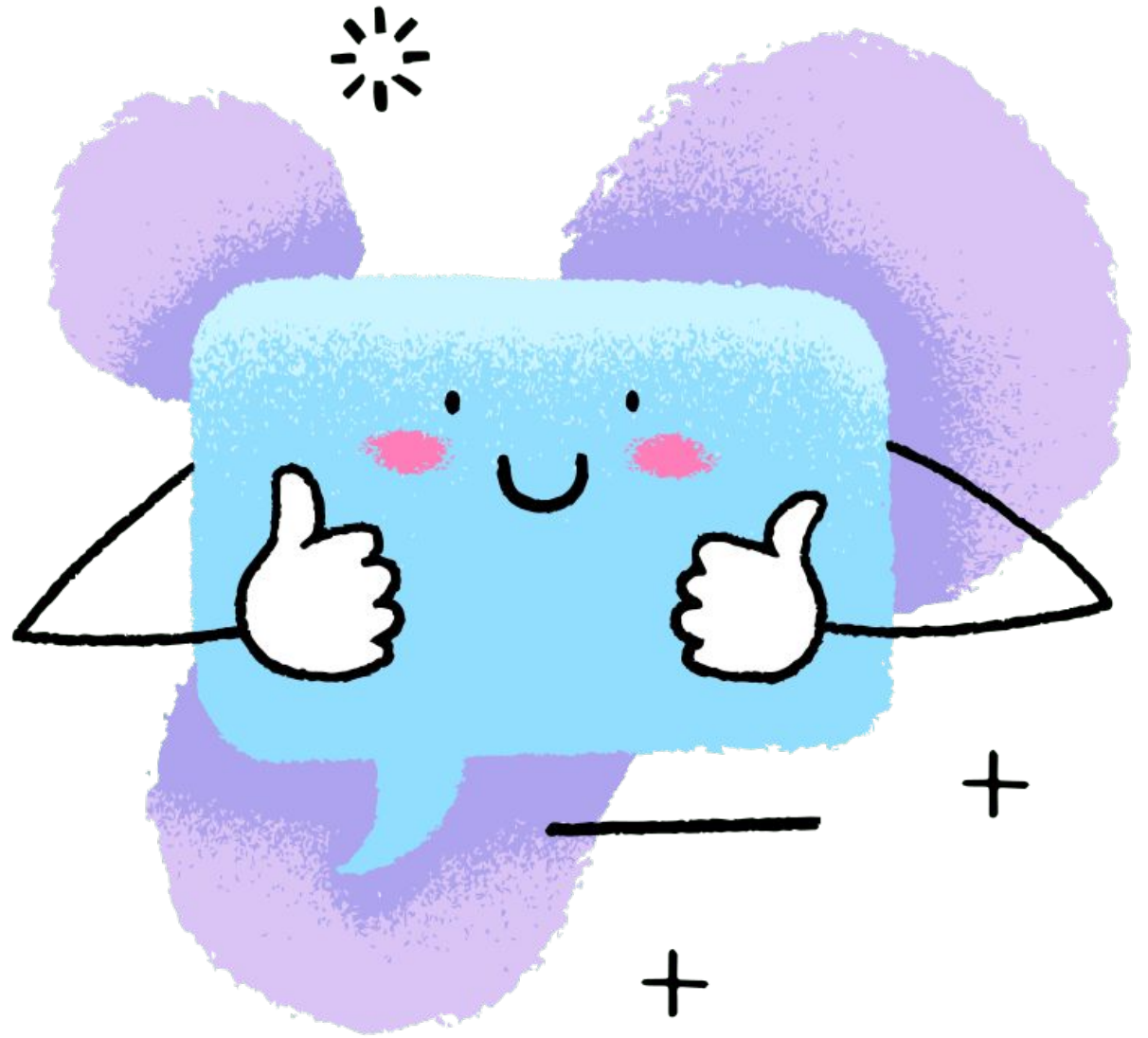


# PySpark



# Знакомство

Иванов Александр Вадимович

работаю в **Сбере** (руководитель направления по исследованию данных): создание и внедрение пайпланов обработки больших объёмов данных для моделирования (банковские транзакции, логи IPA и т.д.)

Образование: физфак МГУ, ФинТех РАНХиГС, MADE (Академия больших данных), New Professions Lab Apache Spark



# План курса

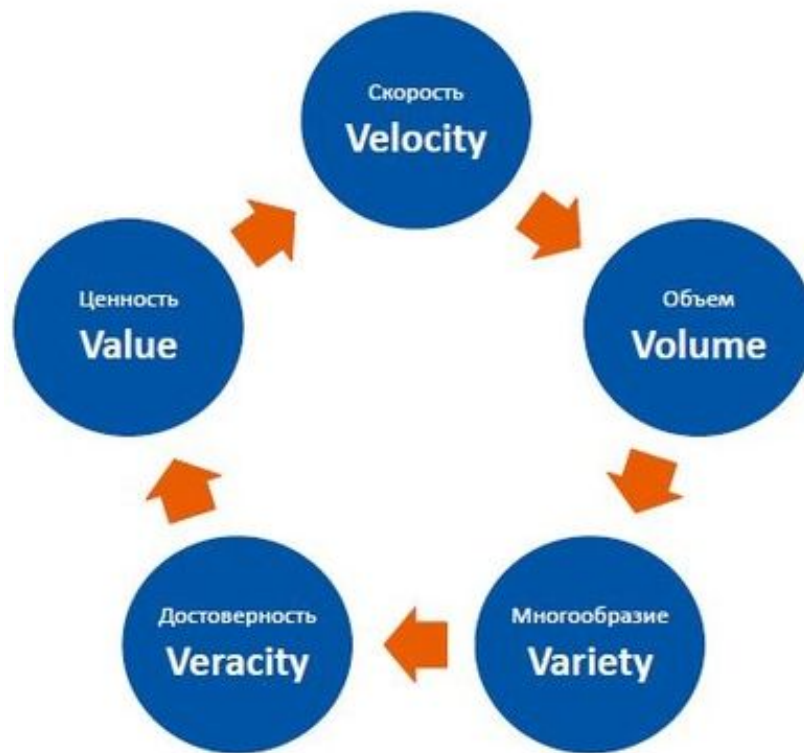
1. Введение. Используемые инструменты. Обзор интерфейса
2. Архитектура Spark. Планы выполнения запросов. Оптимизация запросов.
3. Типы данных, сложные запросы. Пользовательские функции
4. Машинное обучение

# Что будет на уроке

1. Что такое большие данные.
2. Spark API: python, java, scala
3. SparkSession. Параметры приложения spark
4. Форматы хранения данных parquet, avro.

# Что такое большие данные?

1. Время обработки > времени, в течении которых они приносят пользу
2. Объем: не помещается на 1 машине



| Характеристика                     | Традиционная база данных | База Больших Данных                       |
|------------------------------------|--------------------------|---|
| Объем информации                   | От гигабайт до терабайт  | От петабайт до эксабайт                   |
| Способ хранения                    | Централизованный         | Децентрализованный                        |
| Структурированность данных         | Структурирована          | Полуструктурирована или неструктурирована |
| Модель хранения и обработки данных | Вертикальная модель      | Горизонтальная модель                     |
| Взаимосвязь данных                 | Сильная                  | Слабая                                    |

Во многом относительное понятие, если вы Google, то 1Тб для вас не большие данные, вы можете их сохранить (даже на 1 машину), вы можете их обработать. Если вы небольшой стартап, то 100Гб могут стать проблемой....

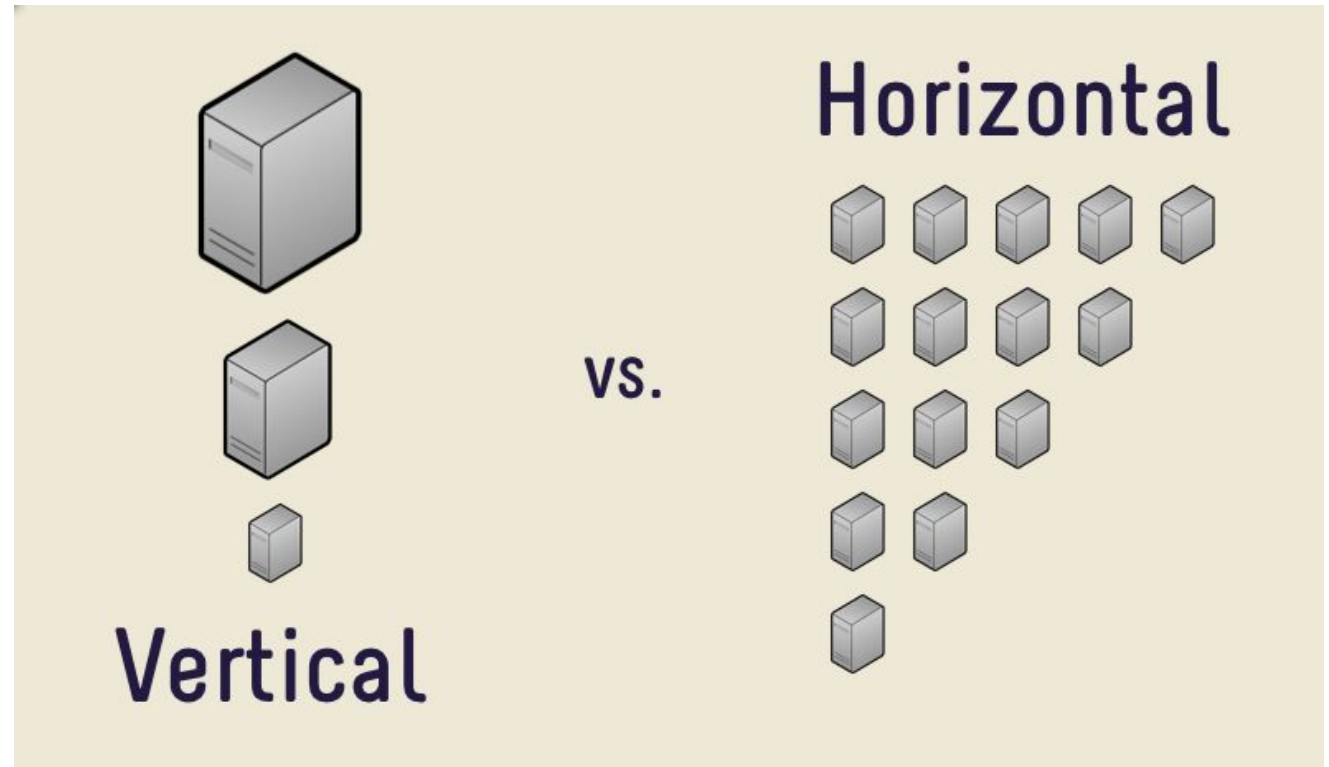
# Вертикальное vs горизонтальное масштабирование

Горизонтальное масштабирование:

- дешевле (много дешевых узлов)
- легко масштабировать (просто добавив новых узлов)
- ...

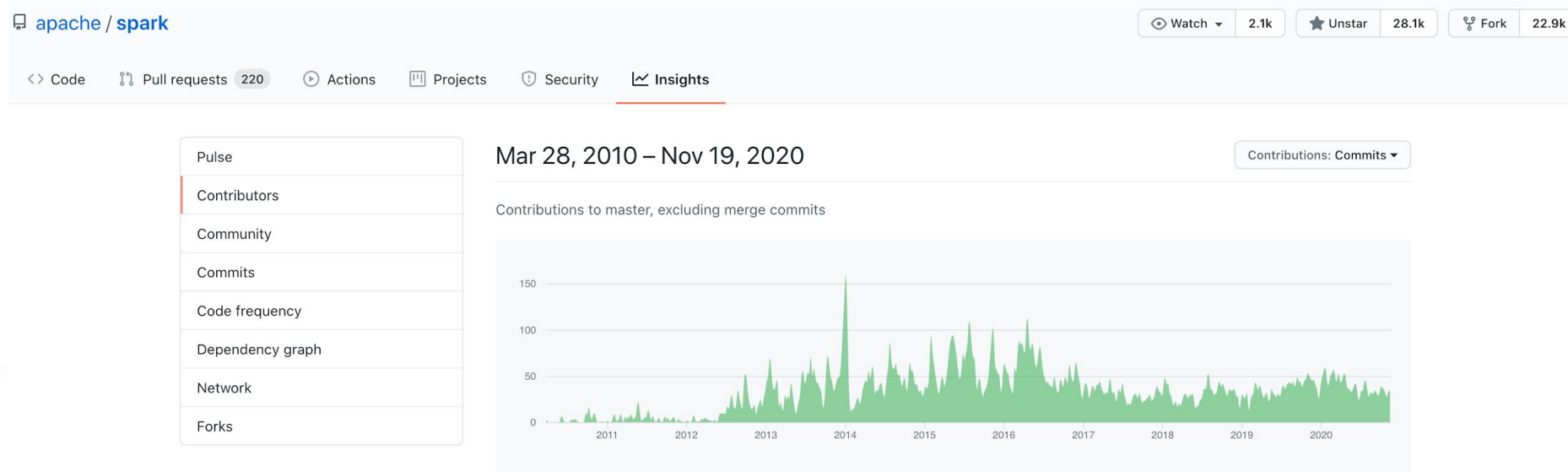
Как распределить данные? как их обрабатывать? На что обратить внимание?

1. Устойчивость к выбытию узлов
2. Парадигма “код к данным”, а не “данные к коду”
3. MapReduce!



# Почему Spark

1. Spark есть везде: банки (масштабируемость), средний и маленький бизнес (бесплатно)
2. Молодая технология, следовательно, мало специалистов
3. Универсальный фреймворк: создание хранилища, запросы к данным, потоковая обработка, ML.
4. Активное развитие



# Откуда пошло

[Google File System \(2003\)](#): Scalable distributed file system for large distributed data-intensive applications

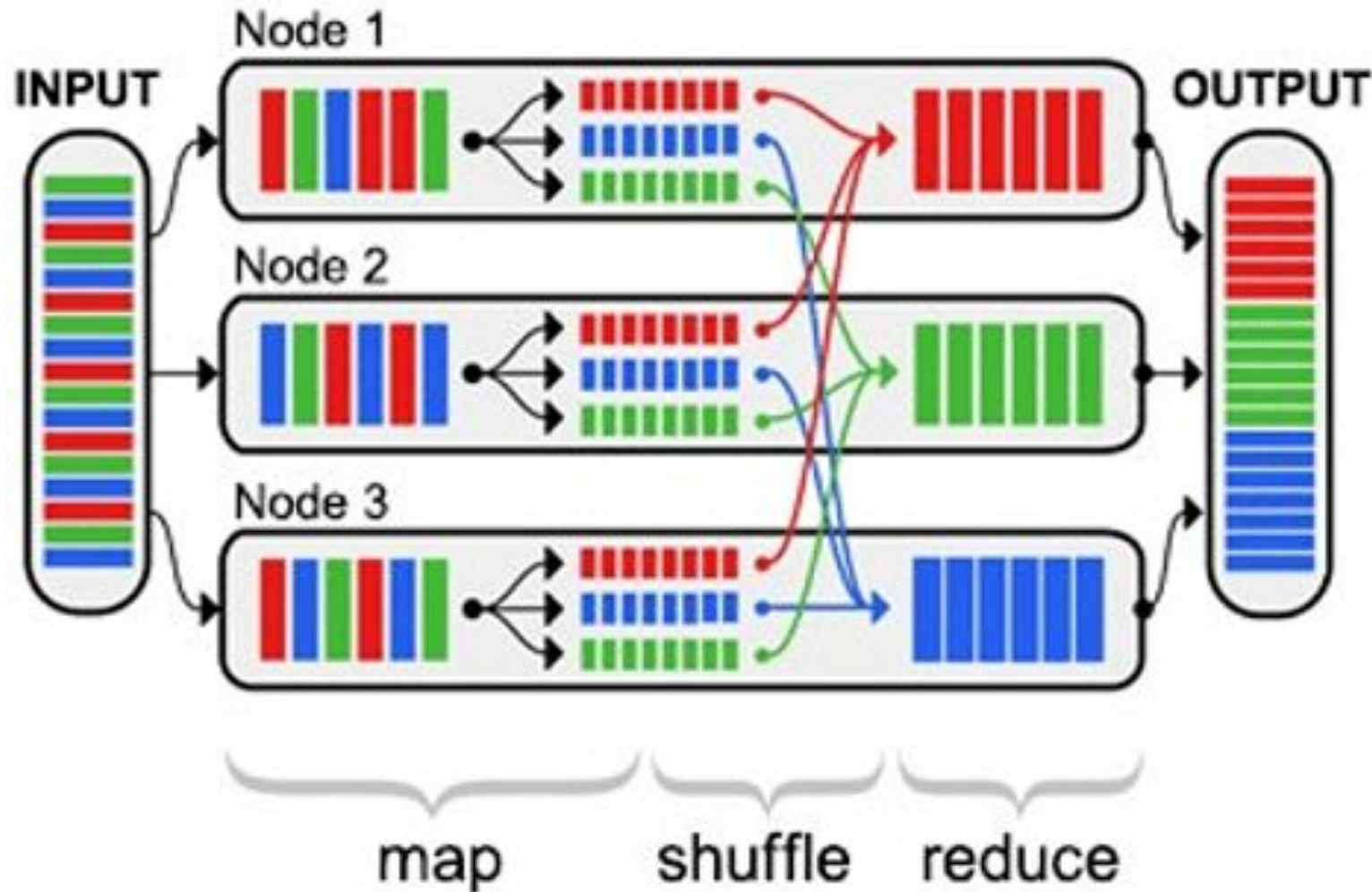
[Bigtable](#): scalable storage of structured data across GFS

[MapReduce \(MR\)](#) is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.



# Вспомним: что такое map-reduce?

## MapReduce



Какие неудобства?

1. Нужно писать все руками (трансформации данных, джобы)
2. Пайплан обработки тоже создается руками
3. Трудности с кешированием, переиспользованием
4. Записи промежуточных шагов на диск (медленно)

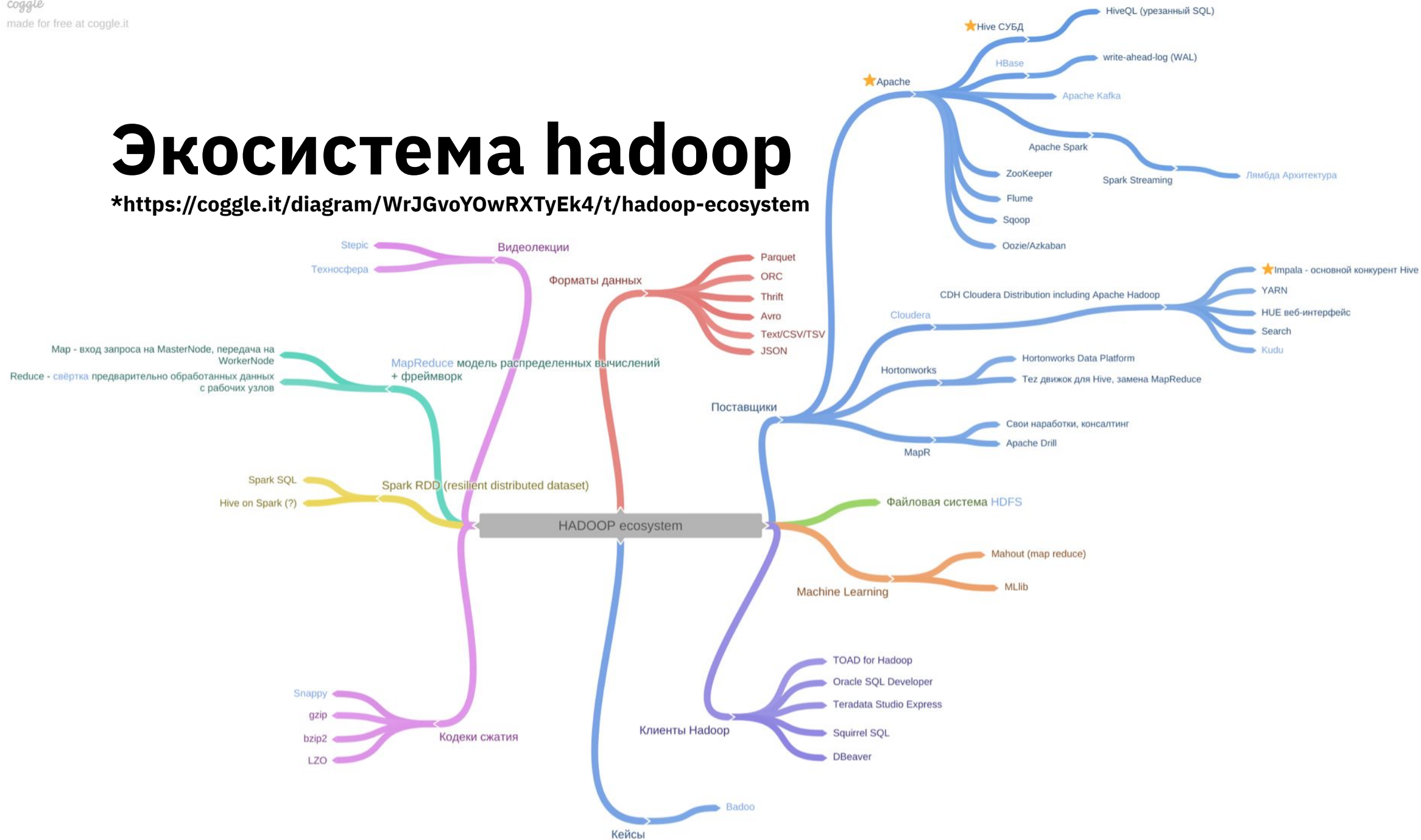
Логика сложнее джойна нескольких таблиц - дни разработки

## Hadoop Modules:

|  |  |
|--|--|
| <b>Others</b><br><b>(For Data Processing)</b>              | <b>MapReduce</b><br><b>(For Data Processing)</b> |
| <b>YARN</b><br><b>(Resource Management For Cluster)</b>    |  |
| <b>HDFS</b><br><b>(A Reliable &amp; Redundant Storage)</b> |  |

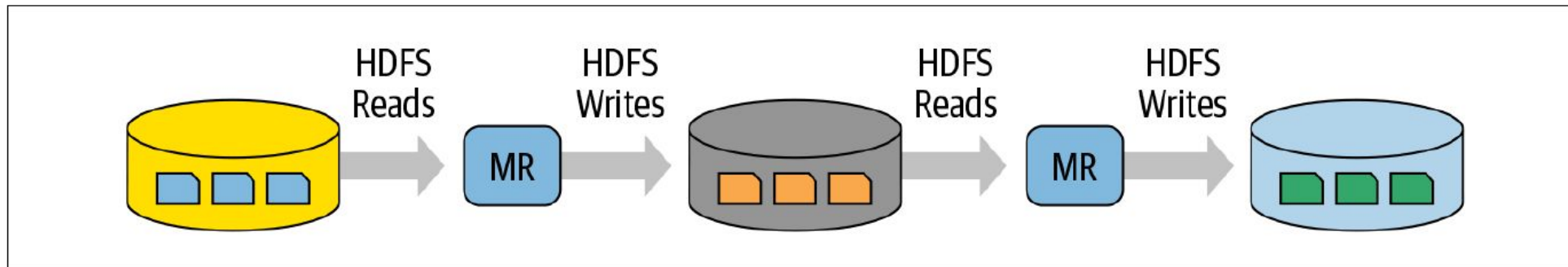
# Экосистема hadoop

\*<https://coggle.it/diagram/WrJGvoYOWRXTyEk4/t/hadoop-ecosystem>



# Недостаток Hadoop MapReduce:

промежуточные результаты последовательных операций записываются на диск

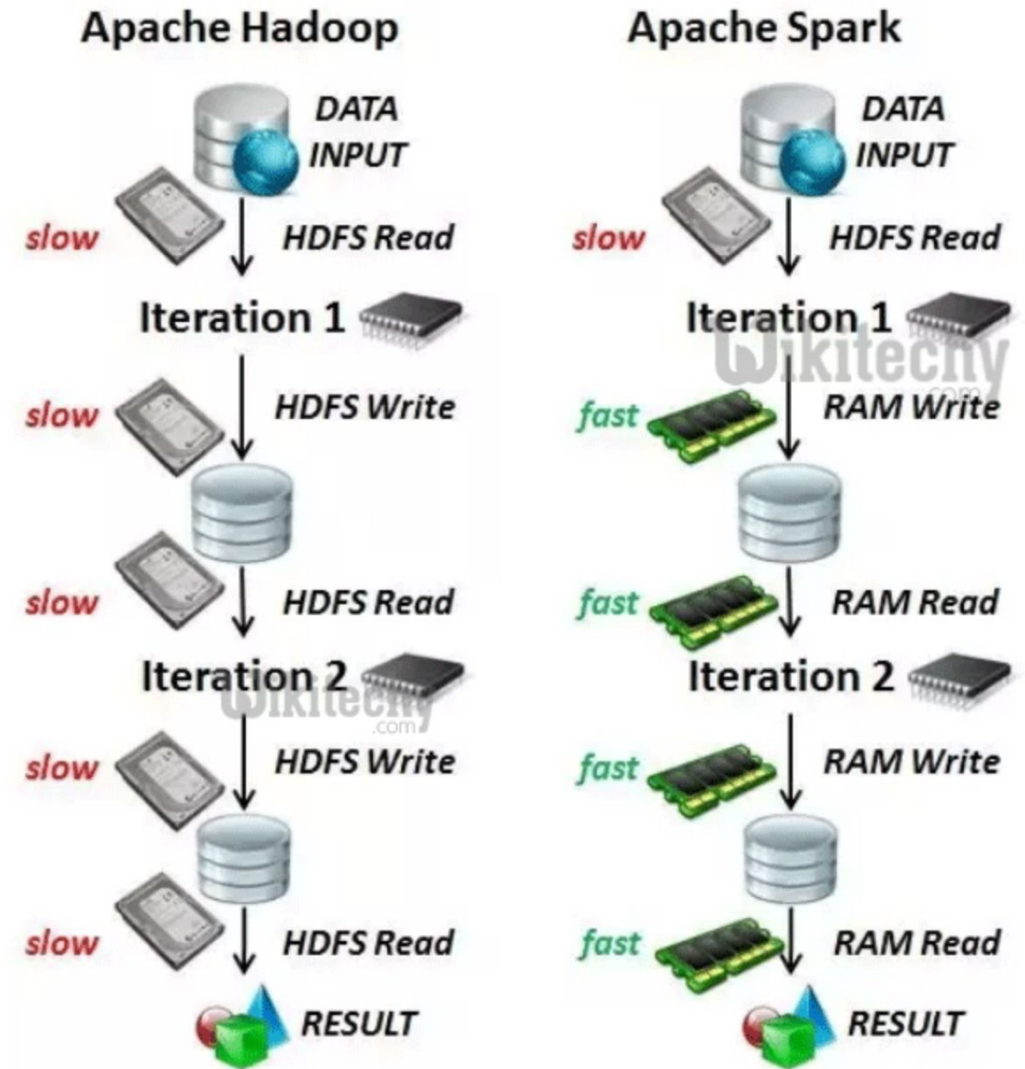


*Figure 1-1. Intermittent iteration of reads and writes between map and reduce computations*

Apache Hive (Tez), Apache Impala

# Spark (2009) vs Hadoop MapReduce:

1. in-memory storage for intermediate results between iterative and interactive map and reduce computations
2. offer easy and composable APIs in multiple languages as a programming model



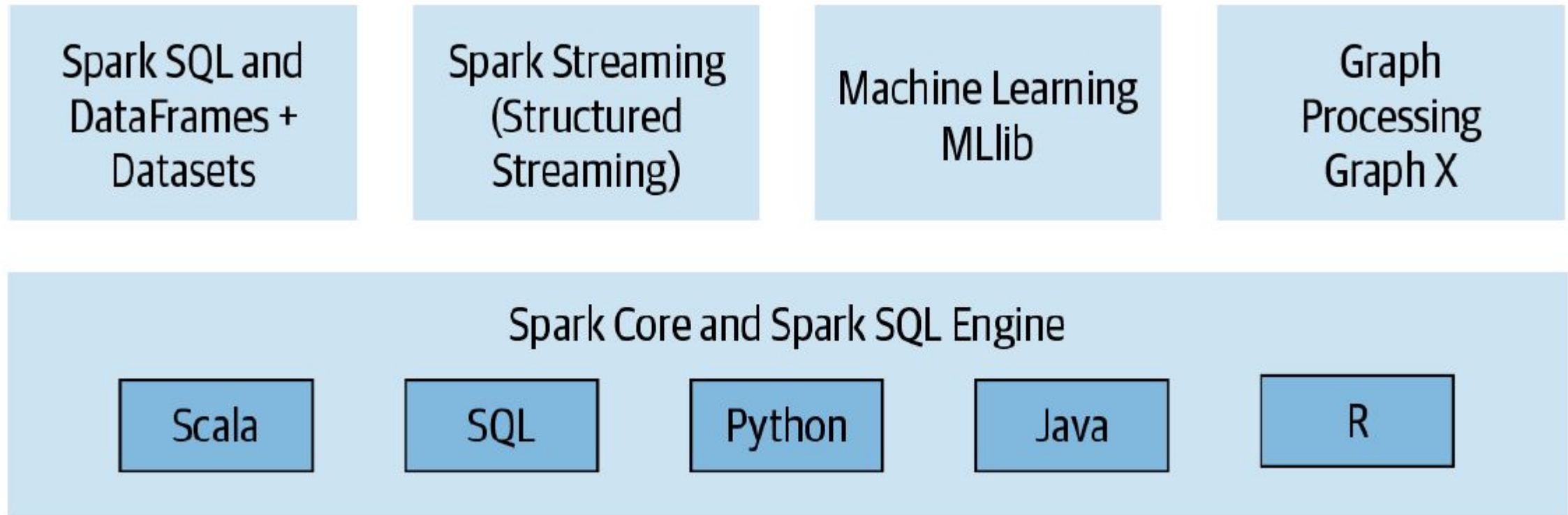
## Причины популярности Spark:

Со времен появления Hadoop MapReduce железо стало лучше по числу ядер CPU и памяти. Вычисления не выполняются последовательно, а оптимизируются после формирования полного запроса.

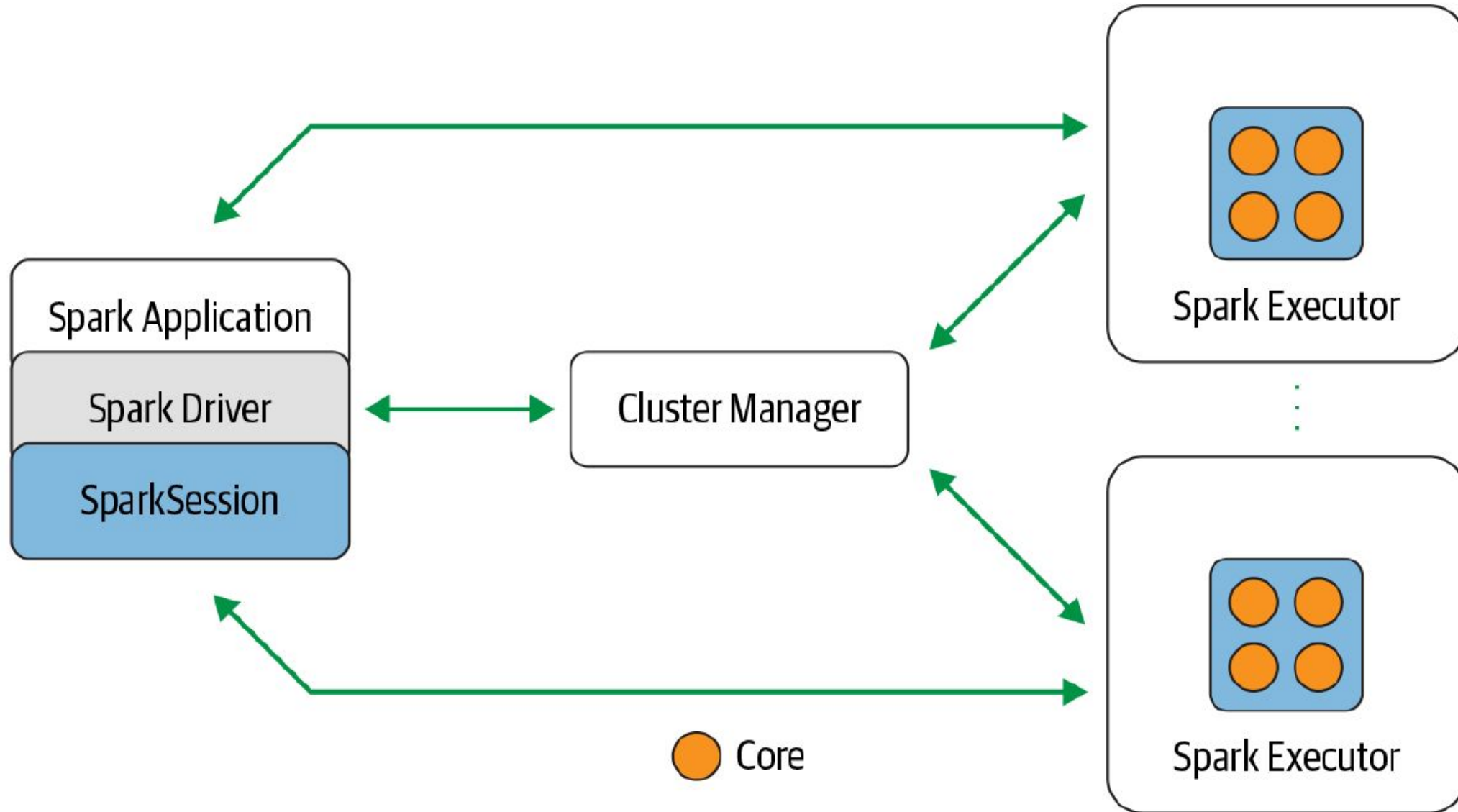
Потребность в разделении хранилища от вычислений: Spark может принимать данные из любых источников и обрабатывать их в памяти, пакетами (jdbc for SQL) или потоком (Kafka)



**Apache Spark** — фреймворк с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop.



# Архитектура приложения Spark





# Cluster manager

## Распределяет ресурсы между spark приложениями

### standalone cluster manager

FIFO исполнение приложений

### Apache Hadoop YARN

стандартное решение

распределяет, освобождает ресурсы между различными приложениями

Часть Hadoop

### Apache Mesos

YARN-like, but большая изолированность процессов

поддерживает не-hadoop приложения

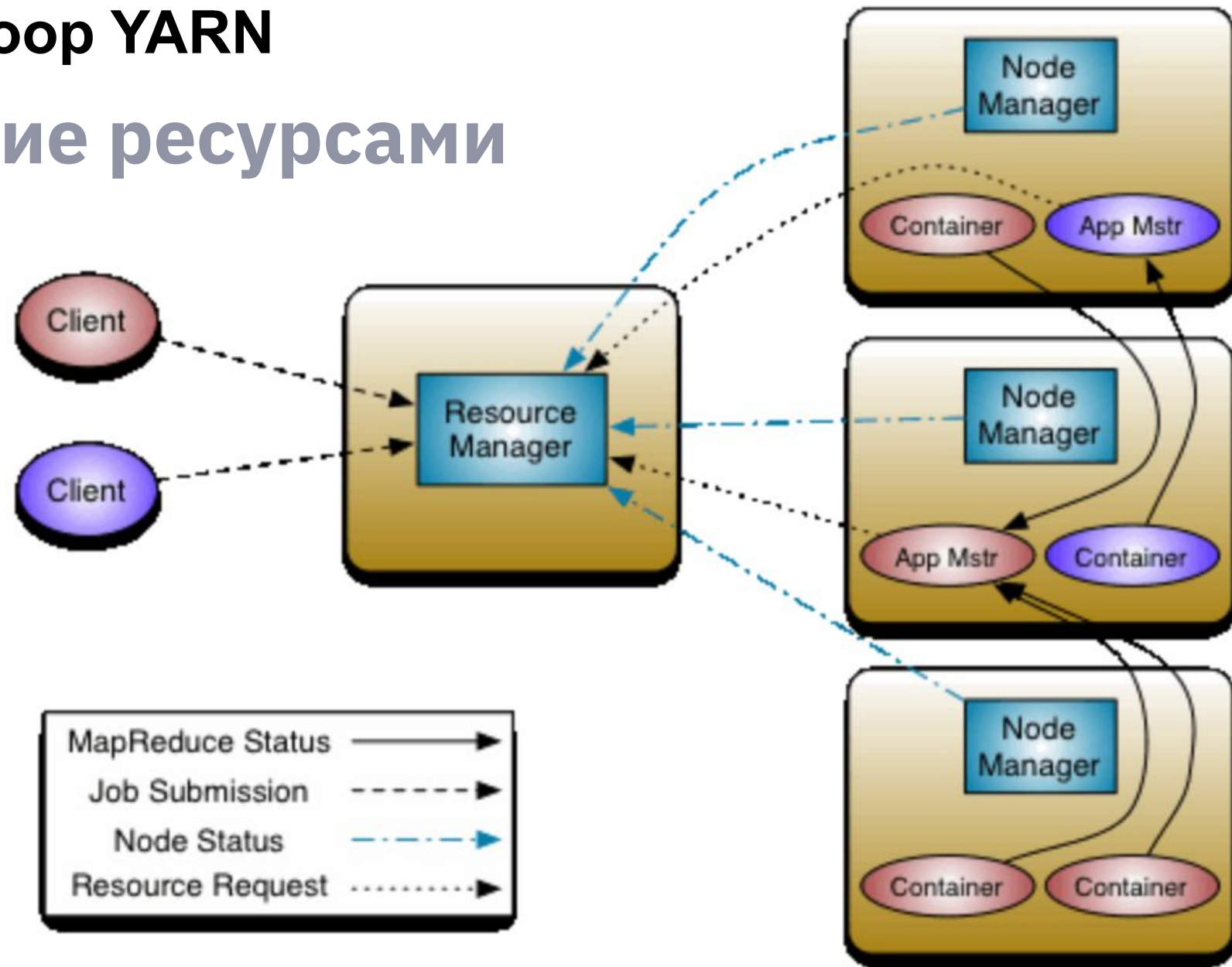
### Kubernetes

запуск в контейнерах = абсолютная изолированность

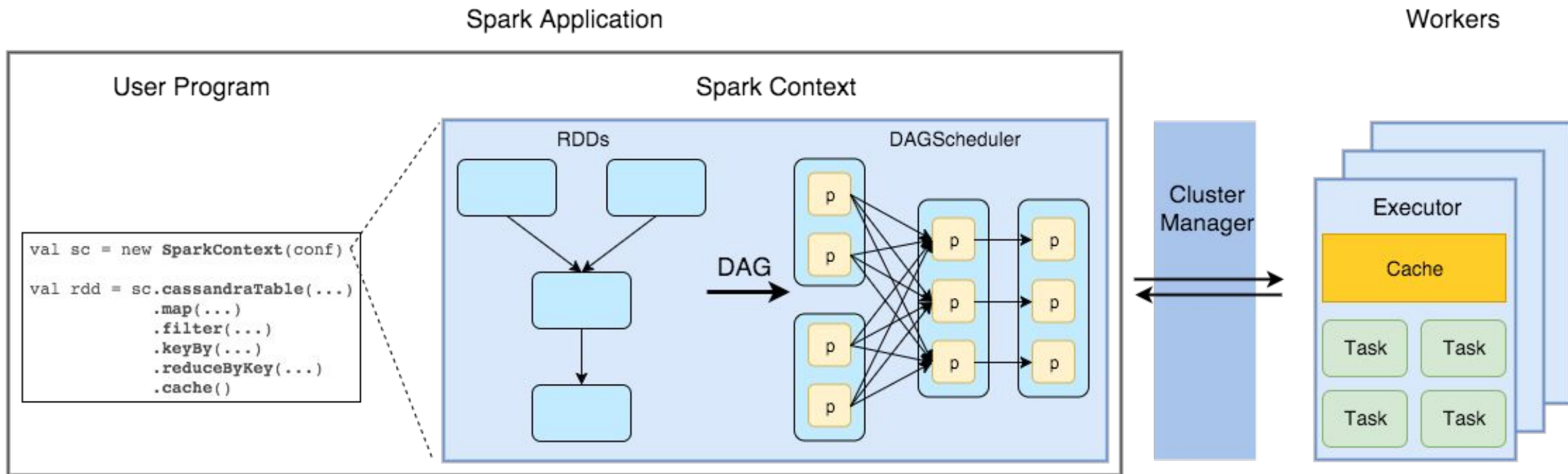
существенно сложнее в поддержке

# Apache Hadoop YARN

## Управление ресурсами



# Исполнение запроса



# Spark driver

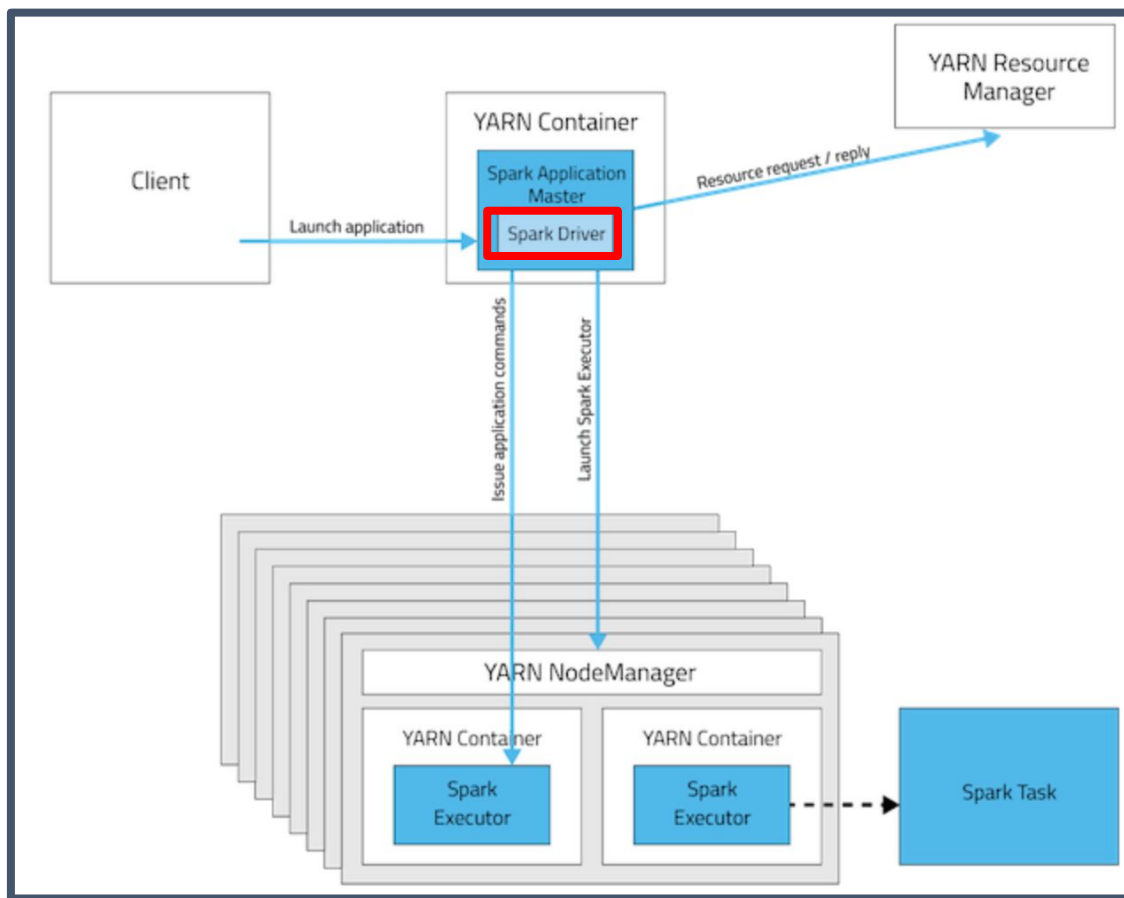
Может

находиться вне  
кластера

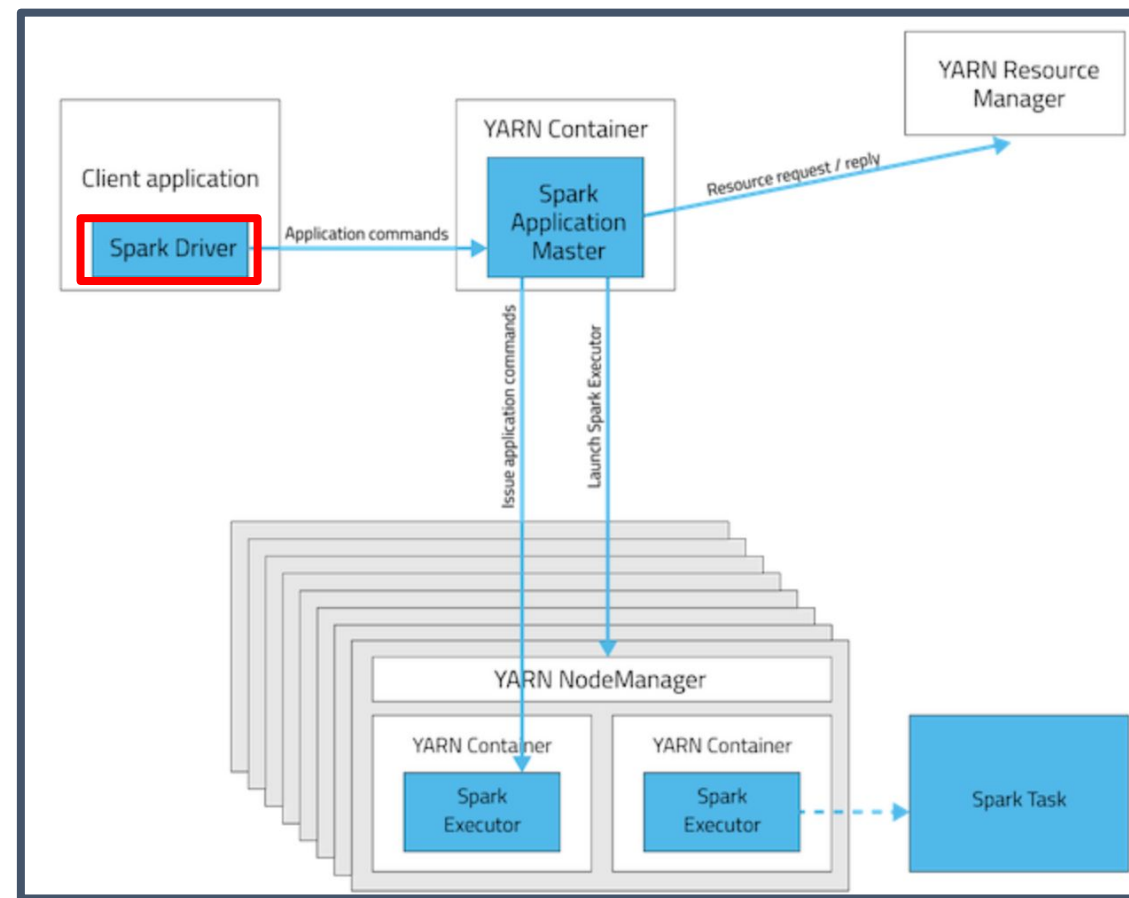
- it communicates with the cluster manager
- it requests resources (CPU, memory, etc.) from the cluster manager for Spark's executors (JVMs)
- it transforms all the Spark operations into DAG computations
- it distributes their execution as tasks across the Spark executors

# Spark on YARN deployment modes

## cluster mode

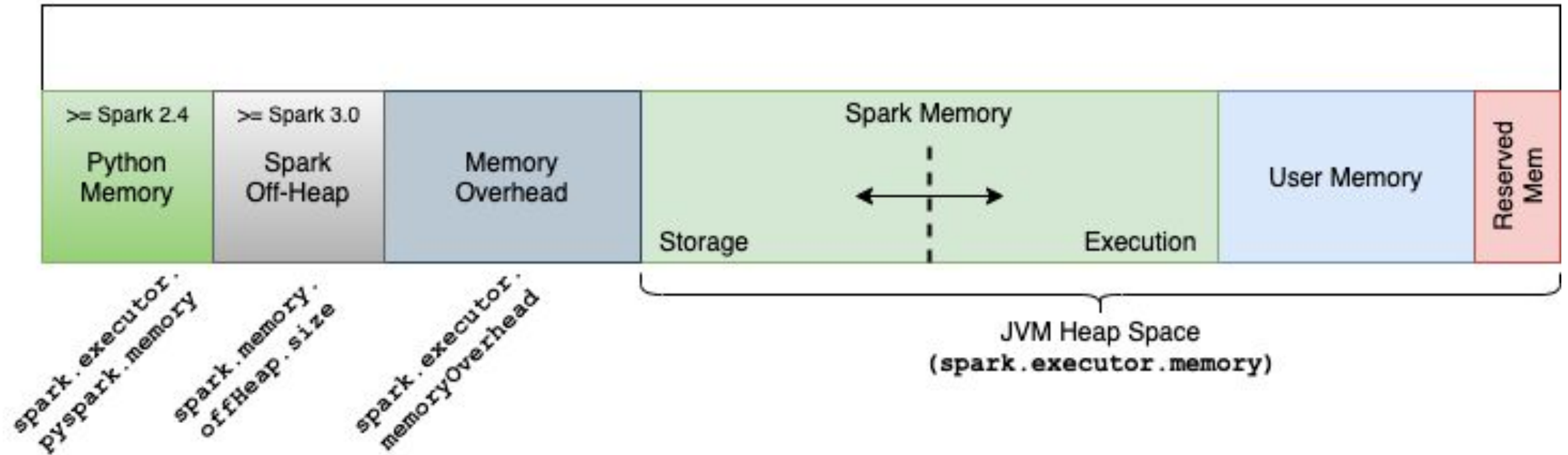


## client mode



# Executor

YARN Container



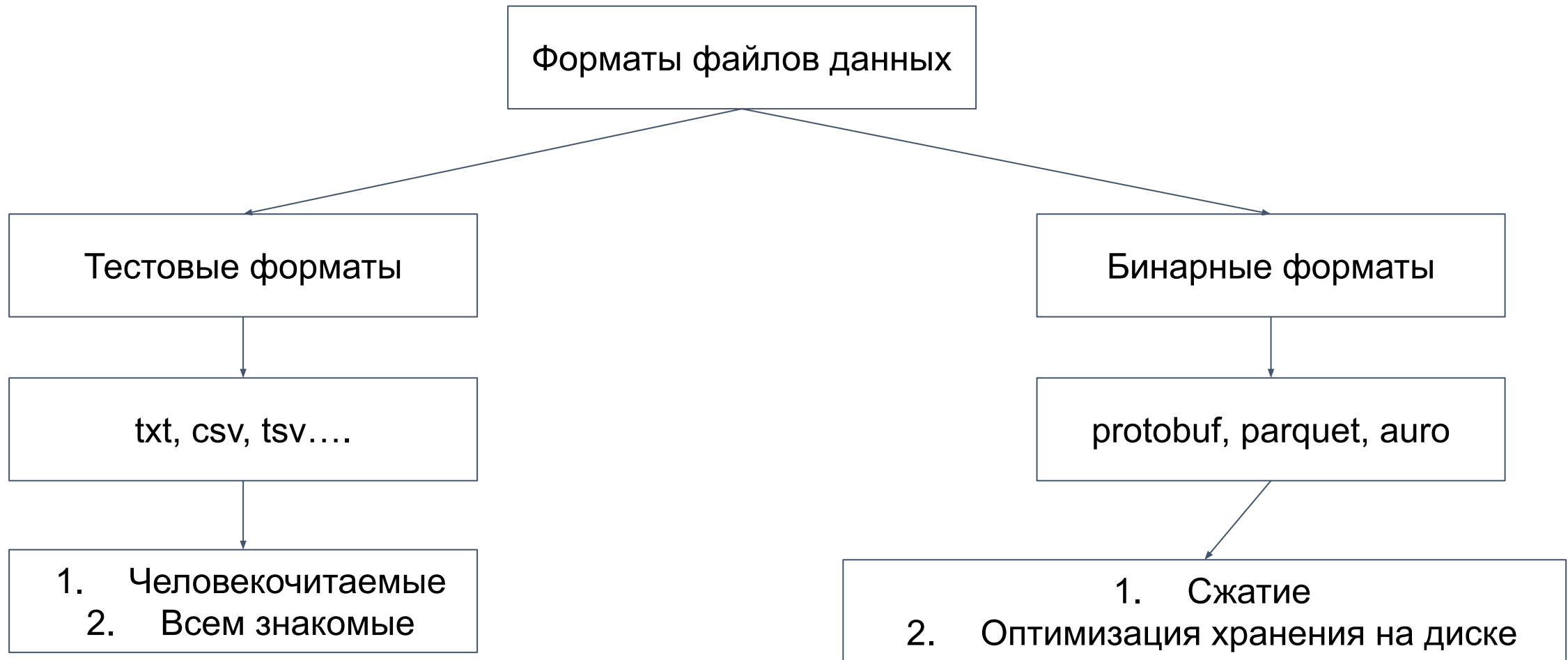
# SparkSession

## Точка входа в приложение

С его помощью можно:

- Управлять параметрами spark приложения
- читать и записывать данные в spark DataFrame
- исполнять SQL

```
1 from pyspark.sql import SparkSession
2
3 spark = SparkSession \
4     .builder \
5     .appName("Python Spark SQL basic example") \
6     .config("spark.some.config.option", "some-value") \
7     .getOrCreate()
```





# .\*sv

## Текстовые данные через разделитель

- Легко читать (если знаешь разделитель)

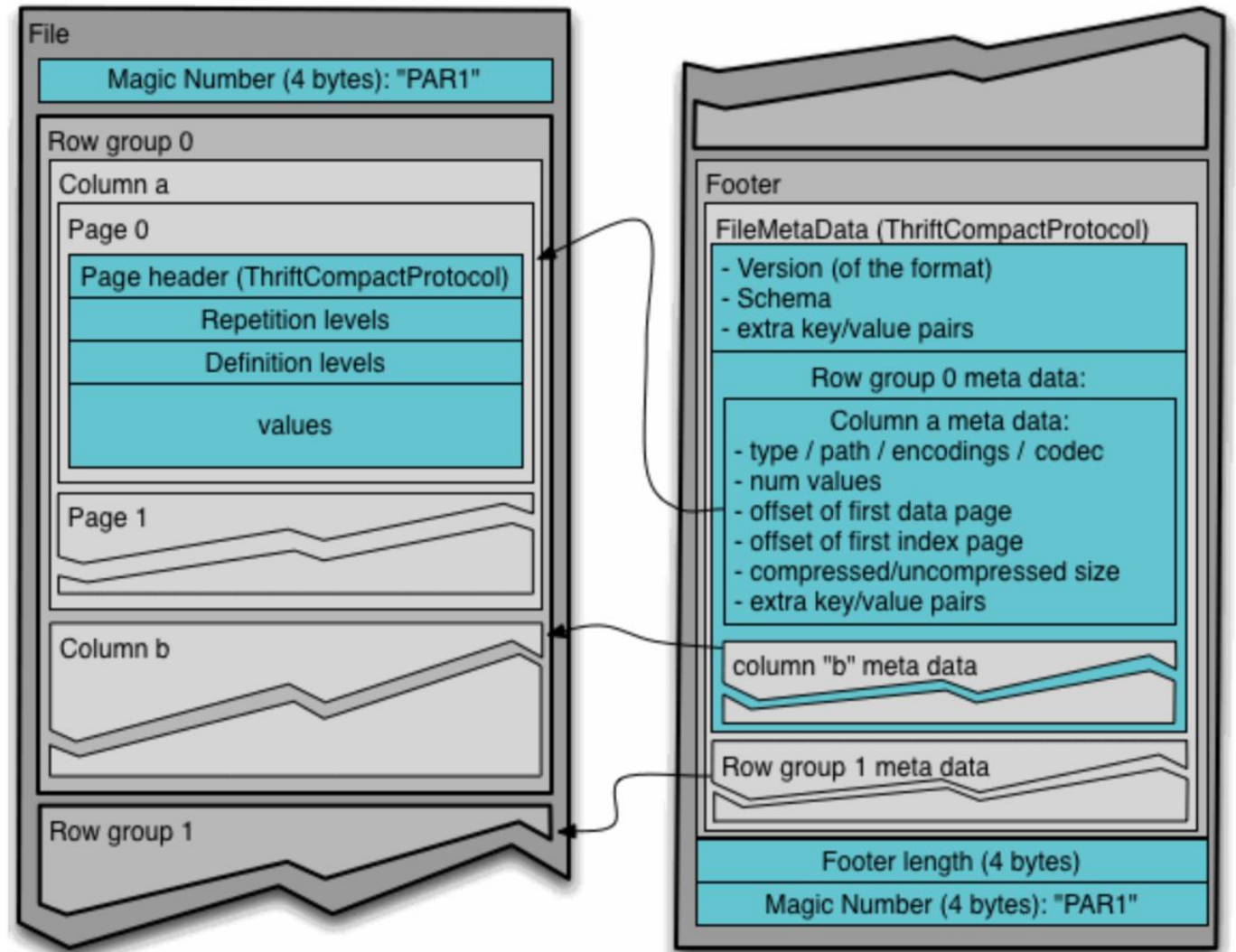


```
name, num1, num2, num3
hello,1,2,3
& this is a comment
hello,1,2,3
hello,1,2,3
hello,1,2,3
hello,1,2,3
hello,1,2,3
hello,1,2,3
hello,1,2,3
```

# Parquet file format

## Формат хранения данных для hadoop


- Колончатый формат
- Разное сжатие для разных типов данных



# Parquet vs CSV

## Меньше размер, быстрее чтение

The following table compares the savings as well as the speedup obtained by converting data into Parquet from CSV.

| Dataset   | Size on Amazon S3           | Query Run Time | Data Scanned          | Cost          |
|---|-----------------------------|----------------|-----------------------|---------------|
| Data stored as CSV files  | 1 TB                        | 236 seconds    | 1.15 TB               | \$5.75        |
| Data stored in Apache Parquet Format  | 130 GB                      | 6.78 seconds   | 2.51 GB               | \$0.01        |
|  Savings | 87% less when using Parquet | 34x faster     | 99% less data scanned | 99.7% savings |

# Типы данных в Spark

| Data type     | Value type  | API to access or create data type  |
|---------------|---|--|
| ByteType      | int or long <b>Note:</b> Numbers are converted to 1-byte signed integer numbers at runtime. Make sure sure that numbers are within the range of -128 to 127.  | ByteType()   |
| ShortType     | int or long <b>Note:</b> Numbers are converted to 2-byte signed integer numbers at runtime. Make sure sure that numbers are within the range of -32768 to 32767.  | ShortType()  |
| IntegerType   | int or long   | IntegerType()  |
| LongType      | long <b>Note:</b> Numbers are converted to 8-byte signed integer numbers at runtime. Make sure sure that numbers are within the range of -9223372036854775808 to 9223372036854775807. Otherwise, convert data to decimal.Decimal and use DecimalType. | LongType()   |
| FloatType     | float <b>Note:</b> Numbers are converted to 4-byte single-precision floating point numbers at runtime.  | FloatType()  |
| DoubleType    | float   | DoubleType()   |
| DecimalType   | decimal.Decimal   | DecimalType()  |
| StringType    | string  | StringType()   |
| BinaryType    | bytearray   | BinaryType()   |
| BooleanType   | bool  | BooleanType()  |
| TimestampType | datetime.datetime   | TimestampType()  |
| DateType      | datetime.date   | DateType()   |
| ArrayType     | list, tuple, or array   | ArrayType(elementType, [containsNull]) <b>Note:</b> The default value of containsNull is True. |

# Типы данных в Spark

|             |   |   |
|-------------|---|---|
| ArrayType   | list, tuple, or array   | ArrayType(elementType, [containsNull]) <b>Note:</b> The default value of containsNull is True.                        |
| MapType     | dict  | MapType(keyType, valueType, [valueContainsNull]) <b>Note:</b> The default value of valueContainsNull is True.         |
| StructType  | list or tuple   | StructType(fields) <b>Note:</b> fields is a Seq of StructFields. Also, two fields with the same name are not allowed. |
| StructField | The value type of the data type of this field (For example, Int for a StructField with the data type IntegerType) | StructField(name, dataType, [nullable]) <b>Note:</b> The default value of nullable is True.                           |

# Настройка pyspark

## Полезные ссылки

Настройка pyspark на локальном  
компьютере

```
from pyspark.sql import SparkSession

spark = SparkSession.builder\
    .master("local")\
    .appName("My_notebook")\
    .getOrCreate()
```

spark

**SparkSession - in-memory**  
**SparkContext**

[Spark UI](#)

**Version**

v3.1.2

**Master**

local

**AppName**

My\_notebook

# Полезные ресурсы

<https://sparkbyexamples.com>

[parquet doc](#)

[A Neanderthal's Guide to Apache Spark in Python](#)

[Matei Zaharia. Презентация от автора Spark](#)



**Спасибо!**  
**Каждый день**  
**вы становитесь**  
**лучше :)**

