ВМК

Задание 4. Hadoop. HDFS. MapReduce.

Практикум 317 группы, весна 2023-2024

Начало выполнения задания: 1 апреля 2024 года, 20:00.
Жёсткий Дедлайн: **1 апреля 2024 года, 23:59.**

Формулировка задания

Данное задание направлено на знакомство с инфраструктурой Apache Hadoop.
Задание состоит из двух частей:

1. Выполнение базовых действий в файловой системе HDFS
2. Реализация и запуск простейшей MapReduce программы

1 Базовые действия в HDFS (1 балл)

В этом пункте необходимо написать скрипт `hdfs.sh`, выполняющий следующую последовательность действий из корневой директории `namenode` Hadoop кластера:

1. Создать локальный файл `test.txt` размером 100Mb
2. Создать hdfs-директории `temp` и `logs`
3. Записать файл `test.txt` в директорию `temp`
4. Посмотреть свойства записанного файла
5. Переместить файл `test.txt` в директорию `logs`
6. Установить фактор репликации для файла равным 1
7. Скопировать `test.txt` в `test2.txt`
8. Скопировать директорию `logs` в `logs2` с помощью `hadoop distcp`
9. Установить права доступа, означающие чтение и запись только для владельца для файла `test2.txt` в директории `logs2`
10. Вывести свойства всех файлов в `logs2`
11. Посмотреть размер всех директорий в `/`
12. Удалить директорию `logs`
13. Запустить `fsck` на директории `logs2`
14. Посмотреть отчет о HDFS через `dfsadmin`
15. Переместить `/logs2/test2.txt` в локальную папку `/`
16. Добавить содержимое локального файла `test2.txt` в конец файла `/logs2/test.txt` в `hdfs`
17. Вывести размер для каждого файла в `/logs2` в Mb

Каждый пункт должен соответствовать одной команде. Результат выполнения (stdout, stderr) `hdfs.sh` сохраните в файле `hdfs.output.txt`.

Также, ответьте на следующие вопросы в файле `hdfs.answers.md`:

1. Сколько сплитов было обработано командой `hadoop distcp`? Сколько мапперов и сколько редьюсеров было задействовано в процессе её выполнения?
2. Сколько реплик блоков отсутствует после выполнения команды `fsck`? Объясните в чём причина отсутствия реплик.
3. Какой размер файловой системы HDFS?

В качестве решения необходимо сдать три файла `hdfs.sh`, `hdfs.output.txt`, `hdfs.answers.md` с именно такими названиями.

2 MapReduce умножение матриц (4 балла)

В данном пункте необходимо реализовать программу для Hadoop Streaming, реализующую умножение матриц.

Положим $A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{m \times k}, C = AB \in \mathbb{R}^{n \times k}$.

Будем представлять входные данные (то есть матрицы A, B) в виде файлов `A.txt`, `B.txt`, где элементы матриц записаны построчно через пробел. При этом, явно пронумеруем все строки. Результат перемножения (`C.txt`) будем представлять в виде пар ключ-значение, где в качестве ключа выступает пара (номер строки, номер столбца). Порядок строк при этом будем игнорировать.

$$A = \begin{pmatrix} 3 & 4 & 3 & 2 \\ 0 & 5 & 2 & 1 \\ 0 & 4 & 2 & 0 \end{pmatrix} \Longrightarrow \underbrace{\begin{matrix} 0 & 3 & 4 & 3 & 2 \\ 1 & 0 & 5 & 2 & 1 \\ 2 & 0 & 4 & 2 & 0 \end{matrix}}_{\text{A.txt}} \qquad C = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \Longrightarrow \underbrace{\begin{matrix} 0{,}1\ 2 \\ 0{,}0\ 1 \\ 1{,}0\ 3 \\ 1{,}1\ 4 \end{matrix}}_{\text{C.txt}}$$

Вам необходимо реализовать MapReduce программу, которая принимает на вход матрицы A, B в описанном выше формате, вычисляет их произведение и возвращает результат в виде списка ключей-значений.

В задании предоставляются некоторые вспомогательные скрипты, которые упростят решение задачи:

1. `generate_task.py` – генерирует случайные матрицы A, B заданного размера вычисляет их произведение и сохраняет A, B, C в необходимом формате.

   Пример запуска: `python3 generate_task.py -n 10 -m 20 -k 30`

2. `check_answer.py` – сравнивает истинный ответ `C.txt` с ответом, сгенерированным MapReduce программой. Скрипт учитывает, что результат MapReduce программы разбит на файлы с соответствии с числом редьюсеров.

   Пример запуска: `python3 check_answer.py -n 10 -m 20 -k 30`

Для решения задачи необходимо реализовать следующие скрипты:

1. `mapper.py`, `reducer.py` – код маппера и редьюсера для перемножения матриц
2. `run_hadoop.sh` – код для подготовки директорий на namenode, запуска Hadoop Streaming задачи и копирования результата из HDFS в локальную файловую систему namenode
3. `run.sh` – скрипт для запуска всего пайплайна. Данный скрипт должен принимать в качестве аргументов командной строки размерность задачи (n, m, k), число мапперов и число редьюсеров, с которым будет запускаться MapReduce задача на кластере.

   Сначала выполняется генерация матриц A, B, C заданного размера, затем происходит копирование данных внутрь контейнера и запуск Hadoop Streaming задачи на namenode. По завершении MapReduce программы, скрипт копирует результат из контейнера и проверяет его на корректность

   Пример запуска: `sh run.sh 10 20 30 5 5`

В результате, проект должен в точности удовлетворять структуре на диаграмме 1. В качестве решения необходимо прислать архив данной структуры с названием `<ФИО>_task_04.zip`. Папку `data` можно не присылать.

Решение будет оценено в полный балл только при условии, что запуск `run.sh` с различными параметрами будет успешен. Ваша программа должна корректно работать при размерности задачи $n, m, k \leq 500$ при любом числе мапперов и редьюсеров.

```
matmul_streaming
├── data
│   ├── input ..................................Входные данные для MapReduce задачи
│   │   ├── A.txt
│   │   └── B.txt
│   ├── output ..............................Результат выполнения MapReduce задачи
│   │   ├── _SUCCESS
│   │   ├── part-00000
│   │   └── ...
│   └── C.txt ..............................................Правильный ответ на задачу
├── src ..........................................Исходники MapReduce алгоритма
│   ├── mapper.py
│   └── reducer.py
├── check_answer.py ................Скрипт для проверки решения на корректность
├── generate_task.py . Скрипт для генерации данных для задачи умножения матриц
├── run_hadoop.sh ..............Скрипт для запуска Hadoop Streaming на namenode
└── run.sh ................Скрипт для генерации MR задачи, её запуска и проверки
```

Рис. 1: Требуемая структура решения

Несколько важных замечаний, которые помогут при решении:

1. Можно использовать решение задачи Word Count. В частности, обратите внимание на скрипт `run_hadoop.sh`
2. Для передачи размерности задачи внутрь маппера и редьюсера можно использовать переменные окружения. Подробнее про установку переменных окружения для воркеров можно прочитать здесь
3. Флаги для изменения числа мапперов, редьюсеров и информацию о других тонких настройках Hadoop Streaming можно найти в документации
4. Для решения задачи Вам может потребоваться определить, какой именно файл обрабатывается воркером в данный момент. Для решения этой задачи можно также использовать переменные окружения
5. При выводе и сохранении чисел с плавающей точкой обратите внимание на число знаков после запятой. Рекомендуется использовать порядка 5 знаков после запятой для достижения приемлемой точности. Учтите, что большое количество знаков после запятой кратно замедляет работу MapReduce приложений из-за увеличения объёма передаваемых данных
6. Для копирования данных в контейнер и из контейнера можно использовать специальные команды docker
7. Для запуска программ внутри контейнера также есть специальные команды docker