GeekBrains

BigData. Введение в экосистему Hadoop.

Урок 3. YARN & MapReduce

На уроке рассматриваем основные принципы организации распределенных вычислений в Hadoop. Что такое YARN и для чего он применяется, а также архитектуру MapReduce вычислений.

Оглавление

Теоретическая часть

Yarn

Apache Hadoop YARN - это система для планирования заданий и управления кластером. До получения официального названия YARN неофициально назывался *MapReduce 2* или *NextGen MapReduce*.

Будучи одним из основных компонентов Apache Hadoop, YARN отвечает за распределение системных ресурсов различным приложениям, работающим в кластере Hadoop, и за и планирование задач, выполняемых на разных узлах кластера.

Архитектура YARN

ResourceManager (RM) - менеджер ресурсов, задачей которого является распределение ресурсов, необходимых для работы приложений, и наблюдение за вычислительными узлами, на которых эти приложения выполняются.

ApplicationMaster (AM) – компонент, ответственный за планирование жизненного цикла, координацию и отслеживание статуса выполнения распределенного приложения. Каждое приложение имеет свой экземпляр ApplicationMaster.

NodeManager (NM) – агент, запущенный на вычислительном узле и отвечающий за отслеживание используемых вычислительных ресурсов (ЦП, RAM и т.д.), за управление логами и за отправку отчетов по используемым ресурсам планировщику менеджера ресурсов ResourceManager/Scheduler. NodeManager управляет абстрактными контейнерами, которые представляют собой ресурсы на узле, доступные для конкретного приложения.

Контейнер (Container) - набор физических ресурсов, таких как ЦП, RAM, диски и др. в одной ноде.

Управление кластером

Кластер YARN вступает в работу с приходом запроса от клиента. ResourceManager выделяет необходимые ресурсы для контейнера и запускает ApplicationMaster для обслуживания указанного приложения. ApplicationMaster выделяет контейнеры для приложения в каждом узле и контролирует их работу до завершения работы приложения.

Планировщик FIFO - простой планировщик, который распределяет все ресурсы по задачам в порядке поступления на выполнение.

Fair-планировщик - обеспечивает все работающие приложения примерно равными ресурсами. Если используются очереди – то учитывается вес очереди для приоритета по ресурсам.

Capacity-планировщик - для каждой очереди конфигурируется минимальная и максимальная квота по ресурсам. Пользователям кластера выдаются права на очереди, в которых они могут запускать задачи на кластере.

MapReduce

MapReduce — это фреймворк для вычисления некоторых наборов распределенных задач на кластере.

Вычисления состоят из фаз Map, Sort, Shuffle, Merge, Reduce. Для упрощения рассматриваем две фазы Map и Reduce.

На фазе Map происходит предварительная обработка и разметка данных.

На фазе Reduce происходит “свертка”, то есть финальная обработка полученных и сохранение результатов

Слабые места такого подхода заключается в том, что все промежуточные результаты сохраняются на диск и передаются через сеть, что замедляется процесс по сравнению с операциями в памяти.

Практическая часть

Для запуска задачи MapReduce нужно выполнить команду

```
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar pi 32 10000
```

Также можно изучить различные демо-задачи, которые поставляются в пакете hadoop-mapreduce-examples.jar

Для отслеживания задания в yarn нужно зайти на страницу по адресу http://185.241.193.174:8088/. Нужно найти задачу под своим пользователем и проверить статус выполнения.

Работа задачи MapReduce на популярном примере подсчета повторения слов в файле. Для выполнение, нужно подготовить входные данные. Загрузить любой файл с помощью команды scp с локального компьютера или скачать с помощью команды wget. Далее нужно создать отдельную папку в hdfs, используя команду mkdir:

```
hdfs dfs -mkdir input
```

Затем загрузить фаил в hdfs-папку с помощью -put

```
hdfs dfs -put test.txt input/.
```

В локальной папке сервера, на котором будет запускаться MapReduce нужно создать два файла mapper.py и reducer.py. Ниже приведен код этих файлов:

Mapper.py

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
    # удаляем лишние пробелы
    line = line.strip()
    # делим строки на слова
    words = line.split()
    # добавляем значение для счетчика
    for word in words:
        # выводим в output пару ключ и значение
        print '%s\t%s' % (word, 1)
```

Reducer.py

```
#!/usr/bin/env python
import sys

current_word = None
current_count = 0
word = None

# читаем строки из STDIN
for line in sys.stdin:
    # удаляем пробелы в конце и начале строки
```

```
    line = line.strip()

    # разделяем пары ключ и значение
    word, count = line.split('\t', 1)

    # преобразуем значение в число
    try:
        count = int(count)
    except ValueError:
        # игнорируем ошибки
        continue

    # на Reduce приходят данные после фазы Sort
    # поэтому этот код будет работать
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # пишем результат в STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# не забываем про последнее слово
if current_word == word:
```

```
print '%s\t%s' % (current_word, current_count)
```

Код в mapper.py и reducer.py написан для выполнения с python3. Нужно помнить об этом и если в системе стоит отличная версия python, то точно указывать с каким интерпретатором запускать. Локальное тестирования вычисления производится следующей командой:

```
cat test.txt | python mapper.py | sort | python reducer.py
```

Для запуска распределенного вычисления нужно воспользоваться следующей командой:

```
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar -input test -output result -mapper "python mapper.py" -reducer "python reducer.py" -file mapper.py  -file reducer.py
```

Обзор результата

```
hdfs dfs -cat result/*
```

Для повторного запуска нужно не забыть удалить старый результат командой:

```
hdfs dfs -rm -r result
```

Домашнее задание

1. Запустить задачу из примеров, например, вычисление pi методом Монте-Карло

   ```
   yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar pi 32 10000
   ```

   зайти на ResourceManager http://manager.novalocal:8088 и найти свою задачу.

2. Запустить WordCount и доработать скрипт из примера, чтобы удалялись знаки препинания и слова считались в нижнем регистре
3. *реализовать алгоритм join на MapReduce

Задачи со * предназначены для продвинутых учеников, которым мало сделать обычное ДЗ.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. https://www.ibm.com/developerworks/ru/library/bd-yarn-intro/.
2. https://ru.bmstu.wiki/YARN_(Yet_Another_Resource_Negotiator).