BIGDATA TEAM

HW #03: MapReduce-advanced

автор задания: BigData Team, коллективная работа.

редактор задания[1]:
- Алексей Казюлин
- Big Data Mentor @ BigData Team

[1] Хочешь стать ментором и оставить след в истории Big Data? Тогда хорошо учись, помогай другим и дай нам знать о своем желании. Смело пиши автору задания или менеджеру учебного курса.

1. Описание задания

Представим, что вашим коллегам из отдела маркетинга срочно нужна информация о ТОП-10 тегах StackOverflow за выбранные года. Данные на HDFS, в память локальной машины не помещаются, а из инструментов сейчас доступен только Map-Reduce, вся надежда на вас!

В данном ДЗ нужно решить 1 задачу. Решение надо выполнить на Hadoop Streaming.

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

1.1. 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" />
```

1.2. Выходные данные

формат вывода (HDFS и STDOUT):

```
year <tab> tag <tab> число_постов_с_указанным_тегом_в_заданный_год
```

И в HDFS, и в 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
```

1.3. Требования к реализации

- вы **НЕ** можете прочитать весь HDFS output в RAM для сортировки.
  Запрещены конструкции вида: hdfs dfs -text *** | sort ***
  Даже если получится с игрушечными примерами на нашем кластере, в бою это будет больно отстреливать в ногу;
- 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).
- использовать как минимум 2 из 3-х оптимизаций: combiner, partitioner, comparator
- реализовать фильтр по году на первой map-стадии
- реализовать фильтры по TOP-10 тегов в последней reduce-стадии
- использовать больше, чем 1 reducer (1 reducer разрешается использовать только в финальной job'e, при сортировке результата)

Скрипт для запуска решения должен называться **run.sh**:

- скрипт использует следующий путь до hadoop-streaming.jar на кластере: `/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar`
- скрипт будет запускаться с помощью команды:
  `bash run.sh $(input_ids_hdfs_path) $(output_hdfs_path) $(job_name)`
- скрипт читает данные из HDFS-папки, указанной первым аргументом (используйте `$1` в `run.sh`)
- скрипт сохраняет данные в HDFS папку `$2`
- скрипт очищает все временные директории в HDFS до и после запуска вычислений. Для временных данных используйте HDFS-папку с суффиксом _tmp (например my_hdfs_folder_tmp)

- обращайте внимание на вывод в "STDOUT". Его форматирование является критически важным для прохождения тестов. Формат[2] должен соответствовать выходному HDFS-формату. Вам нужно прочитать ровно столько строчек в STDOUT из HDFS, сколько указано в задании;
- вывод STDOUT сохраните в файл **`hw03_mr_advanced_output.out`** и приложите к архиву с решением
- в заголовке bash-скрипта указана опция “`set -x`”, вывод STDERR никуда не перенаправляется (он используется для анализа логов исполнения задачи)

2. Job Chaining

Пример запуска связанных MapReduce задач (Job Chaining), представлен в run.sh доступному на github:
github:big-data-team/big-data-course/.../map_reduce/job_chain/run_job_chain.sh

Обратите внимание на конструкцию "`( ... && ... ) || echo 'smth' `", которая позволяет отлавливать исключительные события и не запускать зависимую задачу, если первая не отработала.

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

`/usr/local/share/big_data_course/public_examples/job_chain`

А также на Github курса:
github:big-data-team/big-data-course/.../map_reduce/job_chain

[2] См. hdfs dfs -cat

3. Критерии оценивания

Балл за задачу складывается из:

- **60%** - правильное решение задачи
- **20%** - поддерживаемость и читаемость кода
  - в общем случае см. Clean Code и Google Python Style Guide
  - оценка качества будет проводиться автоматическим вызовом pylint:
    - `pylint *.py -d invalid-name,missing-docstring`
    - качество кода должно оцениваться выше 8.0 / 10.0
    - проверяем код **Python версии 3** с помощью `pylint==2.5.3`
- **20%** - эффективность решения (для сравнения: решение должно отрабатывать[3] в течение 5 минут на ресурсах 3-х вычислительных узлов.

Discounts (скидки и другие акции):

- **100%** за плагиат в решениях (всем участникам процесса)
- **100%** за посылку решения после hard deadline
- **30%** за посылку решения после soft deadline и до hard deadline
- **5%** за каждую дополнительную посылку в тестирующую систему (всего можно делать до 3-х посылок без штрафа):

Пример работы системы штрафов:

День	Посылка	Штраф
День 1	Посылка 1	Без штрафа
День 1	Посылка 2	Без штрафа
День 1	Посылка 3	Без штрафа
День 1	Посылка 4	-5%
День 2	Посылка 5	-5%
День 3	Посылка 6	-5%
Итоговый штраф: -15%		

Для подсчета финальной оценки **всегда** берется **последняя** оценка из Grader.

[3] Оценка производится на основе счетчика “CPU time spent (ms)”

4. Инструкция по отправке задания

Перед отправкой задания оставьте, пожалуйста, отзыв о домашнем задании по ссылке: https://rebrand.ly/bdmade2022q2_feedback_hw. 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.

Оформление задания:

- Код задания (Short name): **HW03:MR_advanced**
- Выполненное ДЗ запакуйте в архив **BD-MADE-2022-Q2**_<Surname>_<Name>_HW#**.zip**, пример -- **BD-MADE-2022-Q2**_Dral_Alexey_**HW03**.zip. (Проверяйте отсутствие пробелов и невидимых символов после копирования имени отсюда.[4]) Если ваше решение лежит в папке my_solution_folder, то для создания архива hw.zip на Linux и Mac OS выполните команду[5]:
  - `zip -r hw.zip my_solution_folder/*`
- На Windows 7/8/10: необходимо выделить все содержимое директории my_solution_folder/ нажать правую кнопку мыши на одном из выделенных объектов, выбрать в открывшемся меню "Отправить >", затем "Сжатая ZIP-папка". Теперь можно переименовать архив.
- Решение задания должно содержаться в одной папке.
- Перед проверкой убедитесь, что дерево вашего архива выглядит так:
  - **| BD-MADE-2022-Q2_<Surname>_<Name>_HW03.zip**
  - **| ---- run.sh**
  - **| ---- *.py**
  - **| ---- hw03_mr_advanced_output.out**
  - При несовпадении дерева вашего архива с представленным деревом, ваше решение будет невозможно автоматически проверить, а значит, и оценить его.
- Для того, чтобы сдать задание, необходимо:
  - Зарегистрироваться и залогиниться в сервисе Everest
  - Перейти на страницу приложения: MADE BigData Grader
  - Выбрать вкладку Submit Job (если отображается иная).
  - Выбрать в качестве “Task” значение:**HW03:MR_advanced**[6]
  - Загрузить в качестве “Task solution” файл с решением
  - В качестве Access Token указать тот, который был выслан по почте

[4] Онлайн инструмент для проверки: https://www.soscisurvey.de/tools/view-chars.php
[5] Флаг -r значит, что будет совершен рекурсивный обход по структуре директории
[6] Сервисный ID: map_reduce.stackoverflow

Внимание: Если до дедлайна остается меньше суток, и Вы знаете (сами проверили или коллеги сообщили), что сдача решений сломана, обязательно сдайте свое решение, прислав нам ссылку на выполненное задание (Job) на почту с темой письма “Short name. ФИО.”. Например: “HW03:MR_advanced. Иванов Иван Иванович.”
Пример ссылки: https://everest.distcomp.org/jobs/67893456230000abc0123def
Чтобы мы видели, какое решение Вы имели до дедлайна и смогли его оценить.

Любые вопросы / комментарии / предложения можно писать в телеграм-канал курса или на почту bd_made2022q2@bigdatateam.org.

Всем удачи!

5. FAQ (часто задаваемые вопросы)

"You are not allowed to run this application", что делать?
Если Вы видите надпись "You are not allowed to run this application" во вкладке Submit Job в Everest, то на данный момент сдача закрыта (нет доступных для сдачи домашних заданий, по техническим причинам или другое). Попробуйте, пожалуйста, еще раз через некоторое время. Если Вы еще ни разу не сдавали, у коллег сдача работает, но Вы видите такое сообщение, сообщите нам об этом.

Grader показывает 0 или < 0, а отчет (Grading report) не помогает решить проблему
Ситуации: • система оценивания показывает оценку (Grade) < 0, а отчет (Grading report) не помогает решить проблему. Пример: в случае неправильно указанного access token система вернет -401 и информацию о том, что его нужно поправить; • система показывает 0 и в отчете (Grading report) не указано, какие тесты не пройдены. Пример: вы отправили невалидный архив (rar вместо zip), не приложили нужные файлы (или наоборот приложили лишние - временные файлы от Mac OS и т.п.), рекомендуется проверить содержимое архива в консоли: `unzip -l your_solution.zip` Если Вы столкнулись с какой-то из них, присылайте ссылку на выполненное задание (Job) в чат курса. Пример ссылки: https://everest.distcomp.org/jobs/67893456230000abc0123def

Что в отчете Grader означает проверка X ?
Как читать отчет: Для каждого теста • Raw_score - балл за конкретный тест. Может быть как бинарным (1\0), так и находиться в интервале от 0 до 1 • Score - Raw_score*weight (вес теста в общей оценке). Вес указан для каждого теста ниже

Итоговая оценка: смотрите строку Score (сумма Score всех индивидуальных тестов) внизу отчета.

Правильность решения задачи:
Test_unzip_is_succesful (weight = 0) - ДЗ заархивировано в .zip архив и грейдер может его разархивировать

Test_map_reduce_execution_is_successful (weight = 0.01) - map-reduce задача выполнилась без ошибок

Test_run_output_contains_expected_line_count (weight = 0.02) - run.sh выводит из результата map-reduce задачи ожидаемое кол-во строк

Map_reduce_update_hdfs_destination (weight = 0.02) - map-reduce задача записывает результаты на HDFS

Test_each_line_of_run_stdout_match_regexp (weight = 0.15) - каждая строчка соответствует формату вывода

Test_no_local_grep_or_sort (weight = 0.15) - локальные консольные утилиты не использовались для фильтрации и сортировки. Отсутствуют такие конструкции
Hdfs dfs - cat /path/to/file/*** | sort ****

Test_exact_run_stdout_comparison (weight = 0.25) - run.sh выводит ожидаемые значения

Поддерживаемость и читаемость кода:
test_py_files_min_lint_score (weight = 0.2) - качество кода в .py файлах оценивается выше 8.0

Эффективность решения:
Test_at_least_2_out_of_3_map_reduce_optimizatons (weight = 0.1) - использовались хотя бы две различные оптимизации

test_solution_has_at_least_2_stages (weight = 0.01) - в цепочке хотя бы две map-reduce задачи

Test_first_job_has_enough_reducers (weight = 0.03) - фаза reduce происходит в распределенном режиме

Test_solution_calculate_all_stats (weight = 0.03) - map-reduce задача выводит ожидаемые значения

test_has_expected_cpu_time_performance (weight = 0.03) - map-reduce задача выполняется не оптимально. Обратите внимание на фильтрацию данных и на то, на каких стадиях она происходит. Скорее всего вы “таскаете” за собой много лишних данных, что усложняет стадии “Shuffle & Sort”.
Скорость выполнения должна быть ниже 1,5 млн CPU-ms (25 CPU-min)