Лабораторная работа №1 (Часть 2): Знакомство с HDFS и Spark Cluster

Цель работы: Получить практический опыт развёртывания standalone-кластера Apache Spark и выполнения базовых операций с распределённой файловой системой Hadoop (HDFS). Освоить процесс запуска простого распределённого приложения.

Стек технологий:

- **Платформа/ОС:** Linux (Ubuntu 22.04 LTS)
- **Фреймворки:** Apache Hadoop (HDFS), Apache Spark (Standalone mode)
- **Язык программирования:** Python (PySpark)
- **Инструментарий:** Terminal, SSH

Теоретическая часть (краткое содержание):

- **Hadoop HDFS (Hadoop Distributed File System):** Распределённая файловая система, предназначенная для хранения очень больших объёмов данных с высокой отказоустойчивостью. Состоит из:
  - **NameNode:** Управляет метаданными файловой системы (иерархия файлов, блоки данных).
  - **DataNode:** Хранит непосредственно блоки данных.
- **Apache Spark (Standalone Mode):** Встроенный простой менеджер кластера, который позволяет запускать Spark-приложения на нескольких машинах без дополнительных систем оркестрации вроде YARN или Kubernetes. Кластер состоит из:
  - **Master (Leader):** Центральный координатор, управляющий работой кластера.
  - **Worker (Slave):** Вычислительный узел, который выполняет код и хранит данные для приложений.

Задание на практическую реализацию:

Этап 1: Установка и настройка Hadoop HDFS и Spark

1. На виртуальной машине из Части 1 установите Java (OpenJDK 8 или 11).
2. Установите Apache Hadoop (только HDFS, без YARN/MR) и Apache Spark в режиме standalone.
3. Настройте одноджузовый (псевдо-распределённый) режим HDFS, где NameNode и DataNode работают на одной машине.
4. Отформатируйте HDFS и запустите сервисы (`start-dfs.sh`).
5. Запустите standalone-кластер Spark (запустите мастер и воркер через `start-master.sh` и `start-worker.sh`).

Этап 2: Основы работы с HDFS

1. Проверьте статус HDFS, используя веб-интерфейс (доступен по умолчанию на `http://<IP-вашей-VM>:9870`) или команду `hdfs dfsadmin -report`.
2. Создайте в HDFS пользовательскую директорию:

```
hdfs dfs -mkdir -p /user/student
```

3. Создайте на локальной машине текстовый файл `test_data.txt` с несколькими строками произвольного текста.
4. Загрузите локальный файл в HDFS:

```
hdfs dfs -put /local/path/to/test_data.txt /user/student/
```

5. Просмотрите содержимое директории в HDFS и убедитесь, что файл загружен:

```
hdfs dfs -ls /user/student
```

6. Просмотрите содержимое удалённого файла:

```
hdfs dfs -cat /user/student/test_data.txt
```

7. Скопируйте файл из HDFS обратно на локальную файловую систему под другим именем (выгрузка):

```
hdfs dfs -get /user/student/test_data.txt /local/path/to/test_data_copy.txt
```

Этап 3: Запуск Spark приложения на развёрнутом кластере

1. Проверьте статус кластера Spark через веб-интерфейс (доступен по умолчанию на `http://<IP-вашей-VM>:8080`).
2. Запустите тестовое приложение для вычисления числа π (Pi) с помощью Spark Submit. Укажите URL вашего мастера:

```
spark-submit --master spark://<your-master-url>:7077 --class
org.apache.spark.examples.SparkPi $SPARK_HOME/examples/jars/spark-
examples_2.12-3.3.1.jar 100
```

Примечание: Версия Spark и Scala в пути к jar-файлу может отличаться. Исправьте путь в соответствии с вашей установкой. Параметр `100` задаёт количество задач для вычисления.

3. Наблюдайте за выполнением задачи в веб-интерфейсе Spark Master/Worker. В логах выполнения и в конце вывода в консоли вы должны увидеть приближённое значение числа Pi.

Требования к оформлению и отчёту:

1. **Скриншоты:** Приложите скриншоты, подтверждающие выполнение каждого этапа:
    - Веб-интерфейс HDFS NameNode (порт 9870), показывающий живые DataNodes.
    - Результат выполнения команд `hdfs dfs -ls` и `hdfs dfs -cat`.
    - Веб-интерфейс Spark Master (порт 8080), показывающий работающий Worker.
    - Веб-интерфейс Spark во время выполнения приложения Spark Pi.
    - Консоль с результатом успешного расчёта числа Pi.
2. **Отчёт:** Краткий письменный отчёт должен содержать:
    - Описание проблем, с которыми вы столкнулись, и их решение.
    - Ответы на вопросы:
        - В чём основная идея и преимущество HDFS по сравнению с локальной файловой системой?
        - Какую роль в кластере Spark играет Master, а какую — Worker?
        - Объясните, почему приложение для расчёта Pi является хорошим примером для демонстрации работы распределённого кластера.

Критерии оценки:

- **Удовлетворительно (3):** Выполнены Этапы 1 и 2. HDFS и Spark установлены, файл успешно загружен в HDFS и прочитан из неё.
- **Хорошо (4):** Выполнены все этапы. Приложение Spark Pi успешно запущено через spark-submit на развёрнутом кластере. Предоставлены скриншоты веб-интерфейсов.
- **Отлично (5):** Выполнены все этапы. В отчёте даны развёрнутые и точные ответы на все теоретические вопросы, демонстрирующие понимание принципов работы HDFS и Spark в standalone-режиме.