Основы Map Reduce

Чего НЕ будет в презентации

- Технические подробности организации MR
- MR глазами разработчика
- Сравнения разных MR на предмет какую выбрать

План

1. Основные понятия **Map Reduce**

2. Примеры **MR** задач

3. Библиотеки для **MR**

1. Основные понятия

Зачем это вообще нужно?

- Горизонтальная масштабируемость

Горизонтальная масштабируемость

Зачем это вообще нужно?

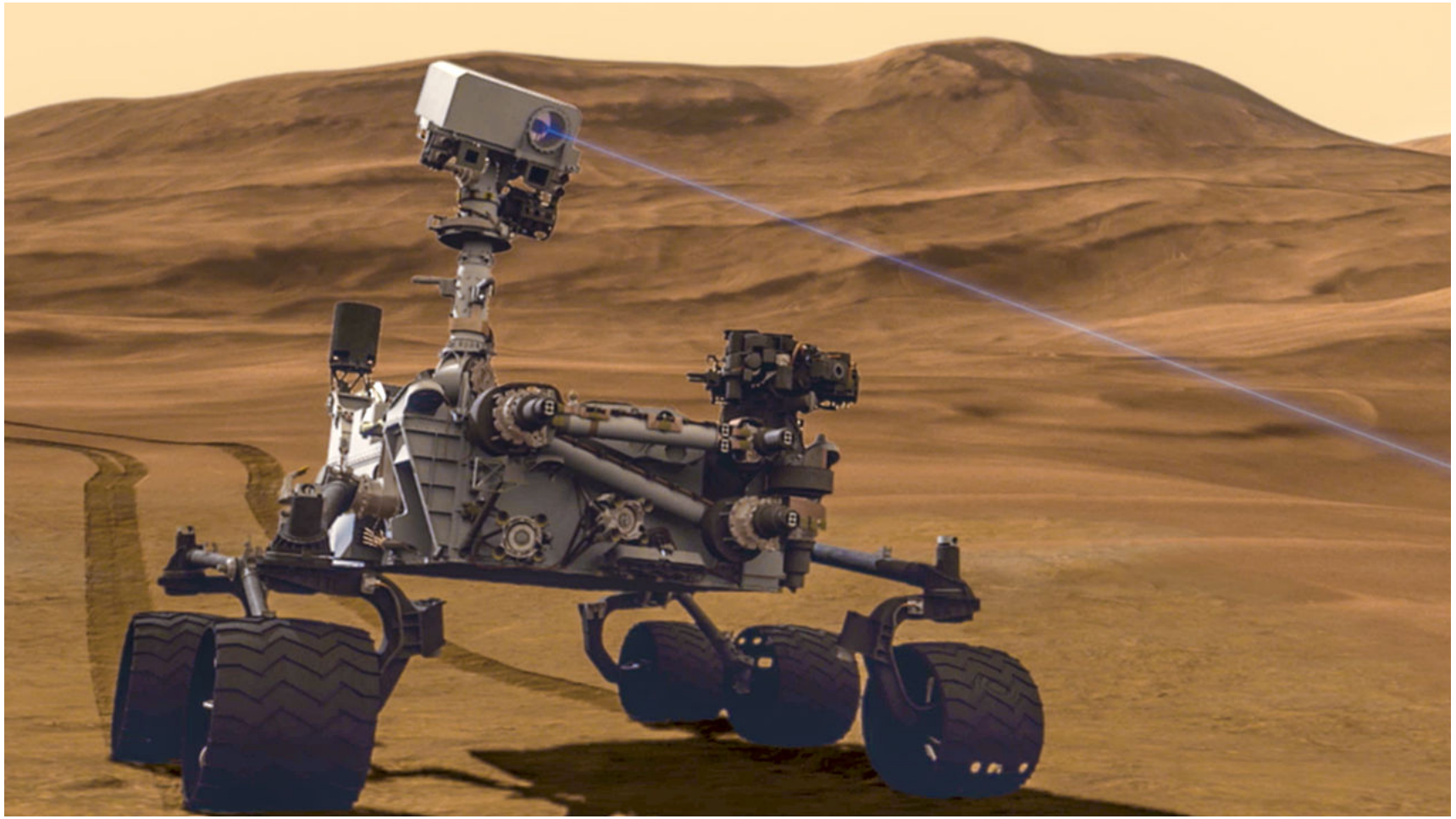
- Горизонтальная масштабируемость
- Отказоустойчивость

Отказоустойчивость

Зачем это вообще нужно?

- Горизонтальная масштабируемость
- Отказоустойчивость
- Локальность данных

Локальность данных



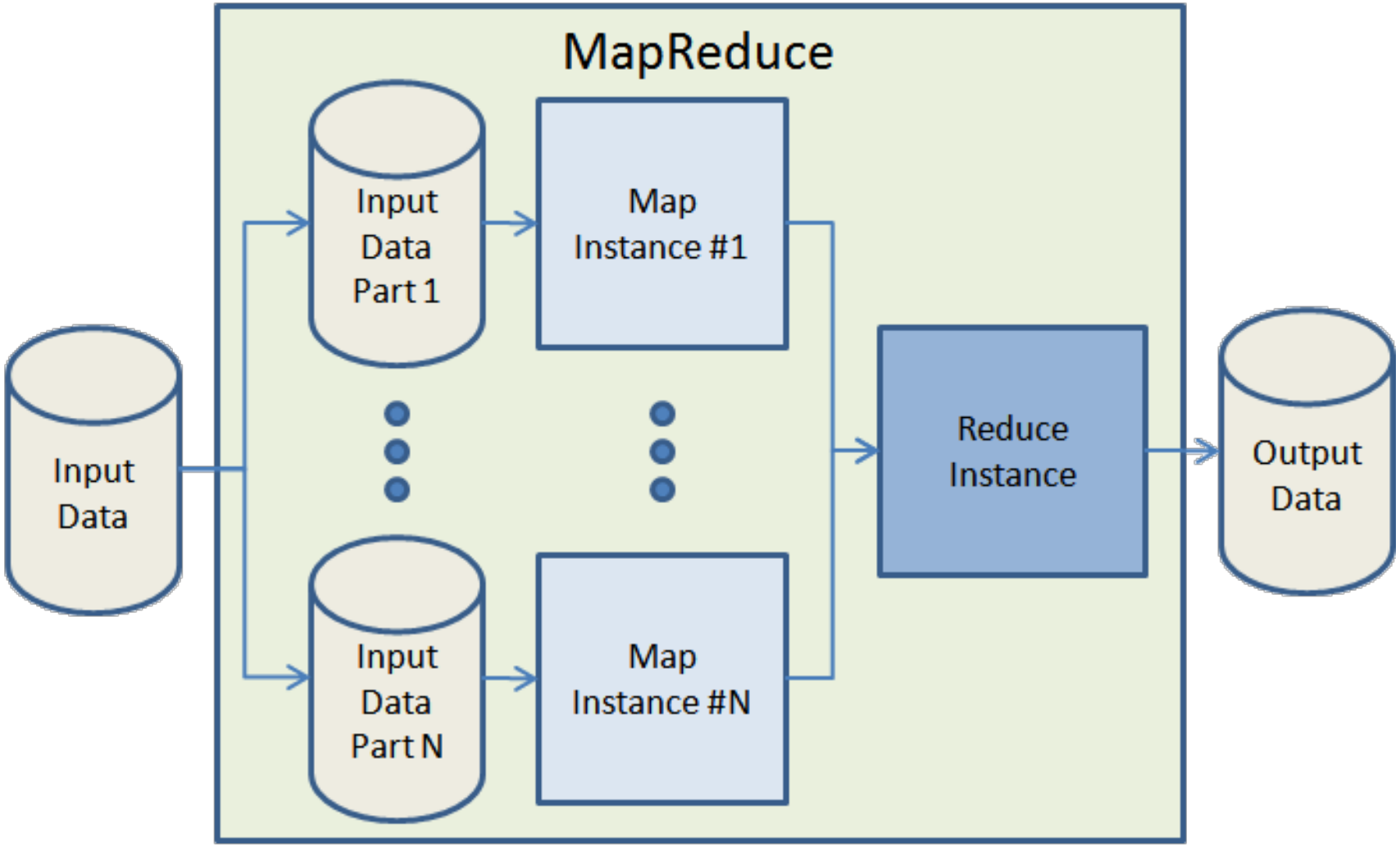
Зачем это вообще нужно?

- Горизонтальная масштабируемость
- Отказоустойчивость
- Локальность данных

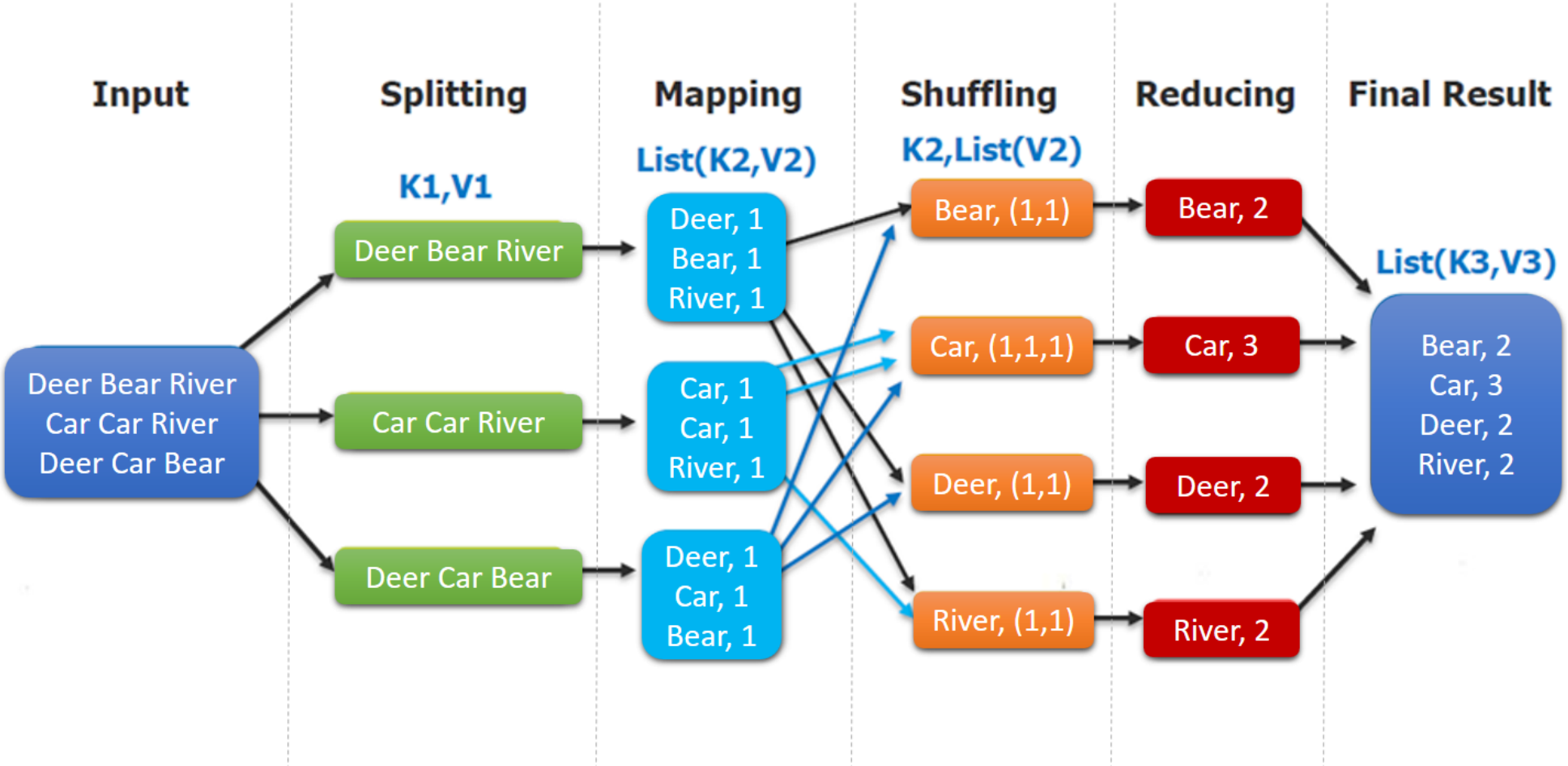
Как это идеологически работает

- Делаете map
- Делаете shuffle
- Делаете reduce
- ????
- PROFIT!!!!

Map Reduce


MapReduce
Input Data
Input Data Part 1
Input Data Part N
Map Instance #1
Map Instance #N
Reduce Instance
Output Data

Подсчет слов



MapReduce Word Count Process

Какие шаги есть на практике

- map
- sort
- join
- groupby
- reduce/aggregate

Какие шаги есть на практике

- map
- sort
- join
- groupby
- reduce/aggregate
- похоже на pandas, не находите?

Какие нюансы стоит знать

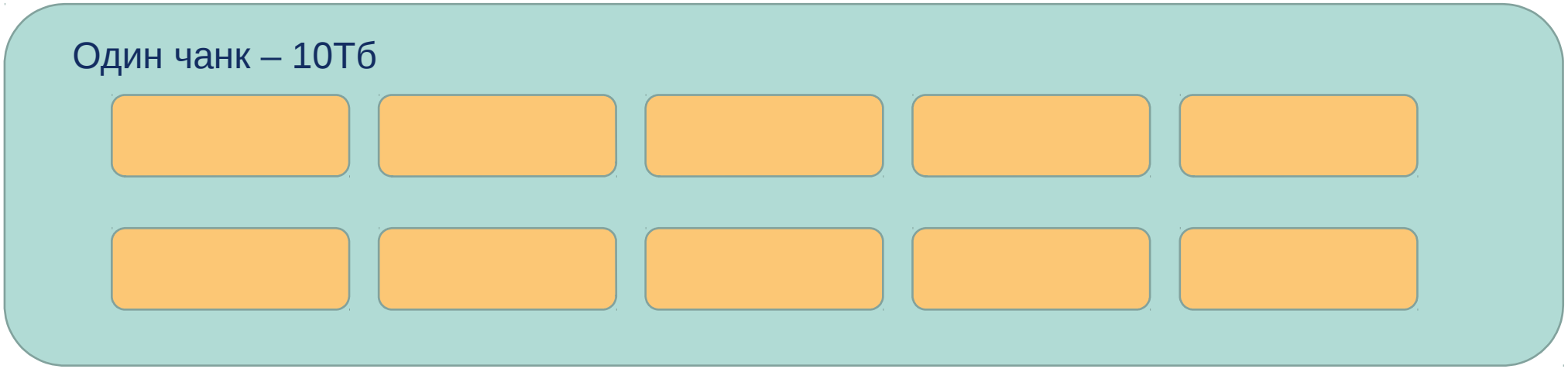
- Файлы с данными нужно хранить
- Есть три лимита: объём данных, количество объектов в системе и количество «чанков»

Какие нюансы стоит знать

- Файлы с данными нужно хранить

- Есть три лимита: объём данных, количество объектов в системе и количество «чанков»

Одна таблица данных, 100ТБ – суммарный объём, при хранении разбито на 10 чанков



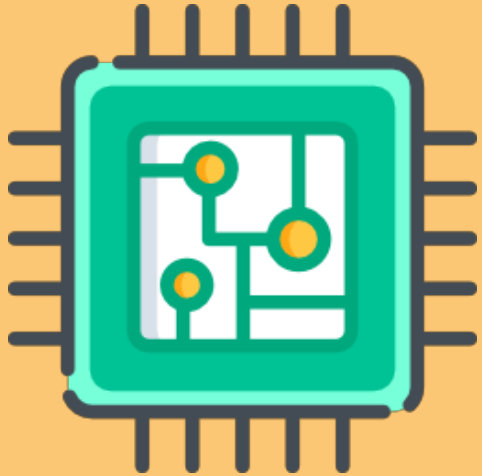
Какие нюансы стоит знать

- Файлы с данными нужно хранить
- Есть три лимита: объём данных, количество объектов в системе и количество «чанков»
- MR операции можно балансировать по CPU/disk
  - Много CPU – много чанков, долгое сохранение файлов
  - Мало CPU – большие чанки, долго ждать вычисления

Какие нюансы стоит знать

- MR операции можно и нужно оптимизировать
- Что-то сделают за вас, что-то нет
  - map + map = map
  - map + aggregate sum – можно считать суммы ещё в map-е
  - предыдущее верно для любые ассоциативных функций
  - join-ы можно делать более эффективно, если ключи уникальны
  - или одна из таблиц помещается в память

2. Примеры MR задач



Поиск максимума

- Есть таблица: дата, id магазина, сумма покупки
- Нужно найти для каждого магазина максимальную сумму покупки за каждый день

Поиск максимума

- Есть таблица: дата, id магазина, сумма покупки
- Нужно найти для каждого магазина максимальную сумму покупки за каждый день
- Что выдавать в map-е, и что делать в reduce?

Поиск максимума

- Есть таблица дата, id магазина, сумма покупки
- Нужно найти для каждого магазина максимальную сумму покупки за каждый день
- Что выдавать в map-е, и что делать в reduce?
- А как использовать тот факт, что в одном map-е будут данные скорее всего одного магазина

Построение истории

- Есть таблица: timestamp, id пользователя, id чека
- Нужно найти для каждого пользователя список его покупок, отсортированный по времени

Построение истории

- Есть таблица: timestamp, id пользователя, id чека
- Нужно найти для каждого пользователя список его покупок, отсортированный по времени
- Что выдавать в map-е, и что делать в reduce?

Построение истории

- Есть таблица: timestamp, id пользователя, id чека
- Нужно найти для каждого пользователя список его покупок, отсортированный по времени
- Что выдавать в map-е, и что делать в reduce?
- А теперь усложним – нужно построить такую историю для каждого чека (нужно знать какие чеки были до)

3. Библиотеки для MR

Библиотеки для MR

Hadoop это стек технологии, вот его основные части:

- **Hadoop Distributed File System (HDFS)**
- **Hadoop YARN**

Библиотеки для MR

Миллионы их, все и не перечесть, но с большой вероятностью это будет Hive или Apache Spark

Аналитикам обычно удобнее SQL-like запросы

А разработчикам написать код на условном python или scala