МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 810 «Информационные технологии в моделировании и управлении»

**Отчет по заданию**

**по курсу «Хранение и обработка больших данных»**

**Решение задачи с помощью технологий MapReduce, Hive SQL, Apache Spark**

Выполнил: А.С.Бобряков

Группа: М8O-103М-19

Преподаватель: Е.С.Гаврилов

Москва, 2020

**Задание**

**Найти самых лучших специалистов по Java в России.**

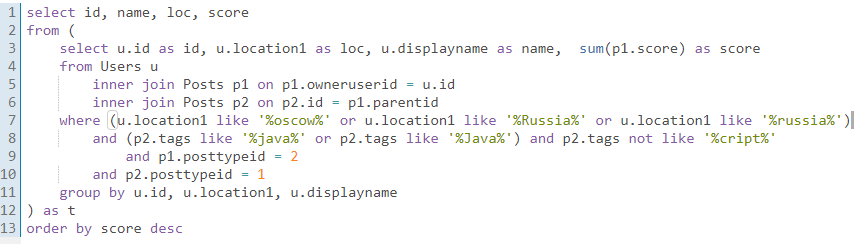
Рекрутеры часто используют специализированные профессиональные сайты для поиска кандидатов с требуемым профилем технологий. Компании нужно найти потенциальных кандидатов в России, которые хорошо знают стек Apache Hadoop.

«Лучшими» специалистами можно считать пользователей с наибольшим рейтингом (Score). В простейшем варианте, можно использовать рейтинг самого пользователя (из Users.xml) и смотреть есть ли у него ответы на вопросы с тегами “Hadoop”. Но туда могут попасть пользователи, которые, например, хорошо отвечали на множество вопросов по Java, а на Hadoop ответили только на 1 вопрос. Более честный вариант был бы только тот, который учитывал рейтинг ответов по тегу “Java”.

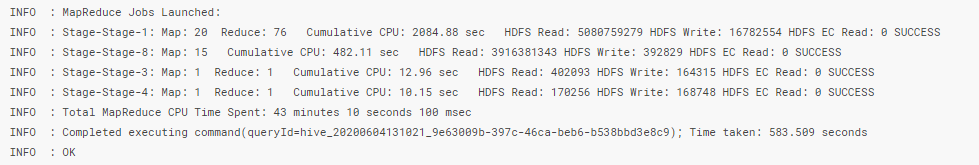
Поле Location в users – текстовое поле со свободным вводом любого текста. Поэтому, для определения того, находится ли пользователь в России, придется провести анализ данных поля Location и определять это по наличию подстрок “Russia”, “Moscow”, “Russian Federation” и др. Для этого можно вывести список уникальных Location, отсортированный по сумме рейтингов и просмотреть глазами верхние N строк, которые покрывают большую часть кандидатов.

**Реализация на Hive SQL**

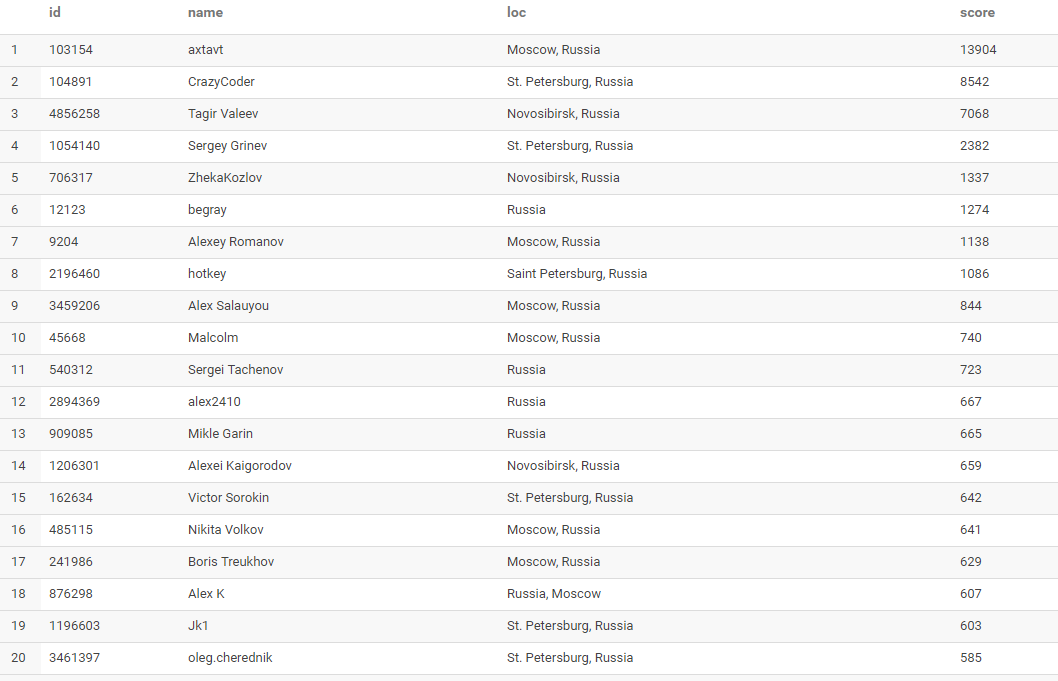
Запрос:



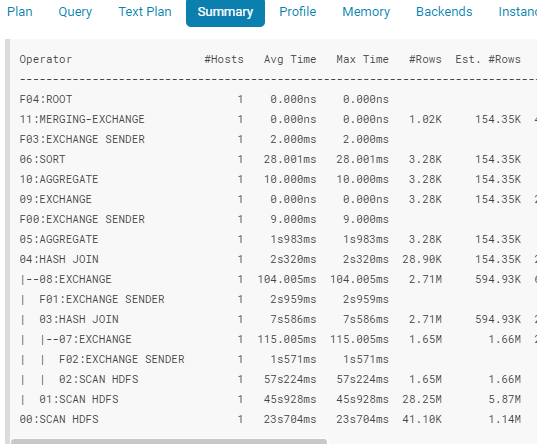
Статистика:



Результат:



Статистика на Impala:



Сравнение результатов на Hive и Impala:

Скорость выполнения запроса на Impala намного выше чем на Hive. В Impala в результирующих логах не отображается статистика, приходится открывать соответствующую job.

**Реализация на MapReduce**

Файл с реализацией приложен к отчету.

Было реализовано две job: для джойна постов-ответов с постами-вопросами (+фильтрации для них) и джойна результата с пользователями (с фильтрами для них), а также вывода топ 100 результата, отсортированного по значению Score.

Результат:

4856258 DisplayName:Tagir Valeev Location:Novosibirsk, Russia Score:4446

103154 DisplayName:axtavt Location:Moscow, Russia Score:3946

104891 DisplayName:CrazyCoder Location:St. Petersburg, Russia Score:1063

548473 DisplayName:GKislin Location:St. Petersburg, Russia Score:931

1054140 DisplayName:Sergey Grinev Location:St. Petersburg, Russia Score:715

3459206 DisplayName:Alex Salauyou Location:Moscow, Russia Score:670

706317 DisplayName:ZhekaKozlov Location:Novosibirsk, Russia Score:656

9204 DisplayName:Alexey Romanov Location:Moscow, Russia Score:642

886697 DisplayName:Stanislav Bashkyrtsev Location:St Petersburg, Russia Score:602

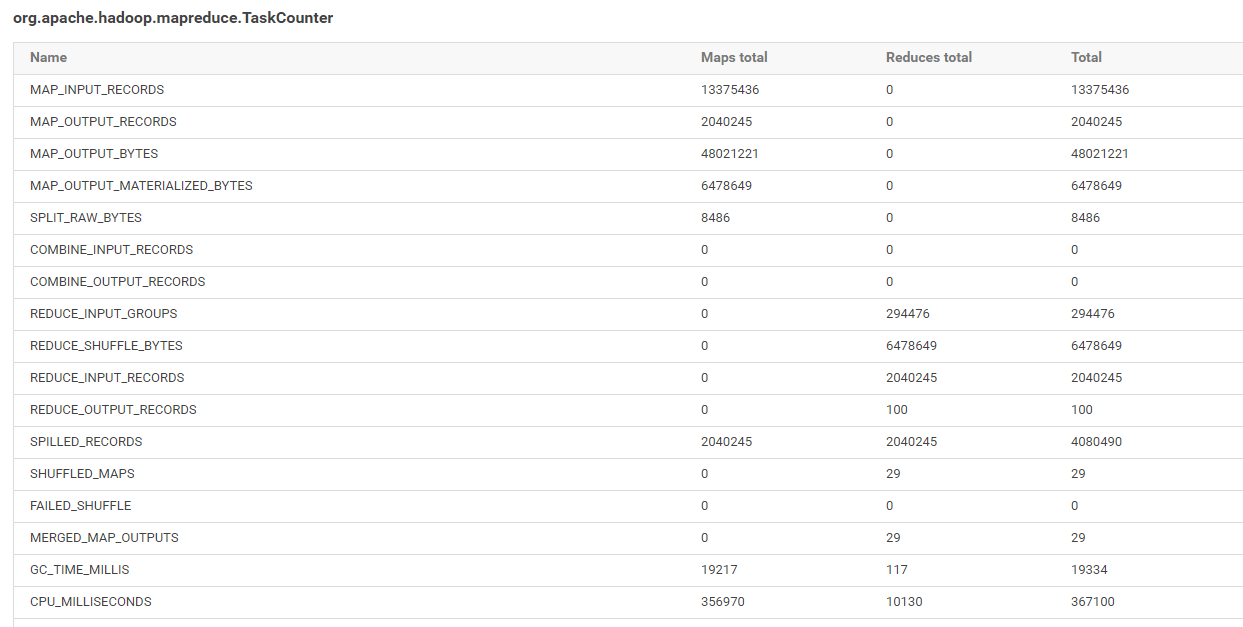
2596109 DisplayName:Oleg Mikhailov Location:St. Petersburg, Russia Score:556

241986 DisplayName:Boris Treukhov Location:Moscow, Russia Score:531

525203 DisplayName:Dmitry Spikhalskiy Location:Moscow, Russia Score:517

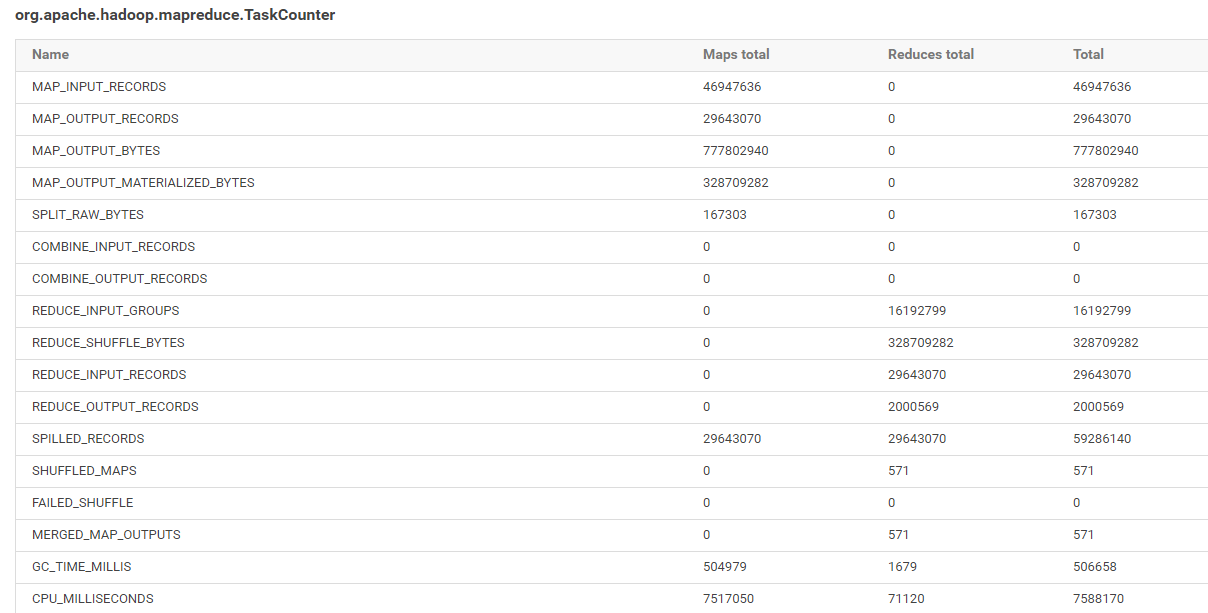
909085 DisplayName:Mikle Garin Location:Russia Score:504

Статистика **Job2**:

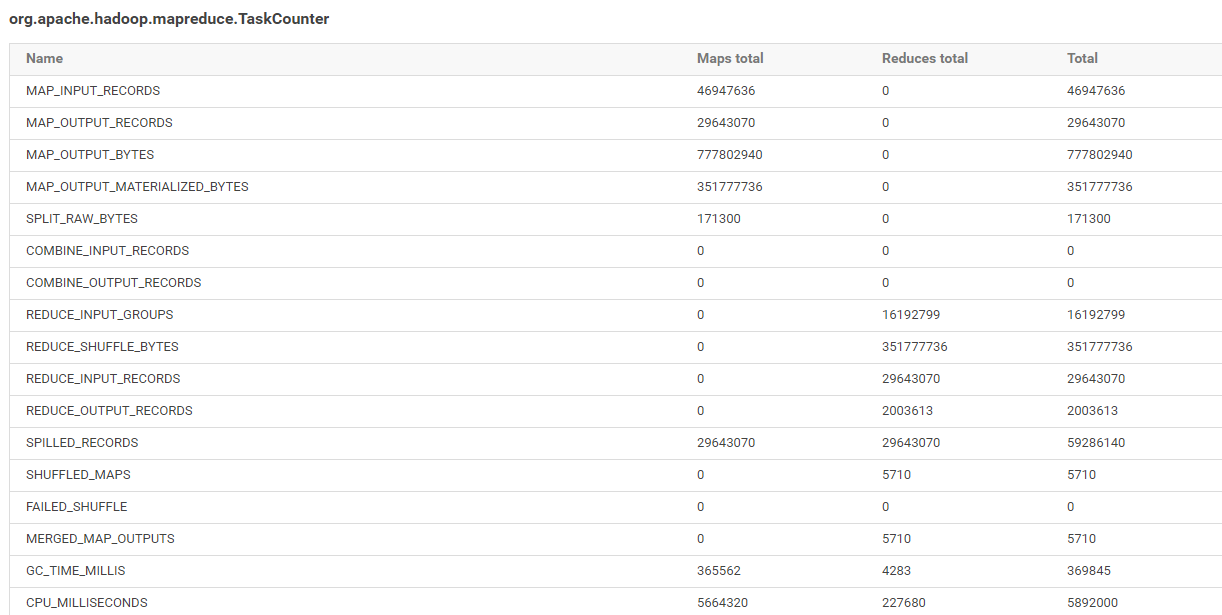


Статистика для различных показателей numReduceTask **Job1**:

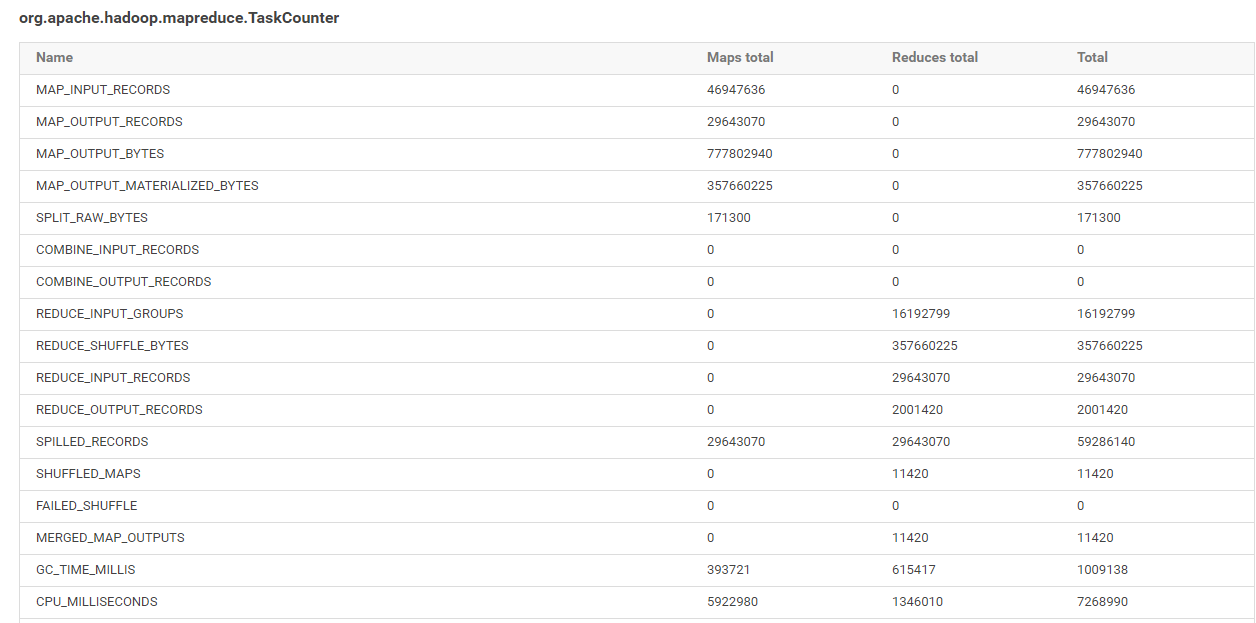
numReduceTask = 1



numReduceTask = 10



numReduceTask = 20



Реализованные варианты оптимизации:

1. Подобрать оптимальное количество редьюсеров по количеству блоков HDFC;
2. Реализация сортировки как topN, вместо всего результата по специфике самой задачи.

Вывод:

Задача на MapReduce выполняется медленнее, чем запрос на Hive SQL (~104 минуты по сравнению с ~40 минутами). Это объясняется тем, что в Hive SQL при выполнении запроса сначала составляет логический план выполнения запроса, который дальше оптимизируется на основе статистик, размеров используемых таблиц, операций в запросе, хинтов прежде чем сгенерировать код на Java и выполнить его.

**Реализация на Apache Spark**

Файл с реализацией приложен к отчету.

Результат:

ID: 4856258 Name: Tagir Valeev Location:Novosibirsk, Russia Score: 5034

ID: 103154 Name: axtavt Location:Moscow, Russia Score: 4595

ID: 706317 Name: ZhekaKozlov Location:Novosibirsk, Russia Score: 2427

ID: 2914140 Name: CoolMind Location:Moscow, Россия Score: 1836

ID: 548473 Name: GKislin Location:St. Petersburg, Russia Score: 1750

ID: 1112963 Name: Zon Location:Perm', Russia Score: 1736

ID: 10445540 Name: Artem Botnev Location:Russia Score: 1700

ID: 104891 Name: CrazyCoder Location:St. Petersburg, Russia Score: 1500

ID: 1206301 Name: Alexei Kaigorodov Location:Novosibirsk, Russia Score: 1208

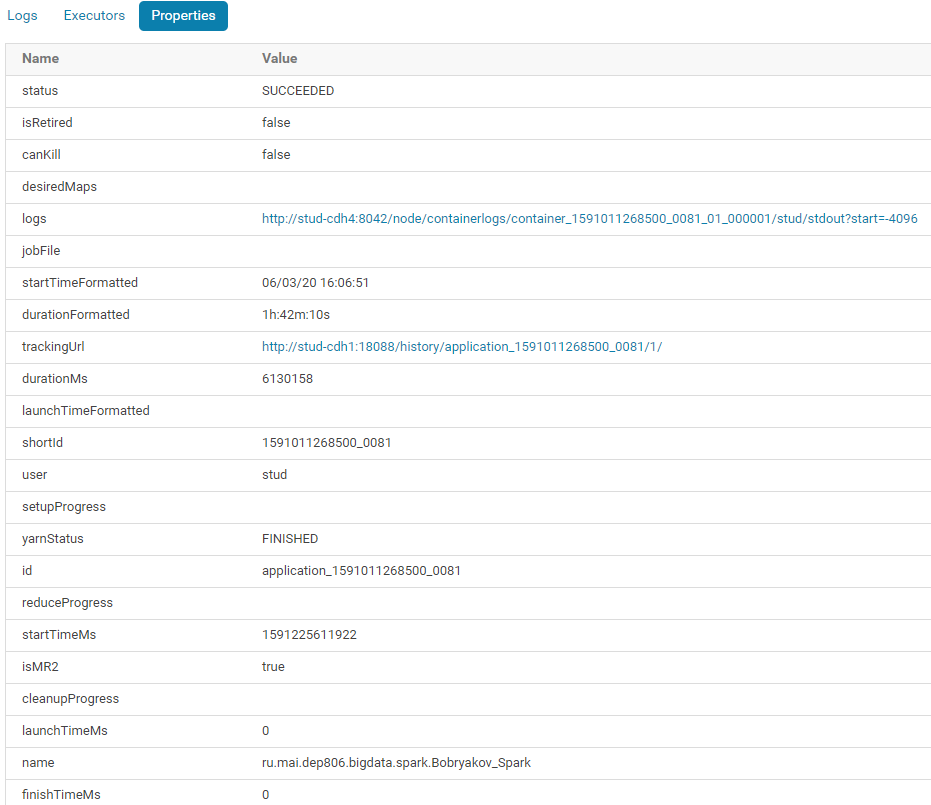
ID: 9204 Name: Alexey Romanov Location:Moscow, Russia Score: 1208

ID: 1054140 Name: Sergey Grinev Location:St. Petersburg, Russia Score: 1169

ID: 11606203 Name: The Rabbit of No Luck Location:Saint Petersburg, Russia Score: 1128

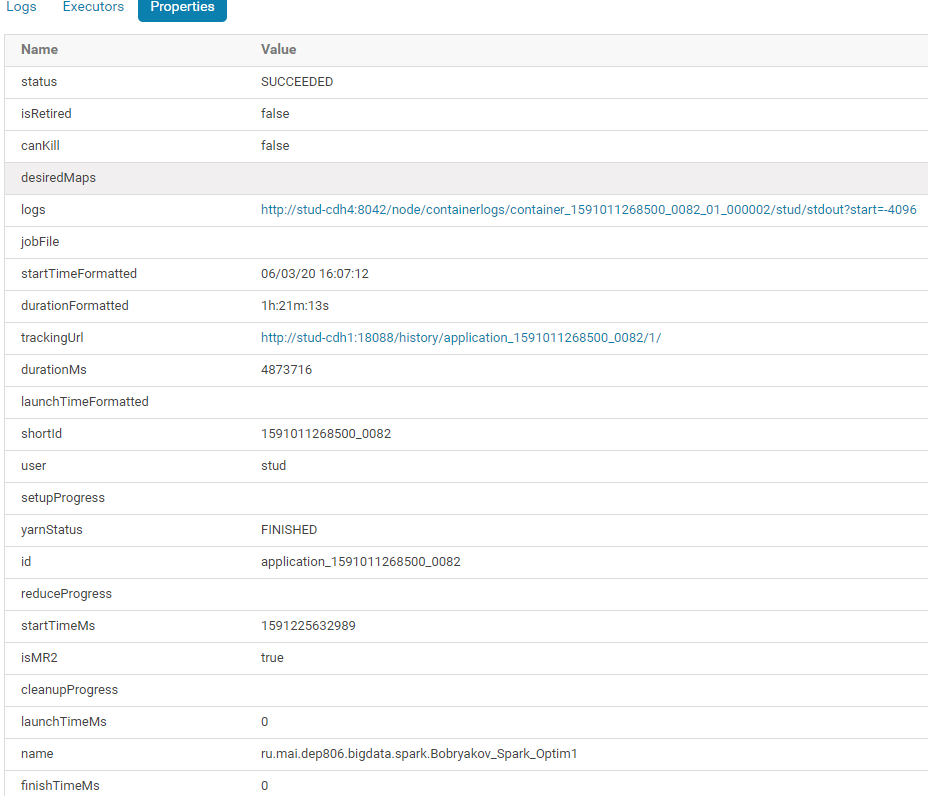
ID: 5554326 Name: Anton Rybochkin Location:Saint Petersburg, Russia Score: 976

Статистика:



Статистика для конфигурации:

sparkConf.set("spark.sql.shuffle.partitions", "16");  
sparkConf  
 // 4 executor per instance of each worker  
 .set("spark.executor.instances", "3")  
 // 5 cores on each executor  
 .set("spark.executor.cores", "3");



Выводы:

Скорость работы на Spark оказалась выше MapReduce, но ниже Hive. Это объясняется тем, что Spark использует оперативную память для операций, пока ее хватает, когда как MapReduce сохраняет данные на диск между фазами и этапами. При том, операции join/агрегирования/сортировки уже реализованы. То есть не нужно терять производительность на их собственноручную реализацию. Но, учитывая наличие оптимизатора запроса в Hive, Spark работает медленнее.

**Возникшие проблемы по всем этапам задания:**

1. MR – невозможность в одной Job обработать один и тот же файл разными мапперами.
2. Hue – иногда не отображает задачу при переходе в нее по клику в истории Job (Could not find job applicantId\_...), приходится перезапускать джобу.
3. MR/Spark – сложно дебажить промежуточные результаты. В итоге результаты разных платформ отличаются по значениям (хотя приблизительно похожи). Ошибку не удалось найти.
4. Spark – в логах не находится статистика: yarn logs -applicationId <application\_id>.
5. Длительное ожидание выполнения задач при работе нескольких других.
6. Для запуска Spark задач приходилось выставлять параметры --driver-memory 2500M --executor-memory 1000M. Иначе файл не обрабатывался: ошибки не падало, которая завершает job, но в логах на Hue можно было найти следующее сообщение:

