Практикум на ЭВМ. Самбурский Александр, 317 группа.
Рекомендательные системы.
Описание решения.

Программная реализация состоит из 6 стадий. Кратко опишем результаты, получаемые каждой из них, алгоритмы для мапперов и редьюсеров, а также сложность по памяти и вычислениям. После этого укажем время работы программы. В конце приведём последовательность команд для запуска программы.

Предварительно сделаем несколько замечаний:

- Для записи алгоритмической сложности стадий используются следующие обозначения:

Обозначение	***Величина***
U	Число пользователей в датасете
I	Число фильмов в датасете
alpha	Средняя доля просмотренных фильмов пользователем
M	Число мапперов в стадии
R	Число редьюсеров в стадии

- Для удобства при описании работы MapReduce-задач используется следующее обозначение передаваемых данных (в терминологии ключ-значение): (***key***: <key>; ***value***: <value>). Например, передача тройки “[user]\t[item] [rating]”, где user – ключ, [item] [rating] – значение, \t – сепаратор, будет обозначаться так: (***key***: user; ***value***: item, rating).

- На некоторых стадиях мапперы/редьюсеры могут принимать вспомогательные файлы. Данные файлы передаются в стадию отдельно от основного потока ввода и позволяют предварительно считать полезную информацию для последующего использования во время обработки основного потока ввода. Как правило, эти файлы занимают небольшой объём памяти (не более 3 Мб), поэтому передача и считывание этих файлов практически не накладывает вычислительную нагрузку на систему. Их использование обусловлено повышением эффективности и наглядности некоторых стадий алгоритма. Далее словосочетание «вспомогательный файл» будет обозначать именно такие файлы, а одно слово «файл» - информационные данные для стандартного потока ввода для MapReduce-задач.

- В вычислительную сложность мапперов включена операция вывода (она играет определяющую роль во времени работы программы)

1 Стадия

Получение средних пользовательских рейтингов.

1. Маппер принимает на вход файл ratings.csv и передаёт (во время считывания) в редьюсер пары (***key***: user; ***value***: rating), что позволит их сгруппировать по пользователям и усреднить в редьюсере.
2. Редьюсер усредняет пришедшие рейтинги по пользователям. Результат его работы – пары (***key***: user; ***value***: mean_rating).

Сложность / Программа	Вычисления	Память
Маппер	O(alpha * U * I / M)	O(1)
Редьюсер	O(alpha * U * I / R)	O(1)

2 Стадия

Группировка выражений $r_{u,i} - \overline{r_u}$ - компонент для последующего вычисления similarity.

1. Маппер сначала принимает на вход вспомогательный файл ratings.csv и заполняет списки, сопоставляющие каждому пользователю усреднённый рейтинг просмотренных им фильмов (получено на предыдущей стадии). Затем в основном потоке ввода маппер считывает файл ratings.csv и передаёт (во время считывания) в редьюсер тройки: (***key***: user; ***value***: movie, $r_{u,i} - \overline{r_u}$).

2. Редьюсер считывает эти тройки и для каждого пользователя распознаёт все пары индексов фильмов, которые он видел $r_{u,j} - \overline{r_u}$ (так как эти индексы пришли в редьюсер по ключу этого пользователя). Во время распознавания редьюсер для каждой такой пары фильмов (i, j) выдаёт четвёрку (***key***: i, j; ***value***: $r_{u,i} - \overline{r_u}$,). Все эти действия редьюсер выполняет, когда полностью считает данные для соответствующего пользователя. На следующей стадии будет достаточно произвести агрегацию этих значений для всех пар (i, j).

Сложность / Программа	Вычисления	Память
Маппер	O(alpha * U * I / M)	O(U)
Редьюсер	O(U * [alpha * I] ^ 2) / R)	O(alpha * I)

3 Стадия

Вычисление $sim(i,j)$

1. От прошлой стадии пришёл файл со всеми компонентами для вычисления $sim(i,j)$ Соответственно маппер должен просто передать всё, что поступило ему на вход.

2. Редьюсер должен для каждой пары фильмов (i, j) произвести агрегацию компонент её similarity - это он может делать через накопление во время считывания потока ввода, поэтому для каждого редьюсера на вычисления не требуется больше памяти, чем O(1).

Сложность / Программа	Вычисления	Память
Маппер	O(I ^ 2 / M)	O(1)
Редьюсер	O(alpha * U * I / R)	O(1)

4 Стадия

Вычисление рейтингов новых фильмов.
Самая тяжеловесная стадия.

1. Маппер предварительно принимает вспомогательный файл ratings.csv для того, чтобы создать для работы таблицы. Эти таблицы сопоставляют пользователям индексы фильмов, которые они видели. Далее маппер принимает на вход 2 файла: ratings.csv и файл со значениями similarity, посчитанные на прошлой стадии.
   Маппер должен передать в редьюсер тройки (key: user, movie; value: rating) или четвёрки (***key***: user, movie; ***value***: seen_movie, similarity[movie, seen_movie]) в зависимости от того, с какого файла считана очередная строчка. При этом эти строчки должны быть поданы для всех пар (user, movie), в которых пользователь user не видел фильм movie. Для программы работает следующий алгоритм (маппер может отличать строчки двух входных файлов по их ключам):
   Если пришёл элемент из similarity, то его необходимо подать всем парам (user, movie), для которых этот элемент будет входить в итоговый рейтинг. Этот элемент similarity подойдёт всем пользователям, которые видели ровно один фильм из двух – это определяется как раз по таблицам из вспомогательного файла.
   Если пришёл элемент из ratings.csv, то его необходимо подать на все пары (user, movie), где movie – фильм, который пользователь user не видел. Опять же, это множество пар определяется из таблиц вспомогательного файла.

2. Редьюсер принимает значения для всех пар (user, movie) и должен вычислить их рейтинги. Он это делает, считав полностью все данные для одной пары (user, movie). Редьюсер должен выдать данные в формате (***key***: user, movie; ***value***: rating).

Сложность / Программа	Вычисления	Память
Маппер	O(U * [I ^ 2] / M + U * [([1 – alpha] * I) ^ 2] / M)	O(alpha * U * I)
Редьюсер	O([1 – alpha] * U * I / R)	O([1 – alpha] * U * I)

5 Стадия

Ранжирование лучших фильмов по их идентификаторам.
Основная часть вычислений была сделана на предыдущих стадиях. Остаётся отсортировать фильмы по рейтингам и отобрать лучшие 100.

1. Маппер принимает тройки значений с предыдущего этапа и преобразует их для эффективного использования сортировки. В редьюсер он передаёт (во время считывания входного потока) следующие тройки: (***key***: ‘<user>_<5.0 - rating>’; ***value***: movie). Это позволит подать на вход редьюсеру уже отсортированный список фильмов для каждого пользователя (так как обратный рейтинг будет содержаться в ключе, и фильмы с самыми высокими рейтингами окажутся в начале отсортированного списка, при этом сохранится порядок соответствия к пользователям).
2. Редьюсер примет эти тройки и для каждого пользователя выведет первые 100 значений. Они уже отсортированы, поэтому редьюсер не требует память больше O(1).

Сложность / Программа	Вычисления	Память
Маппер	O([1 – alpha] * U * I / M)	O(1)
Редьюсер	O([1 – alpha] * U * I / R)	O(1)

6 Стадия

Перевод идентификаторов фильмов в названия.

1. Требуется произвести агрегацию информации файла movies.csv и файла с лучшими фильмами для каждого пользователя. Для этого в качестве вспомогательного файла используется movies.csv. Эта операция замены индекса фильма на его название производится в маппере. Кроме того, маппер должен вывести данные в следующем виде: (***key***: '<user>_<rating>_<movie_name>'; ***value***: anything). Это позволит отсортировать рекомендуемые фильмы как по рейтингу, так и по названию (если рейтинги совпадают).

2. Редьюсер, приняв, эти значения, выводит их в финальный файл в требуемом виде.

Сложность / Программа	Вычисления	Память
Маппер	O(U * 100 / M)	O(I)
Редьюсер	O(U * 100 / R)	O(1)

Время выполнения программы

Программа выполняется за 1 час 45 минут. Основное время затрачивается на маппер 4 стадии (полчаса на обработку similarity + час на обработку ratings.csv – из-за необходимости «размножить» данные для разных пар <пользователь-фильм>).

Запуск программы

Для запуска докер-инфраструктуры необходимо ввести следующие команды в терминале из основной директории:

- docker build -t maksim64/hadoop-base:3.2.1 ./hadoop/base
- docker build -t maksim64/spark-base:3.1.1 ./spark/base
- docker-compose build --no-cache --parallel
- docker-compose up

После этого для запуска задачи необходимо ввести следующие команды (в другом терминале):

- docker cp MapReduce namenode:/
- docker exec -it namenode bash
- cd MapReduce
- sh run.sh

Отметим, что развёртывание множества докер-файлов производится вне скрипта run.sh для удобства запуска. Для копирования результов на компьютер предлагается использовать команды:

- docker exec -it namenode bash
- mkdir /dir_for_result
- hdfs dfs -copyToLocal /MapReduce/data/output/final/* /dir_for_result/
- exit
- docker cp namenode:/dir_for_result/* /<your_desire_path>