

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждения высшего
образования города Москвы «Московский городской педагогический
университет»

Институт цифрового образования
Департамент информатики, управления и технологий

Лабораторная работа 1.
Установка и настройка распределенной системы. Простейшие операции и
знакомство с функциональностью системы.

Выполнил студент группы АДЭУ-221
Джамалова Сабина Шахиновна
Проверил доцент
Босенко Тимур Муртазович

Москва
2024

Цель работы: ознакомление с процессом установки и настройки распределенных систем, таких как Apache(Arenadata) Hadoop или Apache Spark. Изучить основные операции и функциональные возможности системы, что позволит понять принципы работы с данными и распределенными вычислениями.

Задание:

6. Выполнение задачи на агрегацию данных. Данные: Исторические данные по акциям Норильского никеля (GMKN) с сайта Московской биржи (moex.com)

Выполнение:

Apache Spark

1. Загрузка данных:

```
hadoop@devopsvm:~$ wget https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/refs/heads/main/practice/2024/lw_01/AAPL.csv
--2024-10-19 00:00:15-- https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/refs/heads/main/practice/2024/lw_01/AAPL.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 747952 (730K) [text/plain]
Saving to: 'AAPL.csv'
```

```
AAPL.csv          100%[=====] 730.42K  2.01MB/s   in 0.4s
```

```
2024-10-19 00:00:16 (2.01 MB/s) - 'AAPL.csv' saved [747952/747952]
```

2. Перенос в каталог:

```
hadoop@devopsvm:~$ hdfs dfs -mkdir /user/hadoop/spark_data
2024-10-19 00:03:43,692 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$ hdfs dfs -put AAPL.csv /user/hadoop/spark_data
2024-10-19 00:03:53,210 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@devopsvm:~$
```

3. Запуск Spark и выполнение простейших операций:

```
hadoop@devopsvm:~$ pyspark
Python 3.12.3 (main, Jul 31 2024, 17:43:48) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
24/10/18 23:51:51 WARN Utils: Your hostname, devopsvm resolves to a loopback address: 127.0.1.1; using 172.20.10.3 instead (on interface enp0s3)
24/10/18 23:51:51 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/18 23:51:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to

  ____      _
 / ___|    / \
| |  | |  / _ \
| |  | | / ___ \
| |  | || |___) |
| |  | || |___) |
|_|  |_| \____/

version 3.4.3

Using Python version 3.12.3 (main, Jul 31 2024 17:43:48)
Spark context Web UI available at http://172.20.10.3:4040
Spark context available as 'sc' (master = local[*], app id = local-1729284718110).
SparkSession available as 'spark'.
>>>
```

4. Загрузка данных:

```
df = spark.read.csv("file:///home/username/spark_data/AAPL.csv", header=True, inferSchema=True)
```

```
df.show(5)
```

```
>>> df = spark.read.csv("hdfs://localhost:9000/user/hadoop/spark_data/AAPL.csv", header=True, inferSchema=True)
>>> df.show(5)
+-----+-----+-----+-----+-----+-----+-----+
|   Date|   Open|   High|   Low|   Close|Adj Close|  Volume|
+-----+-----+-----+-----+-----+-----+-----+
|1980-12-12|0.128348|0.128906|0.128348|0.128348| 0.098943|469033600|
|1980-12-15| 0.12221|0.12221|0.121652|0.121652| 0.093781|175884800|
|1980-12-16|0.113281|0.113281|0.112723|0.112723| 0.086898|105728000|
|1980-12-17|0.115513|0.116071|0.115513|0.115513| 0.089049| 86441600|
|1980-12-18|0.118862| 0.11942|0.118862|0.118862| 0.09163| 73449600|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

5. Подсчет количества строк:

```
print("Количество строк:", df.count())
```

```
>>> print("Количество строк:", df.count())
Количество строк: 11017
```

6. Вывод схемы данных:

```
df.printSchema()
```

```
>>> df.printSchema()
root
 |-- Date: date (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Adj Close: double (nullable = true)
 |-- Volume: long (nullable = true)

>>> █
```

7. Базовая статистика:

```
df.describe().show()
```

```
>>> df.describe().show()
24/10/19 00:20:11 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+-----+-----+-----+-----+-----+-----+-----+
|summary|   Open|   High|   Low|   Close|Adj Close|  Volume|
+-----+-----+-----+-----+-----+-----+-----+
| count|   11017|   11017|   11017|   11017|   11017|   11017|
| mean|22.63869128065717|22.883548980121645|22.485040575837347|22.654083055913596|21.837056018244503|3.1761971054651904E8|
| stddev|46.68552390873815|47.190141187907436| 46.21839372274598|46.726038078492074| 46.26347693183877|3.3534880876140475E8|
| min| 0.049665| 0.049665| 0.049107| 0.049107| 0.037856| 0|
| max| 236.479996| 237.229996| 233.089996| 234.820007| 234.548523| 7421640800|
+-----+-----+-----+-----+-----+-----+-----+
>>>
```

8. Фильтрация данных:

```
df_filtered = df.filter(df["Date"] >= "2020-01-01")
```

```
df_filtered.show(5)
```

```
>>> df_filtered = df.filter(df["Date"] >= "2020-01-01")
>>> df_filtered.show(5)
+-----+-----+-----+-----+-----+-----+-----+
|      Date|      Open|      High|      Low|      Close|Adj Close|      Volume|
+-----+-----+-----+-----+-----+-----+-----+
|2020-01-02|74.059998|75.150002|73.797501|75.087502|72.876091|135480400|
|2020-01-03|74.287498|75.144997| 74.125|74.357498|72.167587|146322800|
|2020-01-06|73.447502|74.989998| 73.1875|74.949997|72.742661|118387200|
|2020-01-07|74.959999|75.224998|74.370003|74.597504|72.400551|108872000|
|2020-01-08|74.290001|76.110001|74.290001|75.797501|73.565208|132079200|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

9. Группировка и агрегация:

```
from pyspark.sql.functions import year, avg
```

```
df_yearly = df.withColumn("Year",
```

```
year(df["Date"])).groupBy("Year").agg(avg("Close").alias("Avg_Close"))
```

```
df_yearly.orderBy("Year").show()
```

```
>>> from pyspark.sql.functions import year, avg
>>> df_yearly = df.withColumn("Year", year(df["Date"])).groupBy("Year").agg(avg("Close").alias("Avg_Close"))
>>> df_yearly.orderBy("Year").show()

+-----+-----+
|Year|      Avg_Close|
+-----+-----+
|1980|0.13590307692307693|
|1981|0.10854781818181822|
|1982|0.08545888142292486|
|1983| 0.1672740118577075|
|1984| 0.1196512490118577|
|1985|0.09023348809523808|
|1986|0.14491285375494065|
|1987| 0.3477511106719368|
|1988| 0.3708842411067196|
|1989| 0.3719529682539681|
|1990|0.33537290513834017|
|1991|0.46870147826086955|
|1992| 0.4893094763779526|
|1993|0.36630964426877505|
|1994| 0.3042905634920634|
|1995|0.36196595238095214|
|1996|0.22249534645669278|
|1997| 0.1604176679841897|
|1998| 0.2729004801587301|
|1999| 0.515805492063492|
+-----+-----+
only showing top 20 rows
```

10. Создание временного представления и выполнение SQL-запроса:

```
df.createOrReplaceTempView("stock_data")
```

```
spark.sql("SELECT Year(Date) as Year, AVG(Close) as Avg_Close FROM  
stock_data GROUP BY Year(Date) ORDER BY Year").show()
```

```
>>> df.createOrReplaceTempView("stock_data")
>>> spark.sql("SELECT Year(Date) as Year, AVG(Close) as Avg_Close FROM stock_data GROUP BY Year(Date) ORDER BY  
+-----+-----+
|Year|      Avg_Close|
+-----+-----+
|1980|0.13590307692307693|
|1981|0.10854781818181822|
|1982|0.08545888142292486|
|1983| 0.1672740118577075|
|1984| 0.1196512490118577|
|1985|0.09023348809523808|
|1986|0.14491285375494065|
|1987| 0.3477511106719368|
|1988| 0.3708842411067196|
|1989| 0.3719529682539681|
|1990|0.33537290513834017|
|1991|0.46870147826086955|
|1992| 0.4893094763779526|
|1993|0.36630964426877505|
|1994| 0.3042905634920634|
|1995|0.36196595238095214|
|1996|0.22249534645669278|
|1997| 0.1604176679841897|
|1998| 0.2729004801587301|
|1999| 0.515805492063492|
+-----+-----+
only showing top 20 rows
```

11. Выход из Spark:

```
spark.stop()
```

```
exit()
```

```
>>>
>>> spark.stop()
>>> exit()
hadoop@devopsvm:~$
```

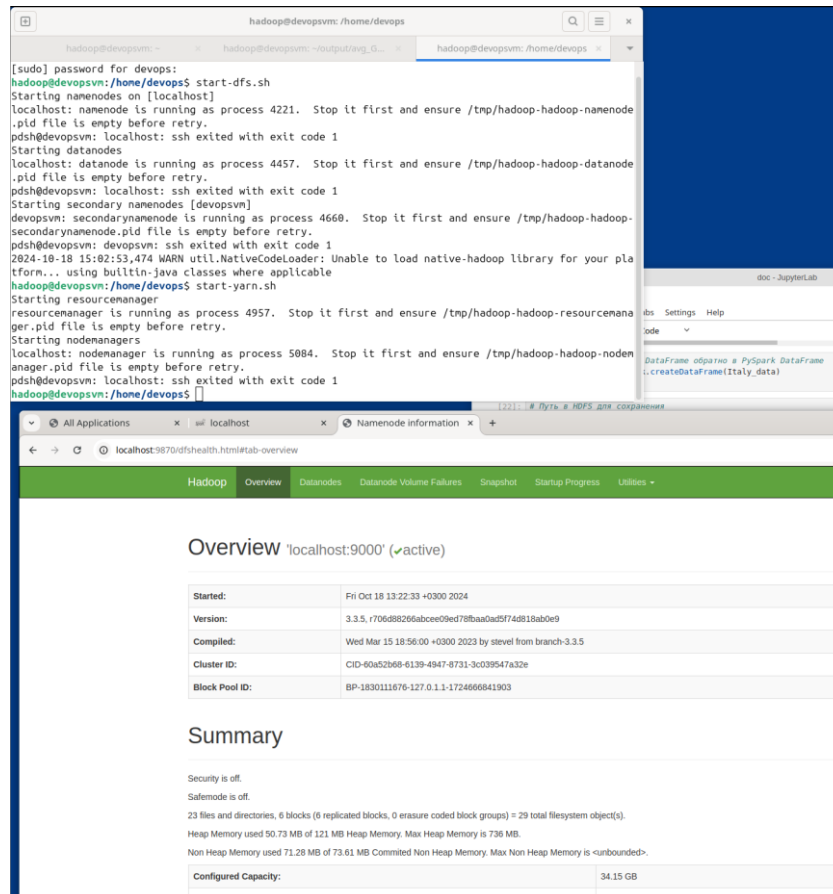
Вариант 6.

1. Запуск Hadoop:

`sudo su hadoop`

`start-dfs.sh`

`start-yarn.sh`



2. Проверка работы Hadoop:

`jps`

```
hadoop@devopsvm: /home/devops$ jps
8736 Jps
4660 SecondaryNameNode
4457 DataNode
5084 NodeManager
4957 ResourceManager
4221 NameNode
hadoop@devopsvm: /home/devops$
```












3. Проверка созданной директории:

`http://localhost:9870`

Browse Directory

Show entries

Search:

<input type="checkbox"/>	 Permission	 Owner	 Group	 Size	 Last Modified	 Replication	 Block Size	 Name	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Oct 18 14:26	0	0 B	economic_data	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Oct 18 13:55	0	0 B	input	

Showing 1 to 2 of 2 entries

Previous

1

Next

4. Подготовка данных

```

<TICKER>,<PER>,<DATE>,<TIME>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL>
GMKN,1,20200825,131300,19500.000000,19500.000000,19500.000000,19500.000000,1
GMKN,1,20200826,193400,20550.000000,20550.000000,20550.000000,20550.000000,1
GMKN,1,20200902,225600,20600.000000,20600.000000,20600.000000,20600.000000,4
GMKN,1,20200902,230500,20600.000000,20600.000000,20600.000000,20600.000000,10
GMKN,1,20200902,234200,20600.000000,20600.000000,20600.000000,20600.000000,5
GMKN,1,20200904,122400,19601.000000,19601.000000,19601.000000,19601.000000,1
GMKN,1,20200904,125700,21929.000000,21929.000000,21929.000000,21929.000000,1
GMKN,1,20200904,173000,21002.000000,21003.000000,19628.000000,19628.000000,4
GMKN,1,20200909,141600,19607.000000,19607.000000,19607.000000,19607.000000,1
GMKN,1,20200909,194400,21900.000000,21900.000000,21900.000000,21900.000000,1
GMKN,1,20200911,175100,19602.000000,19602.000000,19602.000000,19602.000000,1
GMKN,1,20200911,183500,21650.000000,21650.000000,21650.000000,21650.000000,1
GMKN,1,20200915,203600,19892.000000,19892.000000,19892.000000,19892.000000,2
GMKN,1,20200915,231800,20019.000000,20019.000000,20019.000000,20019.000000,4
GMKN,1,20200916,100700,19600.000000,19600.000000,19600.000000,19600.000000,1
GMKN,1,20200916,173000,19680.000000,19680.000000,19680.000000,19680.000000,3
GMKN,1,20200916,204800,19961.000000,19961.000000,19961.000000,19961.000000,1
GMKN,1,20200916,205800,19961.000000,19961.000000,19961.000000,19961.000000,2
GMKN,1,20200916,210300,19960.000000,19960.000000,19960.000000,19960.000000,1
GMKN,1,20200917,121900,19760.000000,19760.000000,19760.000000,19760.000000,1
GMKN,1,20200917,123400,19451.000000,19451.000000,19451.000000,19451.000000,1
GMKN,1,20200917,135600,19450.000000,19450.000000,19450.000000,19450.000000,2
GMKN,1,20200917,181100,19666.000000,19666.000000,19666.000000,19666.000000,1
GMKN,1,20200917,181300,19668.000000,19668.000000,19668.000000,19668.000000,3
GMKN,1,20200921,114800,19450.000000,19450.000000,19450.000000,19450.000000,3
GMKN,1,20200921,115400,19450.000000,19450.000000,19450.000000,19450.000000,6
GMKN,1,20200921,120000,19350.000000,19350.000000,19350.000000,19350.000000,2
GMKN,1,20200921,120100,19350.000000,19350.000000,19350.000000,19350.000000,1
GMKN,1,20200921,121700,19350.000000,19350.000000,19350.000000,19350.000000,3
GMKN,1,20200921,123300,19350.000000,19350.000000,19350.000000,19350.000000,1
GMKN,1,20200921,124100,19694.000000,19694.000000,19694.000000,19694.000000,1
GMKN,1,20200921,125700,19694.000000,19694.000000,19694.000000,19694.000000,1
GMKN,1,20200921,133500,19350.000000,19350.000000,19350.000000,19350.000000,2
GMKN,1,20200921,133600,19350.000000,19350.000000,19350.000000,19350.000000,1
GMKN,1,20200921,134000,19200.000000,19200.000000,19200.000000,19200.000000,1
GMKN,1,20200921,152300,19255.000000,19255.000000,19255.000000,19255.000000,1
GMKN,1,20200921,155600,19255.000000,19255.000000,19255.000000,19255.000000,1
GMKN,1,20200921,164800,19200.000000,19200.000000,19200.000000,19200.000000,1
GMKN,1,20200921,173900,19200.000000,19200.000000,19200.000000,19200.000000,1
GMKN,1,20200921,175700,19004.000000,19051.000000,19004.000000,19051.000000,20

```

5. Загрузка данных:

wget

<https://raw.githubusercontent.com/dzhamalovas/DS/refs/heads/main/GMKN.csv>

```
hadoop@devopsvm:~$ wget https://raw.githubusercontent.com/dzhamalovas/DS/refs/heads/main/GMKN.csv
--2024-10-18 20:51:12-- https://raw.githubusercontent.com/dzhamalovas/DS/refs/heads/main/GMKN.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25155689 (24M) [text/plain]
Saving to: 'GMKN.csv'

GMKN.csv                100%[=====>] 23.99M  1.98MB/s   in 29s


2024-10-18 20:51:41 (849 KB/s) - 'GMKN.csv' saved [25155689/25155689]
```

Проверка загрузки:

```
hadoop@devopsvm:~$ ls
GDP.csv  GMKN.csv  hadoop-3.3.5.tar.gz  hdfs  output  snap  spark-3.4.3-bin-hadoop3.tgz
hadoop@devopsvm:~$
```

```
hdfs dfs -put GMKN.csv /user3/hadoop/economic_data/
```

```
hadoop@devopsvm:~$ hdfs dfs -put GMKN.csv /user3/hadoop/economic_data/
2024-10-18 20:54:09,740 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
hadoop@devopsvm:~$
```

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	23.99 MB	Oct 18 20:54	1	128 MB	GMKN.csv	

Showing 1 to 1 of 1 entries

[Previous](#)
[1](#)
[Next](#)

spark-shell

```
hadoop@devopsvm:~$ spark-shell
24/10/18 21:02:54 WARN Utils: Your hostname, devopsvm resolves to a loopback address: 127.0.1.1; using 172.20.10.3
instead (on interface enp0s3)
24/10/18 21:02:54 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/18 21:02:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-
java classes where applicable
Spark context Web UI available at http://172.20.10.3:4040
Spark context available as 'sc' (master = local[*], app id = local-1729274580507).
Spark session available as 'spark'.
Welcome to
```

```

      /-/- /-/- /-/- /-/- /-/-
     / \ / \ / \ / \ / \ / \
    /   /   /   /   /   /   /
   /___/___/___/___/___/___/
                   version 3.4.3

```

```
Using Scala version 2.12.17 (OpenJDK 64-Bit Server VM, Java 11.0.24)
Type in expressions to have them evaluated.
Type :help for more information.
```

```
scala> 
```

```
val data =
```

```
spark.read.option("header","true").csv("file:///home/hadoop/GMKN.csv")
```



```
scala> val data = spark.read.option("header", "true").csv("file:///home/hadoop/GMKN.csv")
data: org.apache.spark.sql.DataFrame = [<TICKER>: string, <PER>: string ... 7 more fields]

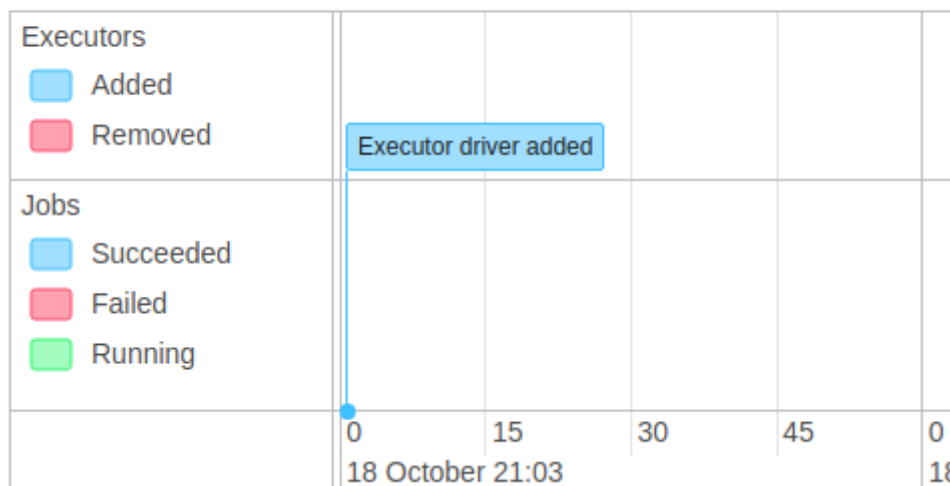
scala> data.printSchema()
root
|-- <TICKER>: string (nullable = true)
|-- <PER>: string (nullable = true)
|-- <DATE>: string (nullable = true)
|-- <TIME>: string (nullable = true)
|-- <OPEN>: string (nullable = true)
|-- <HIGH>: string (nullable = true)
|-- <LOW>: string (nullable = true)
|-- <CLOSE>: string (nullable = true)
|-- <VOL>: string (nullable = true)

scala> █
```

Работа Spark:

▼ Event Timeline

☐ Enable zooming



9. Вычисление агрегации данных:

Добавим столбцы для года и месяца на основе столбца <DATE>

```
scala> val dataWithYear = data.withColumn("Year", year(to_date(col("<DATE>"), "yyyyMMdd")))
dataWithYear: org.apache.spark.sql.DataFrame = [<TICKER>: string, <PER>: string ... 8 more fields]

scala> val dataWithYearAndMonth = dataWithYear.withColumn("Month", month(to_date(col("<DATE>"), "yyyyMMdd")))
dataWithYearAndMonth: org.apache.spark.sql.DataFrame = [<TICKER>: string, <PER>: string ... 9 more fields]

scala>
```

Приведение типов к числовым для корректной агрегации

```
val numericData = data.withColumn("<CLOSE>",
  col("<CLOSE>").cast("double"))
  .withColumn("<HIGH>", col("<HIGH>").cast("double"))
  .withColumn("<LOW>", col("<LOW>").cast("double"))
  .withColumn("<VOL>", col("<VOL>").cast("long"))
```

```
scala> val numericData = data.withColumn("<CLOSE>", col("<CLOSE>").cast("double"))
numericData: org.apache.spark.sql.DataFrame = [<TICKER>: string, <PER>: string ... 7 more fields]

scala> .withColumn("<HIGH>", col("<HIGH>").cast("double"))
res18: org.apache.spark.sql.DataFrame = [<TICKER>: string, <PER>: string ... 7 more fields]

scala> .withColumn("<LOW>", col("<LOW>").cast("double"))
res19: org.apache.spark.sql.DataFrame = [<TICKER>: string, <PER>: string ... 7 more fields]

scala> .withColumn("<VOL>", col("<VOL>").cast("long"))
```

Агрегация данных

```
val aggregatedResults = numericData.agg(
  avg("<CLOSE>").alias("Avg_Close_Price"),
  max("<HIGH>").alias("Max_High_Price"),
  min("<LOW>").alias("Min_Low_Price"),
  sum("<VOL>").alias("Total_Volume")
)
```

```
scala> val aggregatedResults = numericData.agg(
  | avg("<CLOSE>").alias("Avg_Close_Price"),
  | max("<HIGH>").alias("Max_High_Price"),
  | min("<LOW>").alias("Min_Low_Price"),
  | sum("<VOL>").alias("Total_Volume")
  | )
aggregatedResults: org.apache.spark.sql.DataFrame = [Avg_Close_Price: double, Max_High_Price: string ... 2 more fields]
```

Вывод результатов:

```
scala> aggregatedResults.show()
+-----+-----+-----+-----+
| Avg_Close_Price|Max_High_Price|Min_Low_Price|Total_Volume|
+-----+-----+-----+-----+
|21511.068570376072| 28500.00000000|11640.00000000| 1.3570056E7|
+-----+-----+-----+-----+
```

Сохранение результатов в CSV файл:

```
aggregatedResults.write
  .option("header", "true")
  .csv("/home/hadoop/output/aggregated_results.csv")
```

```
hadoop@devopsvm:~$ cd output
hadoop@devopsvm:~/output$ ls
aggregated_results.csv  avg_GDR.csv
hadoop@devopsvm:~/output$
```

10. Выход из Spark.

```
scala> :q
hadoop@devopsvm:~$
```

11. Переименовываем файл:

```
mv part-00000-*.csv aggr.csv
```

```
hadoop@devopsvm:~/output/aggregated_results.csv$ ls
aggr.csv _SUCCESS
hadoop@devopsvm:~/output/aggregated_results.csv$
```

12. Перенос данных в HDFS:

```
hadoop@devopsvm:~$ hdfs dfs -put /home/hadoop/output/aggregated_results.csv/aggr.csv /user3/hadoop/input/
2024-10-18 22:28:56,107 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
hadoop@devopsvm:~$ hdfs dfs -ls /user/hadoop/input/
2024-10-18 22:29:13,847 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
hadoop@devopsvm:~$
```

Browse Directory

/user3/hadoop/input

Go!

Show

25

entries

Search:

<input type="checkbox"/>		Permission		Owner		Group		Size		Last Modified		Replication		Block Size		Name	
<input type="checkbox"/>		-rw-r--r--		hadoop		supergroup		117 B		Oct 18 22:28		1		128 MB		aggr.csv	
<input type="checkbox"/>		-rw-r--r--		hadoop		supergroup		27 B		Oct 18 13:55		1		128 MB		avg.csv	

Showing 1 to 2 of 2 entries

Previous

1

Next

Hadoop, 2023.

13. Путь economic_data:

/user3/hadoop/economic_data

Go!

Show

25

entries

Search:

☐

Permission

Owner

Group

Size

Last Modified

Replication

Block Size

Name

☐

-rw-r--r--

hadoop

supergroup

23.99 MB

Oct 18 20:54

1

128 MB

GMKN.csv

Showing 1 to 1 of 1 entries

Previous

1

Next

Hadoop, 2023.

14. Остановка Hadoop в Ubuntu:

```
stop-yarn.sh
```

```
hadoop@devopsvm:~$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
hadoop@devopsvm:~$
```

```
stop-dfs.sh
```

```
hadoop@devopsvm:~$ stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [devopsvm]
2024-10-18 22:45:44,877 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

stop-all.sh

```
hadoop@devopsvm:~$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [devopsvm]
2024-10-18 22:46:24,037 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Stopping nodemanagers
Stopping resourcemanager
```

Работа в Jupyter:

Установка необходимых библиотек:

```
[1]: !pip install pyspark
Requirement already satisfied: pyspark in /home/devops/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in /home/devops/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from pyspark) (0.10.9.7)

[2]: import pandas as pd
import matplotlib.pyplot as plt
```

Создание сессии и чтение данных:

```
[3]: from pyspark.sql import SparkSession

# Создание SparkSession
spark = SparkSession.builder \
    .appName("Economic Data Analysis") \
    .config("spark.hadoop.fs.defaultFS", "hdfs://localhost:9000") \
    .config("spark.ui.port", "4050") \
    .getOrCreate()

# Установка количества разделов для shuffle операций
spark.conf.set("spark.sql.shuffle.partitions", "50")

24/10/18 23:08:20 WARN Utils: Your hostname, devopsvm resolves to a loopback address: 127.0.1.1; using 172.20.10.3 instead (on interface enp0s3)
24/10/18 23:08:20 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/18 23:08:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

[4]: # Чтение данных из HDFS
file_path = "hdfs://localhost:9000/user3/hadoop/economic_data/GMKN.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Просмотр первых строк данных
df.show(5)

+-----+-----+-----+-----+-----+-----+-----+-----+
|<TICKER>|<PER>|<DATE>|<TIME>|<OPEN>|<HIGH>|<LOW>|<CLOSE>|<VOL>|
+-----+-----+-----+-----+-----+-----+-----+-----+
| GMKN | 1 | 20200825 | 131300 | 19500.0 | 19500.0 | 19500.0 | 19500.0 | 1 |
| GMKN | 1 | 20200826 | 193400 | 20550.0 | 20550.0 | 20550.0 | 20550.0 | 1 |
| GMKN | 1 | 20200902 | 225600 | 20600.0 | 20600.0 | 20600.0 | 20600.0 | 4 |
| GMKN | 1 | 20200902 | 230500 | 20600.0 | 20600.0 | 20600.0 | 20600.0 | 10 |
| GMKN | 1 | 20200902 | 234200 | 20600.0 | 20600.0 | 20600.0 | 20600.0 | 5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

[9]: pandas_df = df.toPandas()
pandas_df.head()
```

	<TICKER>	<PER>	<DATE>	<TIME>	<OPEN>	<HIGH>	<LOW>	<CLOSE>	<VOL>
0	GMKN	1	20200825	131300	19500.0	19500.0	19500.0	19500.0	1
1	GMKN	1	20200826	193400	20550.0	20550.0	20550.0	20550.0	1
2	GMKN	1	20200902	225600	20600.0	20600.0	20600.0	20600.0	4
3	GMKN	1	20200902	230500	20600.0	20600.0	20600.0	20600.0	10
4	GMKN	1	20200902	234200	20600.0	20600.0	20600.0	20600.0	5

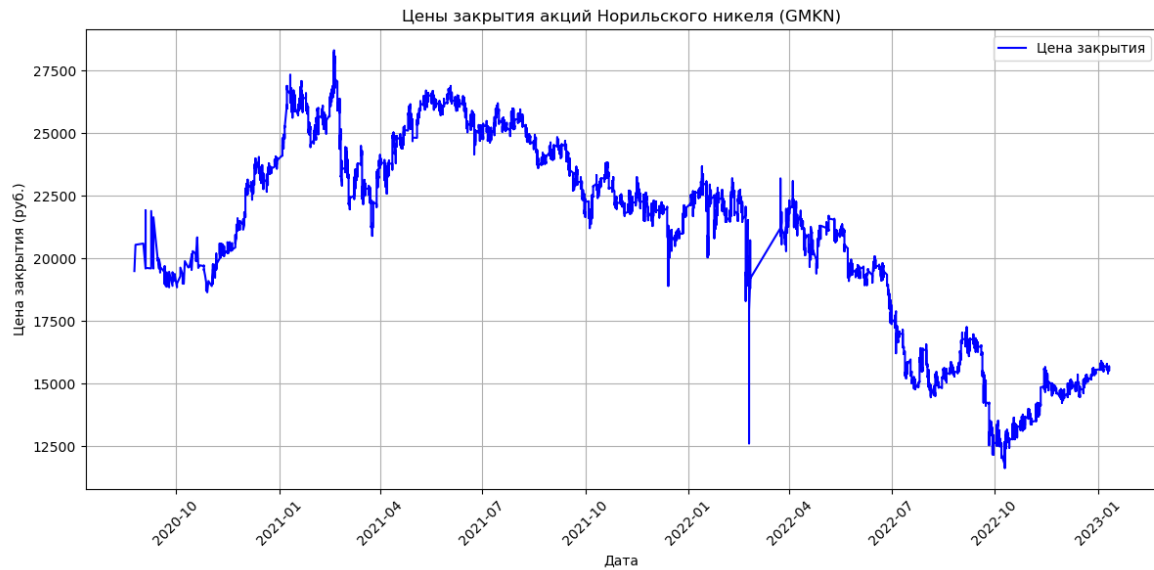
Построение графика цен закрытий Норильского никеля:

```
pandas_df['<DATE>'] = pd.to_datetime(pandas_df['<DATE>'], format='%Y%m%d')

plt.figure(figsize=(12, 6))
plt.plot(pandas_df['<DATE>'], pandas_df['<CLOSE>'], label='Цена закрытия', color='blue')

# Настройка графика
plt.title('Цены закрытия акций Норильского никеля (GMKN)')
plt.xlabel('Дата')
plt.ylabel('Цена закрытия (руб.)')
plt.xticks(rotation=45)
plt.legend()
plt.grid()

# Показать график
plt.tight_layout()
plt.show()
```



Посмотрим уникальные значения <PER> и <VOL>, а затем построим график при объеме торгов = 10

```
[17]: # Получение уникальных значений из столбца <PER> и <VOL>
unique_per = pandas_df['<PER>'].unique()
unique_vol = pandas_df['<VOL>'].unique()

print("Уникальные значения <PER>:", unique_per)
print("Уникальные значения <VOL>:", unique_vol)
```

Уникальные значения <PER>: [1]
 Уникальные значения <VOL>: [1 4 10 ... 909 1229 813]

```
[22]: # Предположим, что мы выбрали уникальное значение объема (например, 1)
selected_volume = 10

# Фильтрация данных по выбранному объему
filtered_data = pandas_df[pandas_df['<VOL>'] == selected_volume]

# Создание графика
plt.figure(figsize=(12, 6))
plt.plot(filtered_data['<DATE>'], filtered_data['<CLOSE>'], label='Цена закрытия', color='blue')

# Настройка графика
plt.title(f'Цены закрытия акций Норильского никеля (GMKN) при объеме торгов {selected_volume}')
plt.xlabel('Дата')
plt.ylabel('Цена закрытия (руб.)')
plt.xticks(rotation=45) # Поворот дат для удобства
plt.legend()
plt.grid()

# Показать график
plt.tight_layout()
plt.show()
```

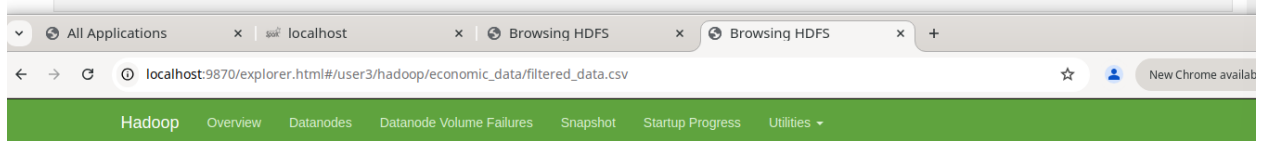


Преобразование данных в PySpark DataFrame и путь в HDFS:

```
[24]: # Преобразование Pandas DataFrame обратно в PySpark DataFrame
filtered_data_spark = spark.createDataFrame(filtered_data)

[31]: # Путь в HDFS для сохранения
hdfs_output_path = "hdfs://localhost:9000/user3/hadoop/economic_data/filtered_data.csv"

# Сохранение отфильтрованных данных в HDFS в формате CSV
filtered_data_spark.write.csv(hdfs_output_path, header=True, mode='overwrite')
```



Browse Directory

/user3/hadoop/economic_data/filtered_data.csv Go!

Show: 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	devops	supergroup	0 B	Oct 18 23:40	3	128 MB	_SUCCESS
<input type="checkbox"/>	-rw-r--r--	devops	supergroup	315.51 KB	Oct 18 23:40	3	128 MB	part-00000-ba8abae7-78c0-4217-a9b1-a2297c22984a-c000.csv
<input type="checkbox"/>	-rw-r--r--	devops	supergroup	363.65 KB	Oct 18 23:40	3	128 MB	part-00001-ba8abae7-78c0-4217-a9b1-a2297c22984a-c000.csv

Showing 1 to 3 of 3 entries

Previous 1 Next

Вывод:

Apache Spark — мощный инструмент для анализа больших объемов экономических данных благодаря высокой скорости и обработке данных в памяти. Он отлично подходит для итеративных задач, реального времени и машинного обучения. Hadoop эффективен для долговременного хранения данных и пакетной обработки, но медленнее из-за использования дискового хранилища.

Spark предпочтителен для анализа экономических данных в реальном времени, машинного обучения и больших вычислений, тогда как Hadoop больше подходит для хранения и обработки больших массивов данных на диске.