BIGDATA TEAM

HW #03: MapReduce-advanced

Deadline: 24.06.2019, 08:00

1. Описание задания.

В данном ДЗ нужно решить **1 из 3 задач**. Решение надо выполнить на Hadoop Streaming (для желающих, можно на Java, для этого см. документацию по Hadoop Java API по адресу - http://hadoop.apache.org/docs/r2.6.1/api/).

При решении задач старайтесь использовать оптимальный MapReduce-алгоритм:

- использовать как можно меньшее кол-во Hadoop Job;
- использовать combiner для ускорения вычислений;
- использовать больше, чем 1 reducer (1 reducer разрешается использовать только в финальной job'e, при сортировке результата)

Номер задачи определяется по ID (см. таблицу с оценками):

```
ID % 3 + 1
```

Перед решением задачи **обязательно** изучите разделы (6) Job Chaining и (7) Общие рекомендации.

2. Задача #1[1]: популярные домены.

Ваша задача найди TOP-10 самых популярных доменов на основе переходов из каждой социальной сети.

Если на вход предоставлен URL “https://www.zrabatowani.pl/welcome”, то доменной частью будет “www.zrabatowani.pl”. Также, для унификации, предлагается откидывать префикс "www.". Таким образом, для указанного URL, доменом будем считать "zrabatowani.pl".

Входные данные

URL stats:

- Путь на кластере: полный датасет - **/data/socnet_urls**
- Формат: текст
- В каждой строке:
  - `URL <tab> число_переходов_из_социальных_сетей в формате social_network1:transition_count1;social_network2:transition_count2;...`

Пример:

```
https://zveromonstr.ru/statistic.html <tab> facebook:3569;reddit:27
```

Выходные данные

формат вывода (HDFS):

```
social_network <tab> domain <tab> число_переходов
```

Вывод на печать (STDOUT):

вывести TOP-10 доменов для каждой социальной сети в порядке vk, facebook, odnoklassniki, twitter, reddit

Пример вывода:

```
vk pikabu.ru 3353546
vk tvzavr.ru 2909927
```

[1] Внутренний ID задачи (для проверяющих) - 115

```
vk ria.ru 2301713
...
facebook incrivel.club 13883799
facebook perfecto.guru 10763323
...
twitter actualidad.rt.com 803542
twitter life.ru 681870
...
```

3. Задача #2[2]: популярные биграммы.

Найдите самые популярные биграммы (пары слов), которые встретилась в наибольшем числе различных статей Википедии. Слова приведите к нижнему регистру.

Ограничения:

- удалите все знаки пунктуации;
- отсортируйте биграммы по числу встречаемости, т.е. пары (bigram, counts) нужно отсортировать по значению (counts);
- если встречаются пары (bigram, counts) с одинаковым значением “counts”, то отсортируйте биграммы лексикографически (сначала "aaa", затем "bbb").

Входные данные

Wikipedia:

- Путь на кластере: `/data/wiki/en_articles`
- Семпл (для тестирования): **`/data/wiki/en_articles_part`**
- Формат: текст;
- В каждой строке: идентификатор_статьи <tab> текст_статьи

Выходные данные

формат вывода (HDFS):

`word_1` **`<space>`** `word_2 <tab>` число_документов_с_указанной_биграммой

Вывод на печать (STDOUT):

вывести TOP-10 биграмм

[2] Внутренний ID задачи (для проверяющих) - 116

Пример вывода:

```
and the 3495
on the 3326
by the 3250
```

4. Задача #3[3]: популярные теги stackoverflow.

На основе выборки из постов stackoverflow необходимо найти TOP-10 самых популярных тегов, которые люди проставляли в 2010 и в 2016 годах (соответственно).

Ограничения:

- из тегов удалить ненужные html-символы < и >. Например, если на входе Tags="<html><browser><timezone>", то тегами будут html, browser и timezone;
- Тройки (year, tag, counts) отсортировать сначала по году (по возрастанию), затем по counts (по убыванию).

Входные данные

Stackoverflow:

- Путь на кластере: `/data/stackexchange/posts`
- Семпл (для тестирования): **`/data/stackexchange_part/posts`**
- Формат: XML;
- Необходимо рассматривать только строчки, начинающиеся на "<row" (в начале строки могут быть еще пробельные символы)

Пример:

```
<row Id="13" PostTypeId="1" AcceptedAnswerId="357"
CreationDate="2008-08-01T00:42:38.903" Score="440" ViewCount="128370"
Body="<p>Is there any standard way for a Web Server to be able to
determine a user's timezone within a web page? Perhaps from a HTTP
header or part of the user-agent string?</p>" OwnerUserId="9"
LastEditorUserId="3604745" LastEditorDisplayName="Rich B"
LastEditDate="2016-11-29T02:17:23.667"
LastActivityDate="2016-11-29T02:17:23.667" Title="Determine a User's
Timezone" Tags="<html><browser><timezone><timezoneoffset>"
AnswerCount="24" CommentCount="3" FavoriteCount="120" />
```

Выходные данные

[3] Внутренний ID задачи (для проверяющих) - 117

формат вывода (HDFS):

`year <tab> tag <tab>` число_постов_с_указанным_тегом_в_заданный_год

Вывод на печать (STDOUT):

вывести TOP-10 тегов для каждого года, сначала для 2010, затем - для 2016.

Пример вывода (посчитан на подвыборке Stackoverflow):

```
2010 .net 2139
2010 asp.net 2041
2016 javascript 9263
2016 java 7435
2016 python 6183
```

5. Критерии оценивания.

Балл за задачу складывается из:

- **60%** - правильное решение задачи
- **20%** - поддерживаемость и читаемость кода (Clean Code, см. например Google Python Style Guide)
- **20%** - эффективность решения

Штрафы:

- **10%** за несоответствие правилам оформления задания
- **30%** за просрочку дедлайн

6. Job Chaining.

Пример запуска связанных MapReduce задач (Job Chaining), представлен, ниже.

run.sh (обратите внимание на конструкцию `"( ... && ... ) || echo 'smth' "`):

```
#!/usr/bin/env bash
set -x

HADOOP_STREAMING_JAR=/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/had
oop-streaming.jar
OUT_DIR="streaming_wc_result"
NUM_REDUCERS=8

hdfs dfs -rm -r -skipTrash ${OUT_DIR}* > /dev/null
```

```
# Wordcount
( yarn jar $HADOOP_STREAMING_JAR \
 -D mapreduce.job.name="Streaming WordCount" \
 -files count_mapper.py,sum_reducer.py \
 -mapper "python3 count_mapper.py" \
 -reducer "python3 sum_reducer.py" \
 -numReduceTasks $NUM_REDUCERS \
 -input /data/wiki/en_articles_part \
 -output ${OUT_DIR}_tmp > /dev/null &&

# Global sorting as we use only 1 reducer
yarn jar $HADOOP_STREAMING_JAR \
 -D
mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.l
ib.partition.KeyFieldBasedComparator \
 -D mapreduce.map.output.key.field.separator=. \[4]
 -D mapred.text.key.comparator.options="-k2,2nr" \
 -mapper cat \
 -reducer cat \
 -numReduceTasks 1 \
 -input ${OUT_DIR}_tmp \
 -output ${OUT_DIR}
> /dev/null ) || echo "Error happens"

hdfs dfs -rm -r -skipTrash ${OUT_DIR}_tmp > /dev/null

hdfs dfs -cat ${OUT_DIR}/part-00000 | head
```

Для удобства копирования run.sh, count_mapper.py и sum_reducer.py доступны по адресу:

/home/aadral/public_examples/job_chain

7. Общие рекомендации.

Чтобы быть уверенным, что Grader (скрипт оценки решения) правильно обработает ваше решение, предлагаем следующие рекомендации:

[4] Если у вас разделитель <tab> и вы хотите использовать 2 колонки для ключа, то указанную строчку нужно заменить на:

```
-D stream.num.map.output.key.fields=2 \
```

- Для временных данных используйте HDFS-папку с суффиксом _tmp (например `my_hdfs_folder_tmp`)
- Убедитесь, что вы удаляете все временные данные после завершения выполнения задачи
- Отслеживайте код возврата MapReduce задач (Job'ов). В случае ошибки первой задачи в цепочке **нет** необходимости запускать следующие.
- Обращайте внимание на вывод в "STDOUT". Его форматирование является критически важным для прохождения тестов. Формат должен соответствовать выходному HDFS-формату. Вам нужно прочитать ровно столько строчек в STDOUT из HDFS, сколько указано в задании.
- Вы **НЕ** можете прочитать весь HDFS output в RAM для сортировки (даже если получится с игрушечными примерами на нашем кластере, в бою это будет больно отстреливать в ногу).
- В реальной жизни вы можете использовать как абсолютные, так и относительные пути. Для сдачи домашних заданий необходимо использовать относительные пути. Это необходимо для автоматического скрипта проверки решений, чтобы иметь возможность подменить пути во время тестирования финального решения (поскольку наш пользователь *Big Datych* не имеет доступа на чтение / запись в вашу личную HDFS директорию). На примерах:

Пример абсолютного пути (НЕ рабочий вариант): `/user/mf_botvinkin/some_folder`
Пример относительного пути (рабочий вариант): `some_folder`

8. Сроки сдачи и правила оформления задания.

Deadline: 24.06.2019, 08:00

Оформление задания:

- Код задания (Short name): **HW3:MR-advanced**.
- Решение задания должно содержаться в одной папке.
- Скрипт для запуска решения называется **run.sh**:
  - скрипт читает данные из HDFS-папки, указанной в задаче
  - скрипт сохраняет данные в HDFS папку hw3_mr_advanced_output

    - скрипт предварительно очищает результирующие папки в HDFS, чтобы туда можно было записать результаты вычислений
    - скрипт выводит на экран (STDOUT) указанное в задание число строк в нужном формате[5]
    - вывод STDOUT сохраните в файл hw3_mr_advanced_output.out
- Выполненное ДЗ запакуйте в архив MF2019Q2_<фамилия>_HW#.zip , к примеру -- MF2019Q2_Ivanov_HW3.zip . Например, ваше решение лежит в папке my_solution_folder, тогда чтобы на Linux и Mac OS создать архив под названием hw.zip и пожать его с помощью zip выполните команду[6]:
    - zip -r hw.zip my_solution_folder/

  На Windows 7/8/10: необходимо нажать правую кнопку мыши на директорию my_solution_folder/, выбрать в открывшемся меню "Отправить >", затем "Сжатая ZIP-папка". Теперь можно переименовать архив.
- Присылайте выполненное задание на почту bigdata_mf2019q2@bigdatateam.org с темой письма “Short name. ФИО.”. Например: “HW3:MapReduce-advanced. Иванов Иван Иванович.”
- Перед отправкой письма, оставьте, пожалуйста, отзыв о домашнем задании по сссылке: http://rebrand.ly/mf2019q2_feedback_hw03. Это позволит нам скорректировать учебную нагрузку по следующим заданиям (в зависимости от того, сколько часов уходит на решение ДЗ), а также ответить на интересующие вопросы.

Любые вопросы / комментарии / предложения можно писать:

- в телеграм-канале: http://rebrand.ly/mf2019q2_telegram_join
- На почту: bigdata_mf2019q2@bigdatateam.org

9. Дорешка.

Решения после получения фидбека на решение ДЗ можно улучшить. Разрешается одна досылка в течение 1й недели после окончания дедлайна за ДЗ. Соответственно, фидбек за дорешенные ДЗ вы получите в течение 24 часов после окончания deadline следующего ДЗ.

Дорешивать неработающие задания - нельзя. Это позволит исправить дисбаланс между

присланными **НЕработающими** заданиями **ДО** deadline

VS

присланными **работающими** заданиями **ПОСЛЕ** deadline

[5] См. hdfs dfs -cat

[6] Флаг -r значит, что будет совершен рекурсивный обход по структуре директории

Всем удачи!