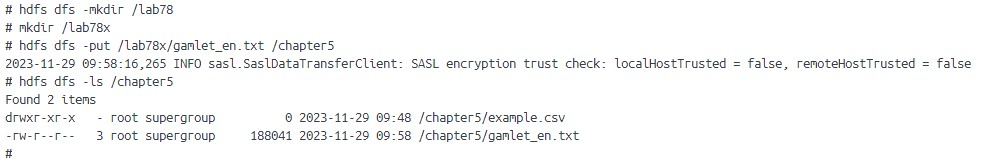


Рисунок 9 — копирование файла в hdfs

Войдем в консоль pyspark и выполним скрипт подсчета слов (рисунок 10). Программа выполняет следующие шаги:

* импорт библиотек, чтение файла из hdfs;
* разбитие файла на слова (метод flatMap);
* каждому слову ставится в соответствие значение 1 (метод map);
* для каждого уникального слова-ключа значения складываются между собой (reduceByKey);
* сохранение расчетов в файл;
* сумма повторений всех слов рассчитывается и выводится на экран.

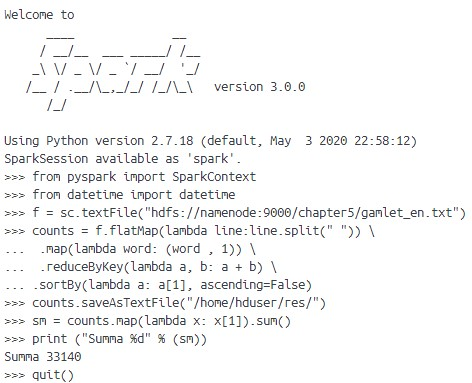


Рисунок 10 — подсчет слов

На рисунке 11 показано содержимое RDD counts.

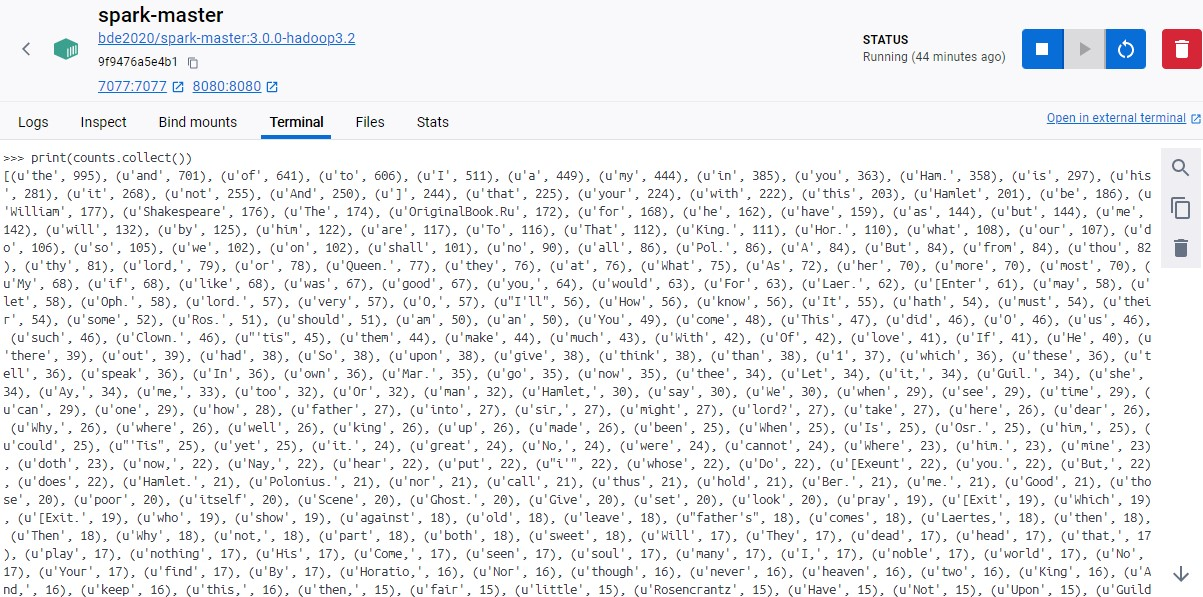


Рисунок 11 — содержимое counts

**Вывод:** в ходе работы были приобретены навыки инсталляции и работы с продуктами Apache Hadoop и Apache Spark, поддерживающих технологию MapReduce. Эти системы используются для обработки больших данных (Big Data).