BIGDATA TEAM

HW #04: Hive

автор задания: BigData Team, коллективная работа.

редакторы задания[1]:

- Ксения Пеньевская**, Александр Ким, Николай Попов*
- Big Data Mentor @ BigData Team
- *Data Engineer @ inDriver
- **Big Data Analyst

[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 преподавателям и менеджерам учебных курсов.

1. Описание задания

Сегодня к вам пришёл менеджер с просьбой показать статистику по читателям новостных сайтов. Как оказалось, сырые логи есть, но нужных Hive таблиц для аналитики нет. Ваша задача - это исправить.

В данном ДЗ нужно решить **3 задачи**. Решение надо выполнить с помощью Hive.

1.1. Описание данных

Логи запросов пользователей новостных сайтов.

`logs_raw`:

- Путь на кластере: полный датасет - **`/data/user_logs/user_logs_M`**
- Семпл (для тестирования): **`/data/user_logs/user_logs_S`**
- Формат: текст
- В каждой строке находятся следующие поля, разделенные знаком табуляции (**иногда не одним**):
    1. `ip STRING` - ip-адрес, с которого пришел запрос,
    2. `date STRING` - время запроса,
    3. `request STRING` - пришедший с ip-адреса http-запрос,
    4. `page_size INT` - размер переданной клиенту страницы в байтах,
    5. `http_status INT` - http-статус запроса.
    6. `user_agent STRING` - User Agent, информация о клиентском приложении, с которого осуществлялся запрос на сервер, в том числе информация о браузере.

Пример:

```
135.124.143.193              20150601013300
http://newsru.com/4712386    235   412   Firefox/5.0 (compatible; MSIE
9.0; Windows NT 6.1; Win64; x64; Trident/5.0)n
```

Важно:

- разделитель между IP и временем запроса состоит из 3 символов табуляции;
- Будем считать, что информация о браузере содержится в начале 6-ого поля лога - символы с нулевой позиции до позиции первого пробельного символа.
    - пример User Agent:
    - `Chrome/5.0 (compatible; MSIE 9.0; Windows NT 8.0; WOW64; Trident/5.0; .NET CLR 2.7.40781; .NET4.0E; en-SG)`
    - тогда браузером будет: `Chrome/5.0`

Подсказка:

- поскольку нас не интересует оставшаяся часть User Agent, то получить тип браузера пользователя можно с помощью правильного регулярного выражения в период чтения `logs_raw`.
- шаблон регулярного выражения должен описывать строку целиком (до символа '\n')

Информация о пользователях.

`users`:

- Путь на кластере: полный датасет - **`/data/user_logs/user_data_M`**
- Семпл (для тестирования): **`/data/user_logs/user_data_S`**
- Формат: текст
- В каждой строке находятся следующие поля, разделенные знаком табуляции:
    1. ip STRING - IP-адрес, с которого пользователь выходит в интернет;
    2. browser STRING - браузер пользователя;
    3. sex STRING - пол (male / female);
    4. age INT - возраст.

Пример:

```
197.72.248.141    Opera/12.0  male  30
```

Геобаза - информация о соответствии ip-адресов регионам.

`ip_regions`:

- Путь на кластере: полный датасет - **`/data/user_logs/ip_data_M`**
- Семпл (для тестирования): **`/data/user_logs/ip_data_S`**
- Формат: текст
- В каждой строке находятся следующие поля, разделенные знаком табуляции:
    1. ip STRING - IP-адрес;
    2. region STRING - регион.

Пример:

```
33.49.147.163     Kemerovo Oblast
197.72.248.141    Belgorod Oblast
135.124.143.193   Krasnoyarsk Krai
...
```

1.2. Задача 1: создание таблиц в Hive (Task ID: hive.ddl)

Создайте внешние (EXTERNAL) таблицы по исходным данным:

1. **`logs_raw`** - логи пользователей;
2. **`users`** - таблица с информацией о пользователях;
3. **`ip_regions`** - таблица с IP и регионами;

Из таблицы логов перенесите данные в другую таблицу, партиционированную по датам – одна партиция на каждый день:

4. `logs` - партиционированная таблица с логами.

Условия:

1. Таблицы и поля должны называться ровно так, как указано в описании задачи. Например, поле для хранения даты (дня) в таблице `logs` нужно оставить таким же, как и в `logs_raw`:
   - `` `date` STRING;[2] ``
2. Сериализация и десериализация данных для таблицы `logs_raw` должна осуществляться с использованием регулярных выражений, см.:
   - **org.apache.hadoop.hive.serde2.RegexSerDe**

Проверить правильность создания таблиц можно с помощью простых SELECT-запросов:

```
SELECT * FROM <table> LIMIT 10
```

Рекомендации:

- предлагается начать с простых таблиц, а потом двигаться к сложным, например: `ip_regions` → `users` → `logs_raw` → `logs`;
- при создании external таблиц не требуется указывать опцию location, нужно использовать путь по умолчанию;
- для создания таблиц `ip_regions` и `users` рекомендуется воспользоваться следующей конструкцией:
  - `ROW FORMAT delimited`

[2] Обратите внимание на экранирование ключевых слов Hive

- Документация по полям, разделяющим колонки, доступна по адресу. Вам необходимо найти способ указать разделитель <tab> вместо стандартного разделителя ^A.

Подсказки по созданию партиционированной таблицы logs:

1. Чтобы выделить день в формате “YYYYMMDD”, достаточно воспользоваться функцией для работы со строками SUBSTR.
2. Посчитайте, сколько уникальных (DISTINCT) дней в “сырых” логах (logs_raw). Это число должно получиться более 100 на датасете размера “_M”.
3. Используйте это число, чтобы задать переменную окружения Hive, которая позволит запустить динамическое создание партиций[3]:
   - `set hive.exec.max.dynamic.partitions.pernode=***;`
4. После этого можно написать запрос:
   - `INSERT OVERWRITE TABLE logs PARTITION(date) SELECT ... FROM logs_raw`

На партиционированной таблице `logs` и нужно будет выполнять запросы в следующих задачах.

1.3. Задача 2: горячий денек (Task ID: hive.hot_day)

Напишите запрос, который считает, какое количество посещений новостных сайтов было в разрезе дней. Полученные результаты отсортируйте (**ORDER BY**) по убыванию популярности. На экран выведите TOP-10 самых “горячих” дней с точки зрения нагрузки на инфраструктуру новостных сервисов в формате:

- день <tab> число посещений

Пример вывода:

```
20140308 96
20140409 94
20140318 89
...
```

Для этого задания таблица logs будет предоставлена, поэтому если вы используете названия колонок, которые не соответствуют схеме из раздела “3. Описание данных”, то Grader не пропустит решение.

[3] Подробную документацию по dynamic partitioning см. здесь: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML#LanguageManualDML-DynamicPartitionInserts

1.4. Задача 3: identify browser sex (Task ID: hive.sex_browser)

Напишите запрос, который считает число употреблений браузера мужчинами и женщинами. Считаем статистику по таблице **logs**, но информацию о браузере берем из таблицы **users**[4]. Выведите **произвольные** 10 записей (LIMIT 10) в формате:

- браузер `<tab>` посещаемость мужчинами `<tab>` посещаемость женщинами

Пример вывода:

```
MsExplorer/11.0 1419872 621124
Chrome 1426114 623333
...
```

Подсказки:

- для решения задачи рекомендуется воспользоваться оператором IF, примеры его использования см. в официальной документации Hive (см. здесь) или в слайдах занятия.
- для решения этой задачи нужно сделать join двух таблиц. Сложность заключается в том, что по умолчанию, из-за небольшого объема данных Hive преобразует этот запрос в Map-Side Join, НО у него **может** не хватить оперативной памяти, чтобы выполнить эту задачу, поэтому:
    1. Нужно отключить авто-конвертацию join в оптимизированный вид join. см. опцию:
        - `set hive.auto.convert.join`
    2. Из-за небольшого объема данных, Hive может запустить все вычисления в рамках Reduce-Side Join на одном редьюсере. Чтобы этого избежать, необходимо изменить число редьюсеров с помощью флага:
        - `set mapreduce.job.reduces`

Для этого задания таблица logs будет предоставлена, поэтому если вы используете названия колонок, которые не соответствуют схеме из раздела 3. Описание данных, то Grader не пропустит решение.

[4] Да, мы согласны, что это глупое предположение и в реальной жизни информацию о браузере нужно брать из User Agent логов. Но мы еще придумаем задачи получше ;)

2. Рекомендации

В задании мы будем пользоваться регулярными выражениями для парсинга входных данных. Для проверки регулярных выражений (regex) рекомендуем пользоваться онлайн regex-чекером: https://regex101.com/

3. Критерии оценивания

Веса задач:

- Задача 1: создание таблиц в Hive (Task ID: hive.ddl) - 33.3%
- Задача 2: горячий денек (Task ID: hive.hot_day) - 33.3%
- Задача 3: identify browser sex (Task ID: hive.sex_browser) - 33.3%

Балл за задачу складывается из:

- **70%** - правильное решение задачи
- **0%** - 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`
- **30%** - эффективность решения (такие как потребляемые CPU-ресурсы, скорость выполнения (в предположении свободного кластера)).

Discounts (скидки и другие акции):

- **100%** за плагиат в решениях (всем участникам процесса)
- **100%** за посылку решения после hard deadline
- **30%** за посылку решения в 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.

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): **HW04:Hive**.
- Выполненное ДЗ запакуйте в архив **BD-MADE-2022-Q2**_<Surname>_<Name>_HW#**.zip**, пример -- **BD-MADE-2022-Q2**_Dral_Alexey_**HW04**.zip. (Проверяйте отсутствие пробелов и невидимых символов после копирования имени отсюда.[5]) Если ваше решение лежит в папке my_solution_folder, то для создания архива hw.zip на Linux и Mac OS выполните команду[6]:
  - `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.
- Название базы данных будет передаваться через CLI с помощью аргумента `"--database=$(db_name)"`, для локальный экспериментов рекомендуется использовать <username>, например **dral**; ваши скрипты **не должны** содержать использование клаузы `"use"`.
- HQL-скрипты для запуска решений следует называть по суффиксу Task ID задачи **task_<Surname>_<Name>_<#task_ID_suffix>.hql**:
  - например решение задачи “hive.hot_day” должно называться task_<Surname>_<Name>_hot_day.hql и его можно запустить с помощью команды:
    - `hive` **`--database=${DB_NAME}`**[7] `-f task_*_hot_day.hql`
  - скрипт выводит на экран (STDOUT) указанное в задании число строк в нужном формате
- **Вывод STDOUT задач просьба сохранить в соответствующих файлах в архиве посылке домашнего задания (например, task_<Surname>_<Name>_<#task_ID>.out).**
- Перед проверкой убедитесь, что дерево вашего архива выглядит так:

[5] Онлайн инструмент для проверки: https://www.soscisurvey.de/tools/view-chars.php
[6] Флаг -r значит, что будет совершен рекурсивный обход по структуре директории
[7] Это означает, что Вы не должны использовать “use <database_name>” внутри скриптов

    - `| BD-MADE-2022-Q2_<Surname>_<Name>_HW04.zip`
    - `| ---- task_<Surname>_<Name>_ddl.hql`
    - `| ---- task_<Surname>_<Name>_hot_day.hql`
    - `| ---- task_<Surname>_<Name>_hot_day.out`
    - `| ---- task_<Surname>_<Name>_sex_browser.hql`
    - `| ---- task_<Surname>_<Name>_sex_browser.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: **HW04:Hive**[8]
    - Загрузить в качестве “Task solution” файл с решением
    - В качестве Access Token указать тот, который был выслан по почте

Внимание: Если до дедлайна остается меньше суток, и Вы знаете (сами проверили или коллеги сообщили), что сдача решений сломана, обязательно сдайте свое решение и напишите письмо, как написано выше, 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!

[8] Сервисный ID: hive.onsite_hw

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 индивидуальны 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