Проектирование баз данных, ч.2

ФИО преподавателя: Смирнов Михаил Вячеславович

e-mail: smirnovmgupi@gmail.com

Лекция 11

Средства извлечения и обработки Больших данных (BigData)

5V’s of Big Data. Volume (объем)

- Для каждой организации или компании существует предел объема данных (**Volume**) которые компания или организация способна обрабатывать одновременно для целей аналитики.
- Как правило этот объем ограничен объемами оперативной памяти серверов корпоративных приложений и баз данных и необходимостью партиционирования (Partitioning) хранимых данных.

- #терабайты #таблицы #файлы #распределение

5V’s of Big Data. Velocity (скорость, быстрота)

- Для каждой организации или компании существуют физические ограничения на количество транзакций/объем данных (**Velocity**), которая корпоративная система может обработать или передать за единицу времени вследствие ограничений (**scale in)** архитектуры этой системы.

- #пакеты #процессы #потоки #реальное/оценочное время

5V’s of Big Data. Variety (разнообразие)

- Традиционные корпоративные системы (построенные на реляционных моделях) могут использовать эффективно только структурированные источники поступления информации, не принимая во внимание разновариантные и неструктурированные источники данных (**Variety**), или имея серьезные ограничения по работе с такими источниками.

- #структурированность #неструктурированность #вероятностность #связанность #динамика

5V’s of Big Data. Veracity (точность)

- Большое количество данных и разнообразие источников требует качества и аккуратности при обработке и анализе данных (твиты, хэштэги, аббревиатуры, сокращения и т.д.). Ошибки, надежность и точность контента ставят под сомнение достоверность (**Veracity**) самих данных так и принятых решений на основе этих данных. Количество не переходит в качество.

- #достоверность #аутентичность #публичность #доступность

5V’s of Big Data. Value (ценность)

- Сбор и анализ больших данных должен предоставлять определенную ценность (**Value**) для бизнеса. Ценность данных неразрывна связанна со стоимость владения и ценностью для бизнеса.

- #корреляция #гипотезы #статистика

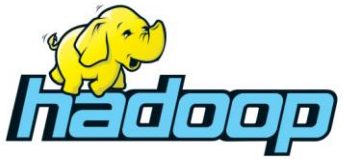
Сценарий приключения Больших Данных в социалочках

Пользователь

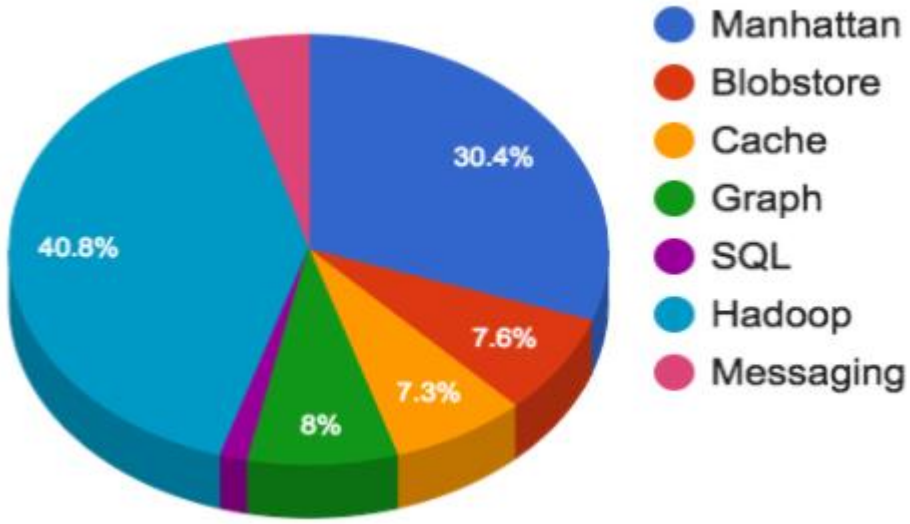
ПК
Браузер

Пользователь

Смартфон
Приложение



Пример. Структура хранения и обмена сообщениями Twitter*



1. Кластеры Hadoop для вычислений и HDFS.
2. Кластеры Manhattan для всех хранилищ key-value с малой задержкой.
3. Хранилища Graph для шардированных кластеров MySQL.
4. Кластеры Blobstore для всех крупных объектов (видео, изображения, бинарные файлы...).
5. Кластеры кэширования.
6. Кластеры обмена сообщениями.
7. Реляционные хранилища (MySQL, PostgreSQL и Vertica).

*Ссылка на полный перевод статьи: https://habr.com/ru/post/325282/

Задачи объектов инфраструктуры хранения в Twitter

- Hadoop: многочисленные кластеры хранят более 500 ПБ, разделённые по четырём группам (реальное время, обработка, хранилище данных и холодное хранилище). В самом большом кластере более 10 тыс. нод. Работает 150 тыс. приложений и запускается 130 млн контейнеров в день.

- Manhattan (бэкенд для твитов, личных сообщений, твиттер-аккаунтов и др.): кластеры read only и read/write для трафика с большой нагрузкой на чтение и запись. Кластер read only обрабатывает десятки миллионов *запросов в секунду (QPS)*, а кластер read/write — миллионы QPS.

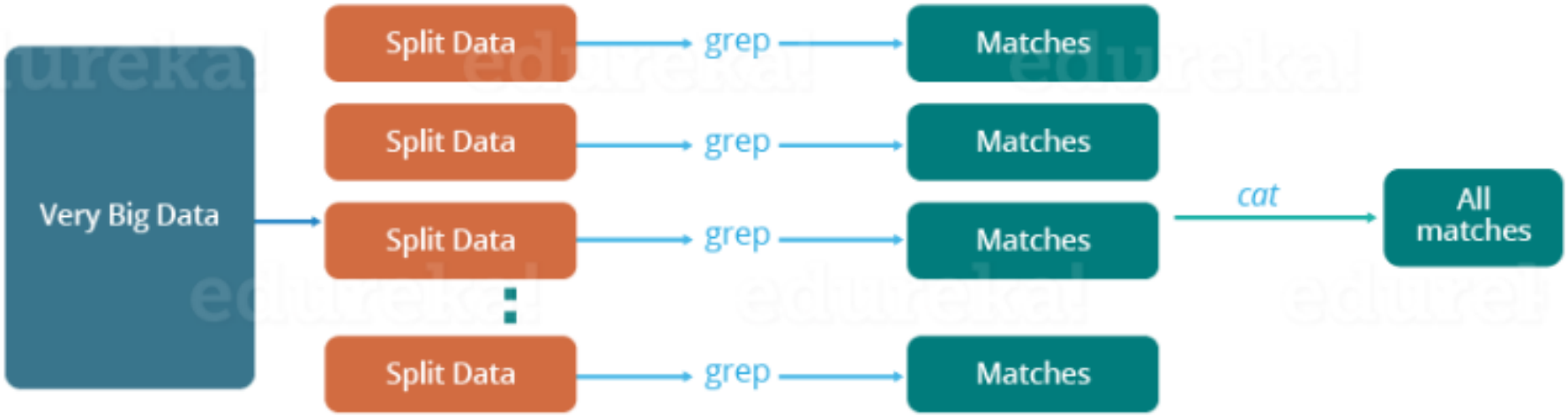
- Graph: социальный граф Flock способен справляться с пиковой нагрузкой в десятки миллионов QPS, распределяя её на серверы MySQL в среднем с 30-45 тыс. QPS.

Задачи объектов инфраструктуры хранения в Twitter

- Blobstore: хранилище для изображений, видео и больших файлов, которое содержит сотни миллиардов объектов.
- Cache: кластеры Redis и Memcache кэшируют пользователей, таймлайны, твиты и др.
- SQL: включает в себя MySQL, PostgreSQL и Vertica. MySQL/PosgreSQL используются там, где нужна строгая целостность, в управлении рекламными кампаниями, обменом рекламы, а также для внутренних инструментов. Vertica — колоночное хранилище, которое часто используется как бэкенд для продаж с поддержкой Tableau* и организации пользователей.

*Ссылка на интересную статью про Табло: https://habr.com/ru/company/luxoft/blog/569016/

Традиционный способ обработки данных*

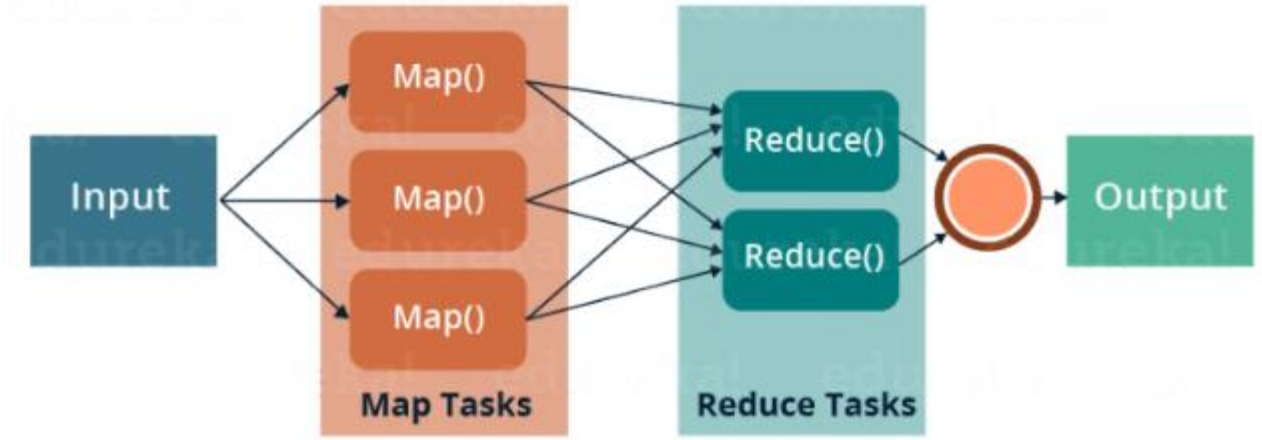


Проблемы традиционной обработки большого объема данных:

- проблема надежности кластера
- вопрос равномерного разделения данных
- проблема надежности “разделки” (split)
- проблема агрегации результата

*Ссылка на статью по теме (англ): https://thirdeyedata.io/hadoop-mapreduce/

Обработка данных через фреймворк MapReduce



Суть метода:

- две разделенные задачи – map, reduce
- map – разложение входных массивов на пары ключ-значение в промежуточный массив
- передача массива на reducer
- агрегация промежуточных пар ключ-значение в результатный массив

Классы фреймворка MapReduce

- Mapper class. Задача – разделить входящий поток данных на пары ключ-значение, заданные исследователем.

  Функции:

  - Input Split – задача в программе

  - RecordReader – разбиение массива на пары ключ-значение

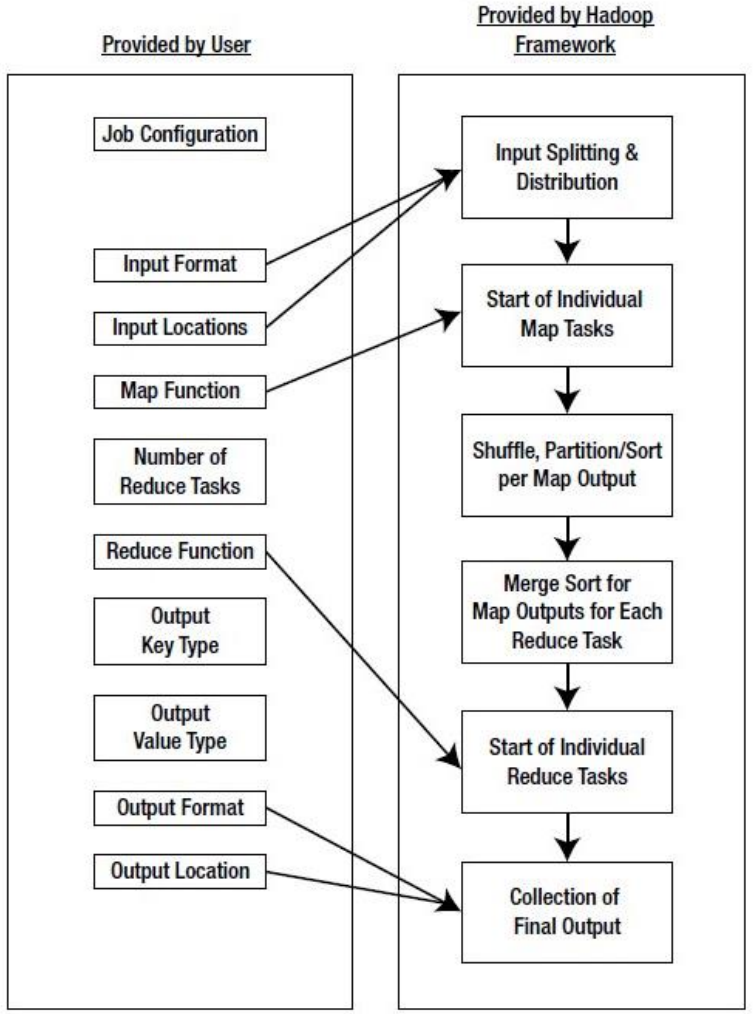
- Reducer class.

  Функция: Формирование выходного массива в формате HDFS (Hadoop Distributed File System).

- Driver class.

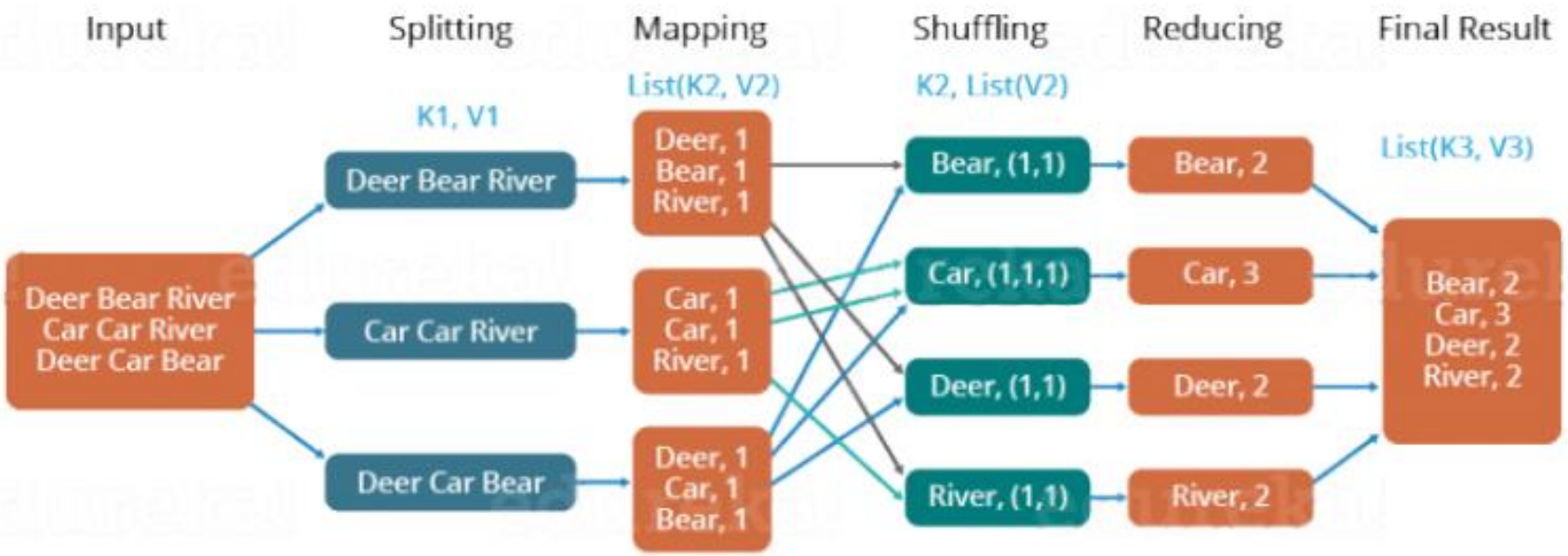
  Функция – выполнение указанных выше классов в Hadoop.

Архитектура MapReduce



Пример реализации классов фреймворка mapreduce

Задача. Разбор текстового файла с содержимым **Dear, Bear, River, Car, Car, River, Deer, Car and Bear.** Визуализация задачи:



Простейший MapReduce в Python и PHP*

```python
words = ["foo", "bar", "baz"]
def map1(word):
  return [word, 1]

arr = ["foo", [1,1]]
def reduce1(arr):
  return [ arr[0], sum(arr[1]) ]
```

```php
$words = array("foo", "bar", "baz")
function map1($word) {
  return array($word, 1);
}

arr = array("foo", array(1,1))
function reduce1(arr) {
  return array( $arr[0], array_sum($arr[1]) );
}
```

*Ссылка на статью MapReduce без зауми на русском: https://habr.com/ru/post/103467/

Спасибо за внимание!