

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
кафедра Информатики

Дисциплина: Операционные системы и среды

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе на тему

РАСПРЕДЕЛЕННОЕ РЕШЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ ЗАДАЧИ

Студент: гр. 753503 Милинкевич М.И.

Руководитель: ассистент кафедры информатики

Шнейдер В.В.

Минск 2020

СОДЕРЖАНИЕ

Введение.....	3
1 Рассматриваемая вычислительная задача. Общее описание.....	4
1.1 Соревнование Netflix Prize.....	4
1.2 Алгоритм построения рекомендаций.....	4
1.2.1 Построение прямоугольной матрицы M	5
1.2.2 Поиск левых и правых сингулярных векторов матрицы M	5
2 Схема распределенной рекомендательной системы.....	7
2.1 Распределенная система. Основные требования.....	7
2.2 Разбиение задачи на множество подзадач.....	7
2.3 Распределение подзадач по вычислительным узлам.....	7
2.4 Получение и объединение результатов вычислений.....	7
3 Решение проблемы холодного старта.....	8
4 Репликация и шардирование.....	11
4.1 Репликация.....	11
4.2 Шардирование.....	11
4.3 Конфигурирование и запуск.....	11
5 Оценка эффективности распределенной системы.....	13
6 Интерфейс приложения.....	15
Заключение	17
Список использованных источников	18

Введение

Распределенные вычисления — это способ решения трудоемких вычислительных задач с использованием нескольких компьютеров.

В таком контексте распределенные вычисления являются частным случаем параллельных вычислений, то есть одновременного решения различных частей одной вычислительной задачи с помощью нескольких вычислительных устройств.

В общем случае распределенная обработка данных не подразумевает параллелизма, но возможность “параллельного использования” распределенных вычислительных систем существует.

1 РАССМАТРИВАЕМАЯ ВЫЧИСЛИТЕЛЬНАЯ ЗАДАЧА.

ОБЩЕЕ ОПИСАНИЕ

1.1 Соревнование Netflix Prize

В рамках данной курсовой работы была разработана рекомендательная система фильмов.

В октябре 2006 года началось открытое соревнование Netflix Prize на лучший алгоритм предсказания оценки, которую зритель поставит фильму, основе предыдущих оценок этого зрителя и других. Данное соревнование легло в основу вычислительной задачи для данной курсовой работы.

Участникам соревнования были предоставлены следующие данные:

- Обучающие данные (training data set) содержат 100.480.507 оценок, которые 480.189 клиентов поставили 17.770 фильмам (на данный момент уже более 42000 фильмов). Каждая оценка представляет собой квадруплет <номер клиента, номер фильма, дата оценки, оценка>. Номера клиентов и фильмов — целые числа, оценка — целое число от 1 до 5 (низшая оценка 1, высшая 5). Таким образом, в среднем каждый клиент поставил около 200 оценок, а каждый фильм получил около 5000. Однако количество оценок сильно варьируется: так, некоторые фильмы получили всего 3 оценки, а один клиент оценил более 17 тыс. фильмов.
- Квалификационные данные (qualifying data set) содержат 2,8 млн триплетов <пользователь, фильм, дата оценки>. Оценки известны только жюри и были опубликованы по окончании соревнования.
- Названия и годы выхода в прокат всех 17.770 фильмов.
- Фрагмент обучающей базы данных, распределённый так же, как квалификационные данные. Этот фрагмент может использоваться, например, для оценки алгоритмов до посылки их на сайт.

Среди описанных выше данных для разработки рекомендательной системы были использованы обучающие данные (файл ratings_small.csv в исходном коде) и названия и годы выхода в прокат фильмов (файл movies_metadata.csv в исходном коде).

1.2 Алгоритм построения рекомендаций

Одним из алгоритмов построения рекомендаций, который придумали участники соревнования Netflix Prize, является использование коллаборативной фильтрации с помощью svd-разложения (далее сингулярное разложение).

Коллаборативная фильтрация — это один из методов построения прогнозов (рекомендаций) в рекомендательных системах, использующий известные предпочтения пользователей для предсказания неизвестных предпочтений другого пользователя. Его основное допущение состоит в

следующем: те, кто одинаково оценивал какие-либо предметы в прошлом, склонны давать похожие оценки другим предметам и в будущем.

Коллаборативная фильтрация – это самый популярный ныне вид рекомендательных систем.

Сингулярное разложение матрицы M позволяет вычислять сингулярные числа данной матрицы, а также левые и правые сингулярные векторы матрицы M .

1.2.1 Построение прямоугольной матрицы M

Матрица M строится с помощью обучающих данных и заполняется следующим образом:

- В ячейку M_{ij} записывается оценка, которую пользователь с номером i поставил фильму с номером j .
- Пустые ячейки матрицы заполняются нулями.
- От всех значений в матрице, кроме нулевых, отнимается средняя оценка. Это делается для того, чтобы рекомендательная система могла предсказывать пользователю не только фильмы, которые он уже смотрел ранее.

Так как очевидно, что матрица M является разрежённой, то для ее программного хранения используется `lil_matrix` из пакета `scipy.sparse`.

1.2.2 Поиск левых и правых сингулярных векторов матрицы M

Сингулярным разложением матрицы M порядка $m \times n$ является разложение следующего вида:

$$M = U\Sigma V^*,$$

где Σ — матрица размера $k \times k$ с неотрицательными элементами, у которой элементы, лежащие на главной диагонали — это сингулярные числа (а все элементы, не лежащие на главной диагонали, являются нулевыми), а матрицы U (порядка $m \times k$) и V (порядка $k \times n$) — это две матрицы, состоящие из левых и правых сингулярных векторов соответственно (а V^* — это сопряженно-транспонированная матрица к V).

Матрицы U, V выбираются с минимальным квадратичным отклонением (RMSE).

Что касается программной реализации, численные алгоритмы нахождения сингулярного разложения встроены во многие математические пакеты. Так, например, в данной курсовой работе была использована функция `svds` из пакета `scipy.sparse.linalg` языка Python, которая находит необходимые левые и правые сингулярные вектора.

После получения матрицы C и V , становится возможным предсказать оценку, которую пользователь поставил бы фильму. Таким образом, оценка, которую i -ый пользователь поставит j -ому фильму вычисляется по формуле:

$$M_{ij} = U_i \times V_j,$$

где U_i — это вектор i -ого пользователя, V_j — это вектор j -ого фильма.

2 СХЕМА РАСПРЕДЕЛЕННОЙ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

2.1 Распределенная система. Основные требования

Как известно, распределенная система должна уметь разбивать одну большую задачу на множество маленьких подзадач, распределять эти подзадачи по вычислительным узлам, принимать результаты вычислений и объединять их в единое целое. Рассмотрим каждый из описанных пунктов в рамках данной курсовой работы.

2.2 Разбиение задачи на множество подзадач

Как уже говорилось ранее, имеются данные по более чем 42000 фильмов мы имеем для каждого фильма вектор размерности 100. Для каждого пользователя также имеется вектор размерности 100. Таким образом, для того, чтобы предсказать оценку по каждому фильму для конкретного пользователя, вычислительному устройству необходимо совершить не менее $42000 \cdot 100 = 4200000$ операций.

Если же исходные данные разбить на n блоков и раздать эти блоки разным вычислительным устройствам, то вычислительная сложность уменьшится в n раз.

2.3 Распределение подзадач по вычислительным узлам

В данной курсовой работе данные были разбиты случайным образом на 10 частей. Каждую часть данных обрабатывает отдельный вычислительный узел (worker).

Worker берет свою часть данных о фильмах и вектор пользователя, далее вычисляет скалярное произведение вектора пользователя со всеми векторами фильмов, которые для него доступны. Все полученные данные сохраняются и из них выбирается 10 с максимальным значением. Затем эти значения отправляются на основной сервер.

2.4 Получение и объединение результатов вычислений

Основной сервер собирает результаты со всех вычислительных узлов и выбирает из них 10 с максимальным значением. Полученная выборка и будет предоставлена пользователю в качестве рекомендаций.

3 РЕШЕНИЕ ПРОБЛЕМЫ ХОЛОДНОГО СТАРТА

Проблема холодного старта — это задача выдачи релевантных рекомендаций для новых пользователей. Так как система не знает, какие фильмы предпочитает пользователь, его вектор U равен нулю и все документы для него имеют одинаковый вес рекомендации.

Когда новый пользователь заходит на сайт рекомендательной системы, ему необходимо зарегистрироваться, для чего он вводит имя пользователя и пароль.

Рассмотрим следующий способ решения проблемы холодного старта, используемый на платформе Netflix и реализованный в данной курсовой работе:

- После регистрации пользователю предоставляется выборка из 50 лучших фильмов по мнению предыдущих пользователей. А именно, была выбрана эвристика:

$$f(m) = \hat{vote} \cdot popularity + \log(revenue)$$

Где для фильма m : его средняя оценка равна \hat{vote} , популярность (процент просмотревших его пользователей) $popularity$ и прибыль в прокате $revenue$.

- Таким образом получается выборка популярных фильмов с высокими оценками, которые получили высокую прибыль в прокате. Высока вероятность того, что новый пользователь знаком с этими фильмами и сможет их оценить.

На рисунках 3.1-3.2 показан интерфейс, который доступен пользователю для выбора фильмов. Изначально все постеры фильмов черно-белые, а после того, как пользователь выбирает некоторый фильм, его постер становится цветным, затем необходимо нажать на кнопку *Submit*, чтобы отправить данные на обработку. На рисунке 3.3 показано то, как для пользователя выглядит процесс обработки данных.

После получения данных о предпочтениях пользователя среди предложенных фильмов, рекомендательная система дополняет уже имеющиеся данные о пользователях новыми. То есть, к описанной ранее матрице M приписывается новая строка с оценками пользователя. В качестве оценок выставляется 5.0. Затем пересчитываются левый и правый сингулярные вектора, после чего мы можем запустить распределенную систему для поиска 10-ки лучших фильмов.

Так как множество пользователей увеличилось на 1, а множество фильмов не изменилось, достаточно обновить на основном сервере ресурс с векторами пользователей, что займет несколько секунд.

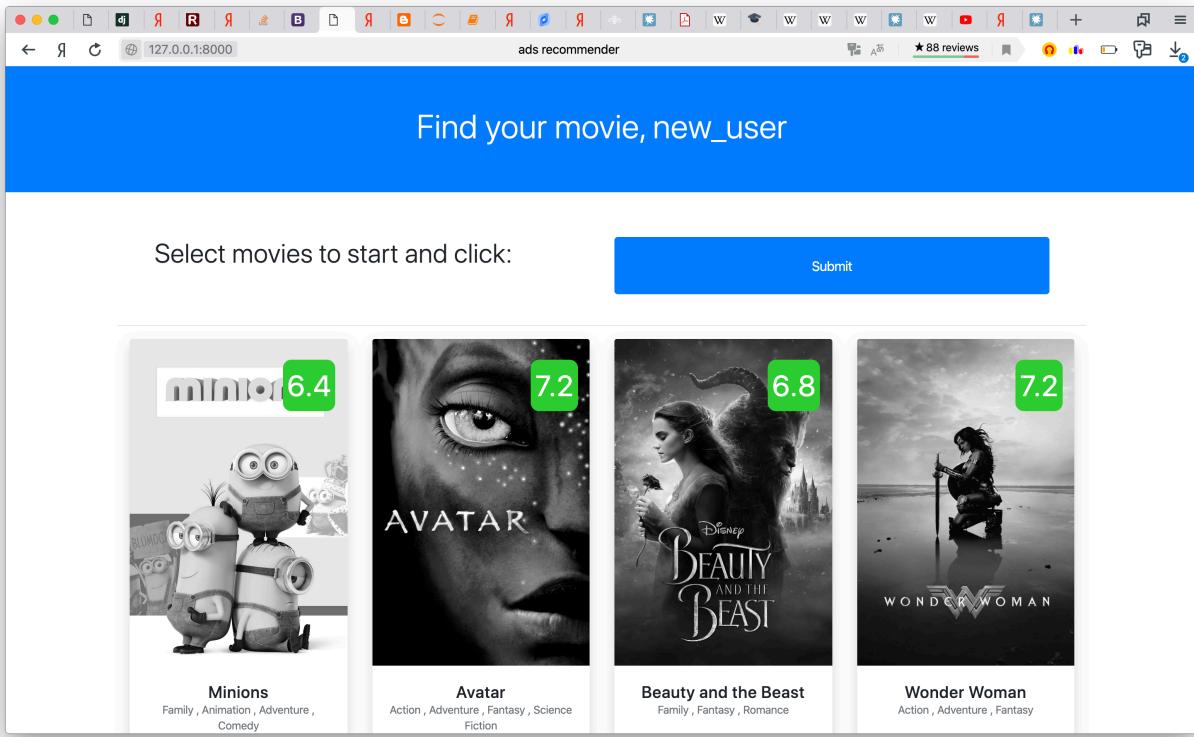


Рисунок 3.1 Интерфейс главной страницы рекомендательного приложения

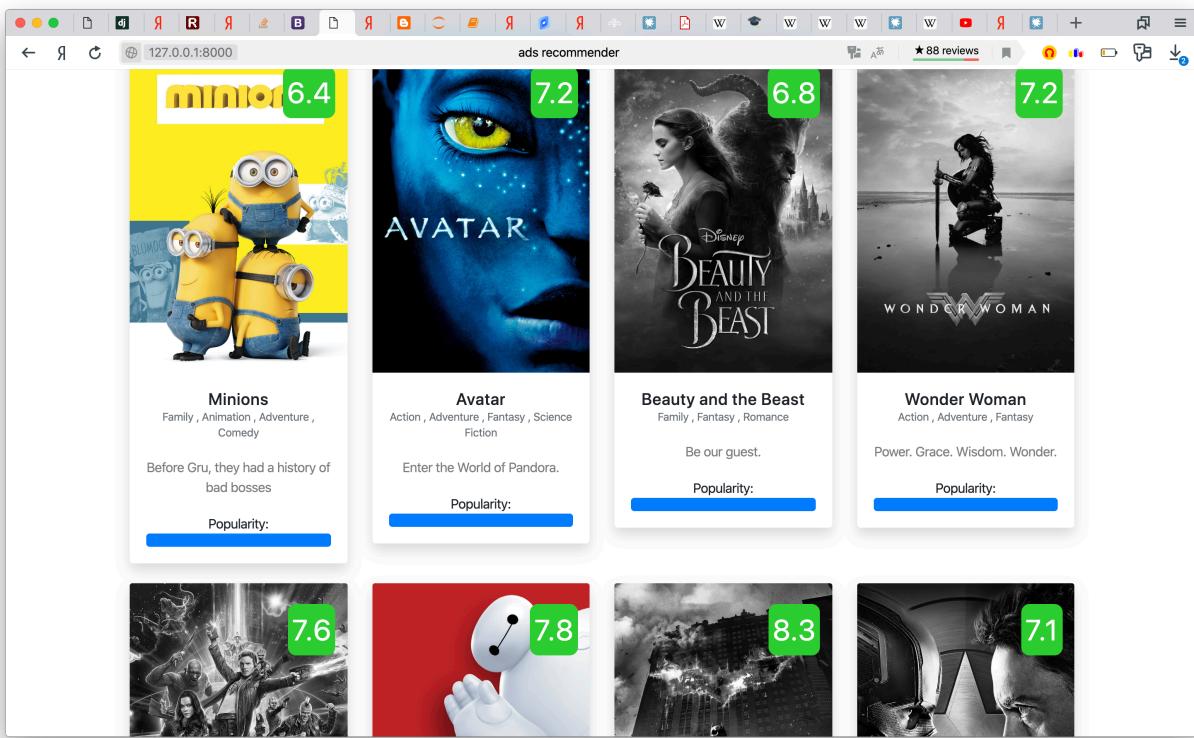


Рисунок 3.2 Интерфейс главной страницы рекомендательного приложения после выбора некоторых фильмов

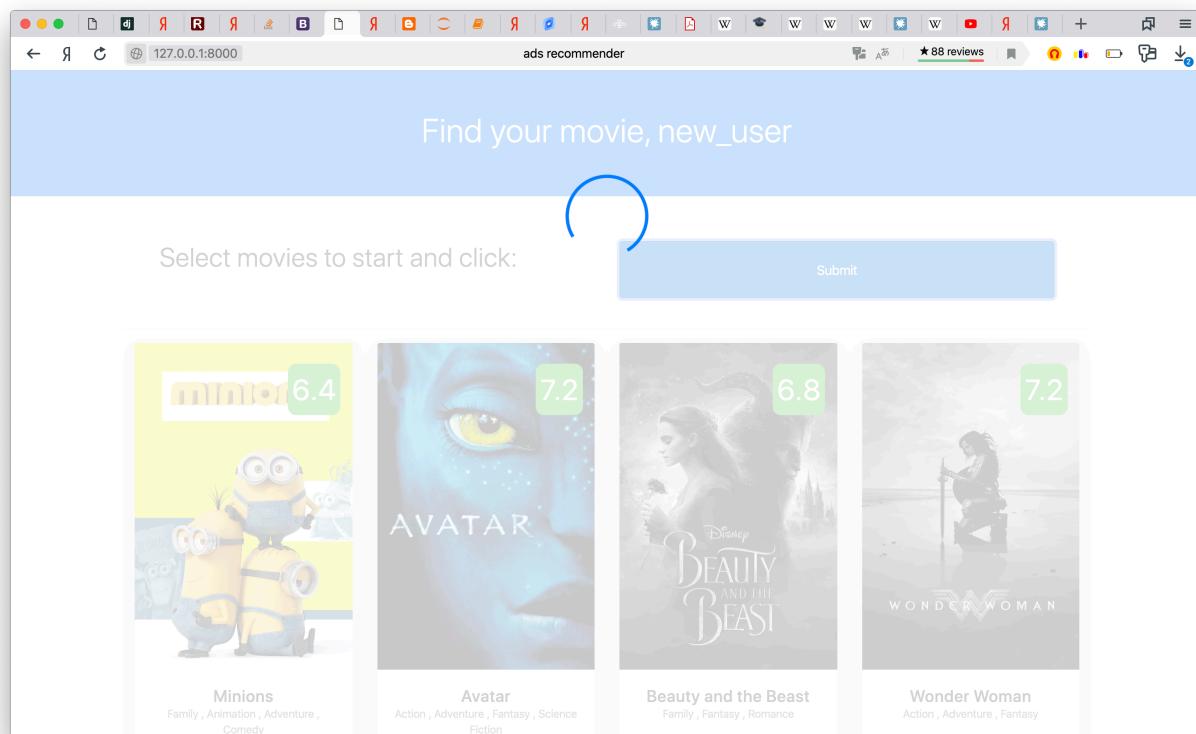


Рисунок 3.3 Ожидание завершения обработки данных

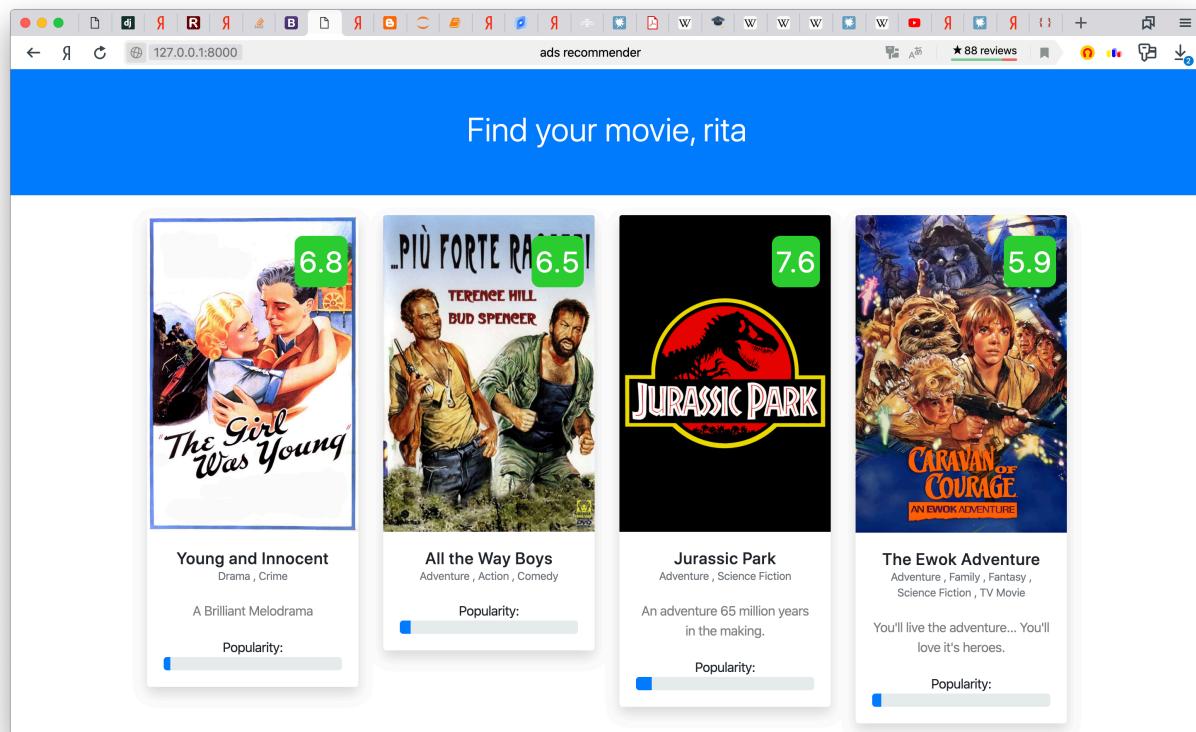


Рисунок 3.4 Подборка фильмов по полученным данным

4 РЕПЛИКАЦИЯ И ШАРДИРОВАНИЕ

4.1 Репликация

Для создания устойчивой к различным неполадкам и условиям системы применим метод репликации.

Репликация (англ. replication) — механизм синхронизации содержимого нескольких копий объекта (например, содержимого базы данных). Репликация — это процесс, под которым понимается копирование данных из одного источника на другой (или на множество других) и наоборот.

К worker-m, описанным во 2-м пункте добавим 10 клонов, которые запускаются на отдельных серверах, над тем же множеством документов. При недоступности основного адреса, отправим запрос на реплику. Таким образом, без потери эффективности мы увеличили устойчивость.

4.2 Шардирование

Разбив всю базу фильмов на 10 равных случайных частей, мы применили метод шардирования.

Шардирование — стратегия горизонтального масштабирования данных, при которой части коллекций базы данных размещаются на разных хостах кластера.

Шардирование предлагает распределить нагрузку по хостам базы данных — это позволяет преодолеть ограничения ресурсов одного сервера, что особенно актуально при больших объемах данных или необходимости в интенсивных вычислениях.

4.3 Конфигурирование и запуск

Для удобства и возможности переиспользования спроектированной системы были разработаны следующие файлы конфигурации, представленные на рисунках 4.1 — 4.3.

```
{  
  "workers": [  
    {  
      "address": "http://127.0.0.1:8001",  
      "shard": 1  
    },  
    {  
      "address": "http://127.0.0.1:8002",  
      "shard": 1  
    },  
    {  
      "address": "http://127.0.0.1:8003",  
      "shard": 2  
    },  
    {  
      "address": "http://127.0.0.1:8004",  
      "shard": 2  
    },  
    {  
      "address": "http://127.0.0.1:8005",  
      "shard": 3  
    },  
    {  
      "address": "http://127.0.0.1:8006",  
      "shard": 3  
    },  
    {  
      "address": "http://127.0.0.1:8007",  
      "shard": 4  
    },  
    {  
      "address": "http://127.0.0.1:8008",  
      "shard": 4  
    },  
    {  
      "address": "http://127.0.0.1:8009",  
      "shard": 5  
    },  
    {  
      "address": "http://127.0.0.1:8010",  
      "shard": 5  
    }  
  ]  
}
```

Рисунок 4.1 Конфигурация основного сервера

```
{  
    "movie_vec": "ads_worker/movie_vectors/2.csv",  
    "movies": [{"id": 22102}, {"id": 6951}, {"id": 11959}, {"id": 2616}, {"id":  
}
```

Рисунок 4.2 Конфигурация 2-го worker-a

```
if IS_META:  
    META_CONFIG_PATH = os.path.join(BASE_DIR, 'ads_recommender/config.json')  
    META_CONFIG = json.load(open(META_CONFIG_PATH))  
    Meta()  
else:  
    shard = int(input('What shard to run?\n'))  
    CFG_PATH = os.path.join(BASE_DIR, 'ads_worker/config/{}.json'.format(shard))  
    CONFIG = json.load(open(CFG_PATH))  
  
Worker()
```

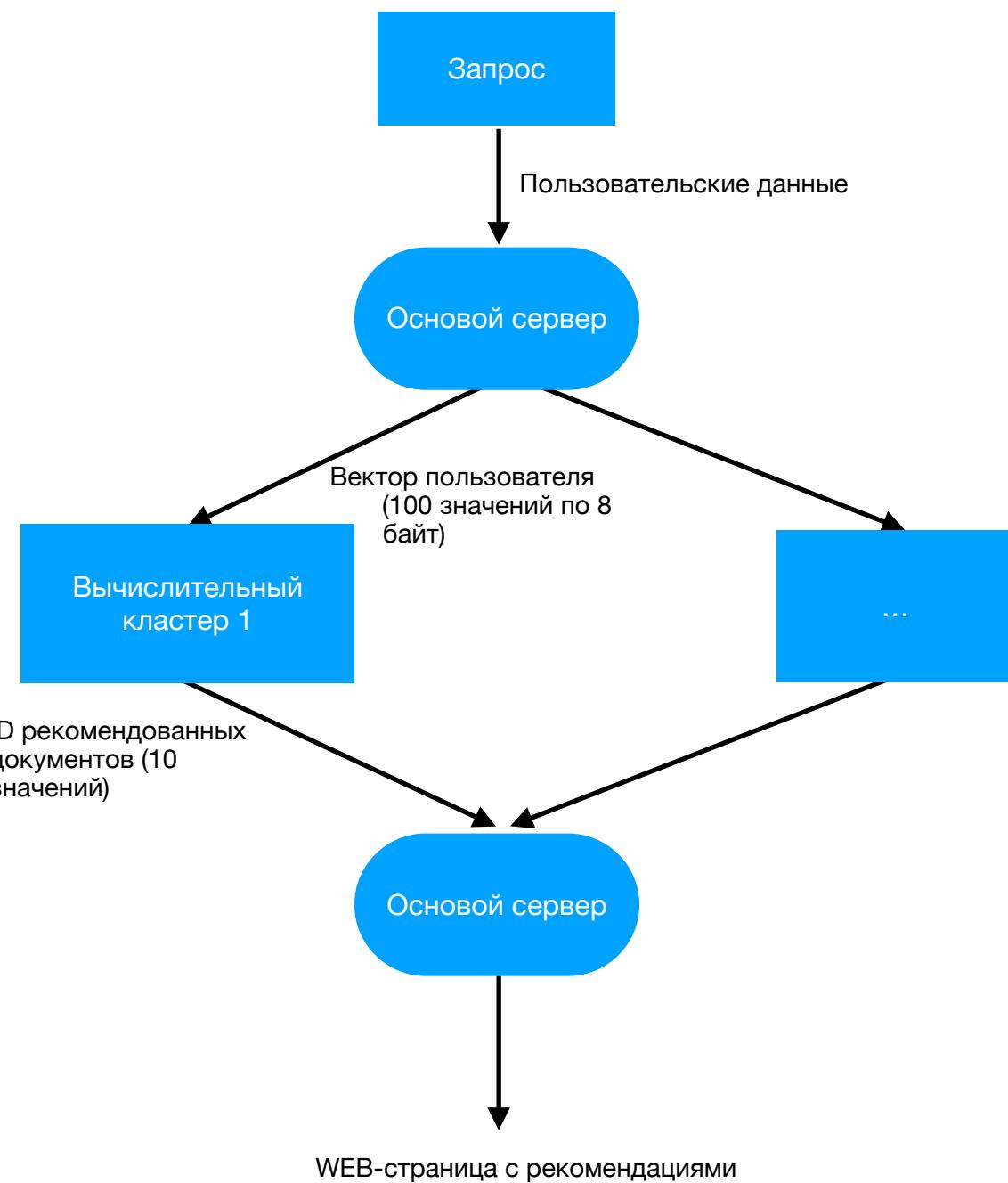
Рисунок 4.3 Программный выбор одной из конфигураций

Таким образом, при запуске программы, необходимо иметь только ресурс с векторами пользователей (в случае запуска основного сервера) или соответствующий шарду worker-a ресурс с векторами фильмов. Готовую систему можно переиспользовать для рекомендации чего угодно, например, рекламных баннеров, заведений, музыки, социальных связей.

5 ОЦЕНКА ЭФФЕКТИВНОСТИ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

Для оценки системы будем учитывать два параметра на запрос: количество затраченного времени и сети.

Жизненный цикл запроса, на котором выделены данные, передающиеся по сети:



Самым узким местом в передаче данных по сети является передача пользовательского вектора (~8кб на запрос).

Количество операций на процессоре в одном вычислительном кластере составляет $4500 * 100 = 4.5$ млн операций. Так как в 10 кластерах вычисления скалярных произведений выполняются параллельно, это и будет итоговой оценкой по времени.

При увеличении количества шардов будет увеличиваться сетевой трафик, но снижаться количество процессорных операций.

6 ИНТЕРФЕЙС ПРИЛОЖЕНИЯ

Ранее уже были приведены примеры скриншотов главной страницы рекомендательной системы, а также было описано, что в приложении имеется функционал, который позволяет новым пользователям регистрироваться в системе, а ранее зарегистрированным заходить под своим именем пользователя.

На рисунках 6.1-6.2 изображен интерфейс страницы регистрации и аутентификации.

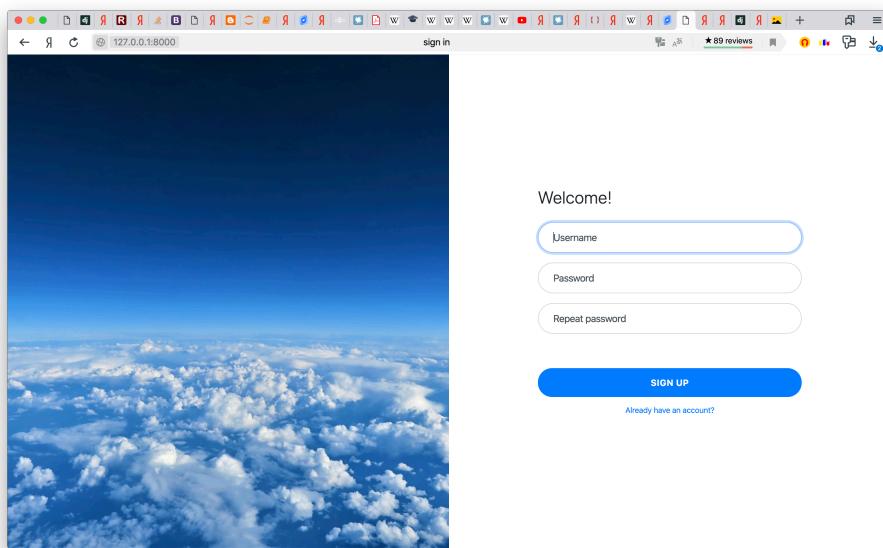


Рисунок 6.1 Интерфейс страницы регистрации

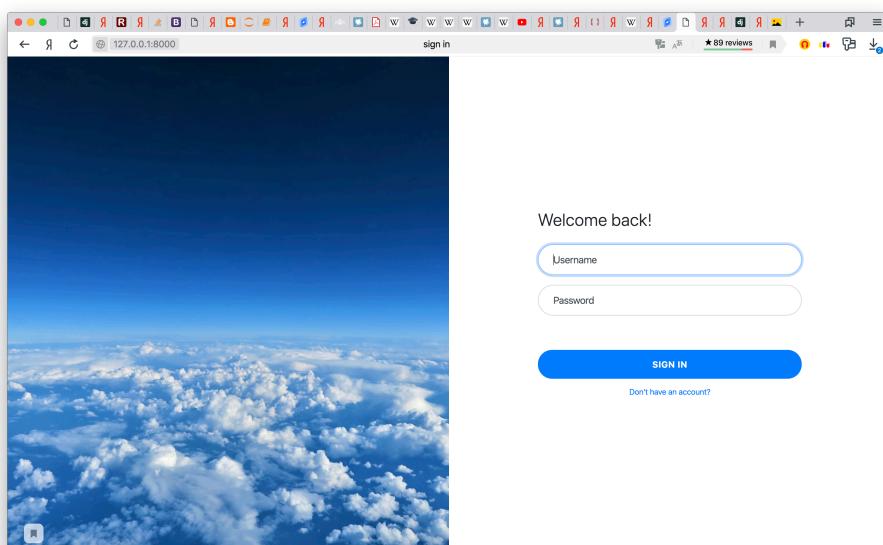


Рисунок 6.2 Интерфейс страницы авторизации

В системе можно авторизоваться за любого из 671 пользователя, которые были предоставлены для обучения. После авторизации пользователю предлагается подборка из 10 фильмов, которая обновляется после того, как новый пользователь зарегистрируется в системе. Так, например, на рисунке 6.3 изображена подборка фильмов для пользователя с $id = 501$.

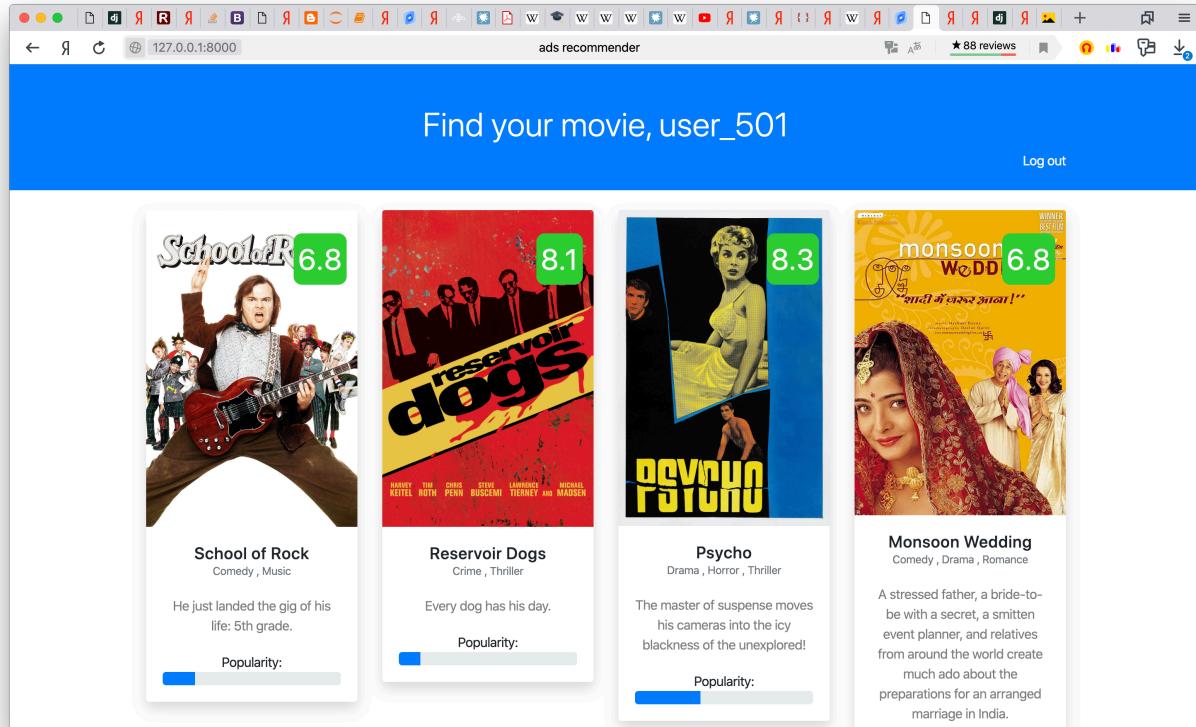


Рисунок 6.3 Рекомендации для пользователя user_501

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была реализована рекомендательная система, которая работает на основе распределенных вычислений. Имеется 10 компьютеров, каждый из которых обрабатывает свою порцию данных и выдает свой ответ для основного сервера. Использование распределенных вычислений значительно ускоряет процесс обработки данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Habr [Электронный ресурс]. – Режим доступа : <https://habr.com/ru/company/surfingbird/blog/168733/>
- [2] Habr [Электронный ресурс]. – Режим доступа : <https://habr.com/ru/company/surfingbird/blog/139863/>
- [3] Habr [Электронный ресурс]. – Режим доступа : <https://habr.com/ru/post/232361/>
- [4] Яндекс облако [Электронный ресурс]. – Режим доступа : <https://cloud.yandex.ru/docs/managed-mongodb/concepts/sharding>
- [5] Wikipedia [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki/Netflix_Prize
- [6] Wikipedia [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki/Коллаборативная_фильтрация
- [7] Wikipedia [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki/Сингулярное_разложение
- [8] Wikipedia [Электронный ресурс]. – Режим доступа : https://en.wikipedia.org/wiki/Root-mean-square_deviati