Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Инструменты для хранения и обработки больших данных

**Лабораторная работа № 5.1**

**Тема:**

**«**Развертывание и настройка кластера Hadoop**»**

Выполнил(а):

Вереина М.С., АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2024

**Цель работы:**

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

**Постановка задачи**

Проанализировать экономические данные, содержащиеся в вашем файле, который находится в файловой системе Hadoop (HDFS). Задача заключается в извлечении, обработке, и анализе данных с целью выявления закономерностей, тенденций, и создания визуализаций на основе предоставленных данных.

**Вариант 3.** Установка Apache Hadoop и выполнение задачи на сортировку данных.

Данные: Исторические данные по акциям Лукойла (LKOH) с сайта Московской биржи (moex.com).

Операции: Фильтрация данных за последние 5 лет, расчет минимальной цены закрытия, группировка по годам.

**Ход работы**

1. Запуск Hadoop и загрузка файла

Шаг 1. Запуск Hadoop.

1. Запуск надстройки над основной файловой системой.
2. Запуск службы YARN, необходимой для управления и контроля за затрачиваемыми ресурсами на выполнение задач, связанных с большими данными.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рисунок 1 – Подключение к Hadoop и запуск Yarn

Шаг 2. Проверка работы Hadoop.

Воспользуемся jps — это утилита командной строки, которая отображает информацию о процессах виртуальной машины Java (JVM), запущенных в системе.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 2 – Выполнение команды jps

После выполнения команды можно увидеть следующие процессы: NameNode, DataNode, SecondaryNameNode, ResourceManager, NodeManager.

В стандартной конфигурации Hadoop HDFS предоставляет веб-интерфейс, доступный через веб-браузер на порту 9870. Этот интерфейс позволяет просматривать состояние и структуру HDFS, а также выполнять некоторые операции.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рисунок 3 – Веб-интерфейс Hadoop в веб-браузере

На веб-интерфейсе YARN можно увидеть информацию о запущенных приложениях, статусе узлов и других метриках кластера. Это удобный способ управления и мониторинга задач.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 4 – Веб-интерфейс Yarn

Шаг 3. Работа с данными.

Необходимо создать директорию в HDFS:

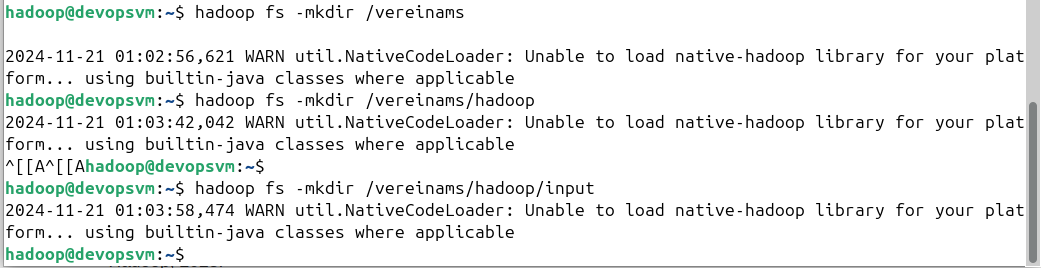


Рисунок 5 – Использование команд для создания каталога в HDFS

При успешном создании данная директория будет доступна в веб-интерфейсе Hadoop.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 6 – Отображение директории в веб-интерфейсе Hadoop

Загрузка данных в файловую систему HDFS.

Для загрузки файла необходимо перейти в ранее созданную директорию и воспользоваться иконкой для загрузки.

Перед этим предоставить необходимые права доступа к созданным директориям.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 7 – Предоставление доступа к директориям

В рамках указанного варианта в Hadoop будут загружены 225МБ данных об изменении акций компании Лукойл.

Изображение выглядит как текст, программное обеспечение, число, веб-страница

Автоматически созданное описание

Рисунок 8 – Элемент веб-интерфейса для загрузки файла в HDFS

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 9 – Загрузка файла в HDFS

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 10 – Результат загрузки файла в HDFS

Запуск Spark

Далее необходимо запустить фреймворк Spark с помощью команды spark-shell, с помощью которого можно выполнять операции с загруженными данными.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 11 – Запуск Spark

1. Подключение к Hadoop и загрузка данных.

Шаг 1.

Подключиться к HDFS и убедиться, что файл доступен по пути hdfs://localhost:9000/vereinams/hadoop/input/LKOH.csv.

Использовать PySpark или Pandas для загрузки данных из HDFS в DataFrame, который можно будет использовать для анализа.

Для этого необходимо создать SparkSession – это способ инициализации базовой функциональности PySpark для программного создания PySpark RDD, DataFrame и Dataset. SparkSession можно создать с помощью SparkSession.builder, который представляет собой реализацию шаблона проектирования Builder

Для того чтобы убедиться, что файл доступен по указанному пути, необходимо воспользоваться командой spark.read.scv, в рамках которой задать путь до файла. Если он корректен, то с файлом можно работать дальше, например, отобразить первые 5 строк.

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 12 – Создание SparkSession и отображение первых 5 строк.

1. Исследование и очистка данных.

1) Проверить структуру данных и типы столбцов (например, с помощью printSchema() для PySpark или describe() для Pandas).

Изображение выглядит как снимок экрана, текст, линия, программное обеспечение

Автоматически созданное описание

Рисунок 13 – Проверка структуры данных и типов столбцов

2) Убедиться, что все данные корректны, и преобразовать необходимые столбцы в числовые форматы, если они изначально представлены в виде строк.

С помощью методов printSchema() и df.dtypes получилось определить структуру и типы столбцов. Все они указаны корректно:

1. **Date: date**. Столбец с датой сделок/торговых сессий. Использование типа date позволяет хранить только дату без времени.
2. **Number of Transactions: integer**. Столбец, который хранит количество транзакций. Тип integer подходит, так как количество транзакций обычно не превышает 2 миллиардов.
3. **Volume of Transactions (shares): double**. Столбец, который хранит объем транзакций в акциях. Тип double позволяет учитывать дробные значения, объем может быть нецелым.
4. **Volume of Transactions (rubles): double**. Столбец, который хранит объем транзакций в рублях. Использование типа double также корректно для хранения денежных сумм, требуется учитывать дробные значения.
5. **Max Price: double**. Столбец, который хранит максимальную цену акций за период, они могут иметь дробные значения.
6. **Min Price: double**. Столбец, который хранит минимальную цену акций за период. Аналогично предыдущему столбцу, тип double корректен.
7. **Percent Change: double**. Столбец, который хранит процентное изменение цены акций. Использование типа double оправдано, так как процентные изменения могут быть дробными.
8. **Number of Participants: integer**. Столбец, который хранит количество участников торгов. Тип integer является подходящим выбором для хранения целых чисел.

3) Проверить данные на наличие пропущенных или некорректных значений, удалить или заполнить такие значения в зависимости от ситуации.

Для этого необходимо перевести коллекцию Spark в датафрейм Pandas, после чего проверяем наличие пропущенных нулей в каждом столбце:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 14 – Поиск пропущенных значений

3) Анализ данных.

- Провести базовый статистический анализ данных.

- Вычислить средние значения, медианы, минимумы и максимумы для экономических параметров.

Пропущенных значений в данном случае не оказалось, поэтому можно использовать метод описательной статистики describe(), чтобы сделать дальнейшие выводы по корректности данных.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 15 – Результат выполнения команды describe()

Приведенные значения указывают на корректность данных, как с точки зрения качества данных, так и по их наполненности. На основании этого можно сделать следующие выводы:

**Количество транзакций (Number of Transactions)**:

* **Среднее значение**: 50 000 шт

Среднее количество транзакций указывает на активность торговли акциями Лукойла. Наличие нулевых значений лишь указывает на дни с отсутствием торговых операций.

**Объем транзакций в акциях (Volume of Transactions (shares))**:

* **Среднее значение**: 19 400 000 акций

Объем торгов показывает высокую активность на рынке.

**Объем транзакций в рублях (Volume of Transactions (rubles))**:

Высокий средний объем транзакций в рублях говорит о значительных финансовых операциях с акциями Лукойла.

**Максимальная и минимальная цена акций (Max Price и Min Price)**:

Эти значения показывают диапазон цен за рассматриваемый период и могут быть полезны для анализа волатильности акций, в среднем составляют ~ 4500 рублей.

**Процент изменения цены акций (Percent Change)**:

* **Среднее значение**: −5.27%

Отрицательное среднее значение указывает на общее снижение цен за рассматриваемый период.

**Количество участников торгов (Number of Participants)**:

* **Среднее значение**: 2.51×1022.51×102 (или около 251 участника)

Данные показывают высокую активность торговли акциями Лукойла с большим объемом транзакций как в количестве акций, так и в денежном выражении. Наблюдается высокая волатильность цен с отрицательным средним изменением цены за рассматриваемый период. Несмотря на наличие дней без торговых операций, общая тенденция указывает на активность и интерес к акциям компании.

1. Основное задание:

Установка Apache Hadoop и выполнение задачи на сортировку данных.

Данные: Исторические данные по акциям Лукойла (LKOH) с сайта Московской биржи (moex.com).

Операции: Фильтрация данных за последние 5 лет, расчет минимальной цены закрытия, группировка по годам.

Исходя из того, что при просмотре первых 5 значений датафрейма, данные находились вразброс, необходимо применить к ним сортировку для корректного отображения, в данном случае по дате, так как данные представляют собой исторические значения акций с помощью метода sort\_values().

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 16 – применение сортировки к данным

Далее работаем с фильтрацией, необходимо отфильтровать данные за последние 5 лет. Для этого используется метод DateOffset. Data offset в Python используется для смещения данных с целью сравнения изменений во времени или создания производных признаков. Это применяется в финансовом анализе, прогнозировании временных рядов и при подготовке данных для алгоритмов машинного обучения.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 17 – Применение фильтрации по датам

После применения фильтрации данные начинают отображаться с 01.11.2019 – что является датой ровно 5 лет назад.

Далее для того, чтобы вывести минимальную цену закрытия и дату, в которую оно произошло, необходимо воспользоваться встроенной функцией min() к минимальным ценам закрытия. Далее по индексу минимальной цены необходимо вывести соответствующую ему дату с помощью функции loc().

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 18 – Определение минимальной цены закрытия

Далее для того, чтобы сгруппировать исторические данные акций по годам, необходимо в первую очередь создать столбец Года, который будет заполняться по данным из столбца Даты. Далее необходимо для каждого используемого в группировке столбца определить соответствующее агрегированное значение. Для денежных данных, а также для процентных колебаний необходимо использовать вычисление среднего mean, для количественных показателей, таких как объем и количество транзакций, а также их количество необходимо использовать вычисление суммы sum.

Сама группировка осуществляется посредством использования метода groupby().

Изображение выглядит как текст, снимок экрана, меню, число

Автоматически созданное описание

Рисунок 19 – Результат применения группировки к данным

Также по данным, полученным из Hadoop, можно строить графики, однако воспроизводиться они будут очень медленно.

Изображение выглядит как текст, снимок экрана, Шрифт, График

Автоматически созданное описание

Рисунок 20 – Визуализация данных

Также важно уметь сохранять измененные данные обратно в Hadoop, это можно сделать с помощью write.mode().

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 21 – Сохранение измененных данных в Hadoop

Выводы: в ходе выполнения данной лабораторной работы был изучен процесс установки и настройки распределенных систем, таких как Apache Hadoop. Изучены основные операции и функциональные возможности системы, что позволило понять принципы работы с данными и распределенными вычислениями.

Hadoop вместе со Spark и Yarn позволяет работать с большими данными гораздо эффективнее засчет следующих параметров:

* 1. Скорость обработки:

Spark значительно быстрее Hadoop MapReduce благодаря обработке данных в памяти, что делает его идеальным для интерактивных и итеративных задач.

* 1. Универсальность:

Spark поддерживает различные типы обработки данных: пакетную, потоковую, SQL-запросы и машинное обучение, в то время как Hadoop в основном ориентирован на пакетную обработку через MapReduce.

* 1. Интеграция:

Spark может работать с данными, хранящимися в HDFS, что позволяет использовать Hadoop как систему хранения. Это обеспечивает масштабируемость и долговременное хранение данных.

* 1. Управление ресурсами:

YARN (Yet Another Resource Negotiator) позволяет эффективно распределять ресурсы между Hadoop и Spark, что обеспечивает оптимальное использование вычислительных мощностей.

* 1. Обработка больших данных:

Использование Spark вместе с Hadoop позволяет обрабатывать большие объемы данных более эффективно, что особенно полезно для аналитических приложений и задач, требующих высокой скорости обработки.

* 1. Отказоустойчивость:

Оба инструмента обеспечивают отказоустойчивость: Hadoop записывает данные на диск после каждой операции, а Spark использует концепцию RDD (Resilient Distributed Dataset) для восстановления данных в случае сбоя.

В ходе проделанной работы на экспериментах с выполнением операций с RDD Spark и DataFrame Pandas первый показал лучшие результаты. При работе со вторым датафреймом виртуальная машина не справлялась, в связи с чем приходилось несколько раз перезагружать ее, теряя текущий результат. Spark же работает более стабильно и эффективно.