Работа с источниками данных (ввод-вывод в Spark)

МОДУЛЬ 6

ШКОЛА БОЛЬШИХ ДАННЫХ

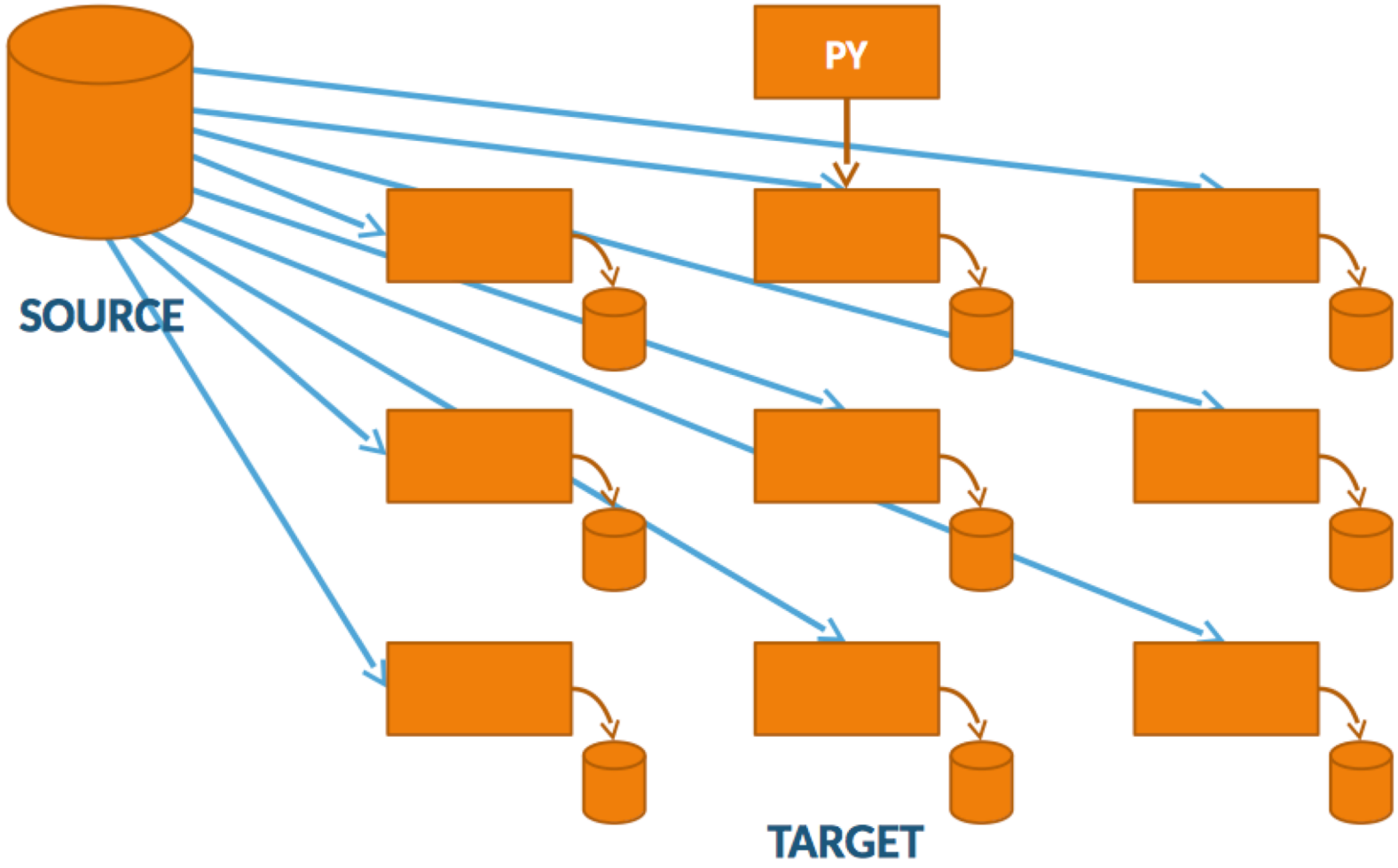
Hadoop и большие данные

- **Big Data**
- **data intensive** вычисления
- hadoop **кластер**
- парадигма "**доставки вычисления к данным**"

Загрузка и выгрузка данных

- данные находятся **вне кластера**
- структурированные и неструктурированные
- задачи
  - ✓ загрузить в кластер
  - ✓ выгрузить из кластера

Идеальный мир



Если без Spark

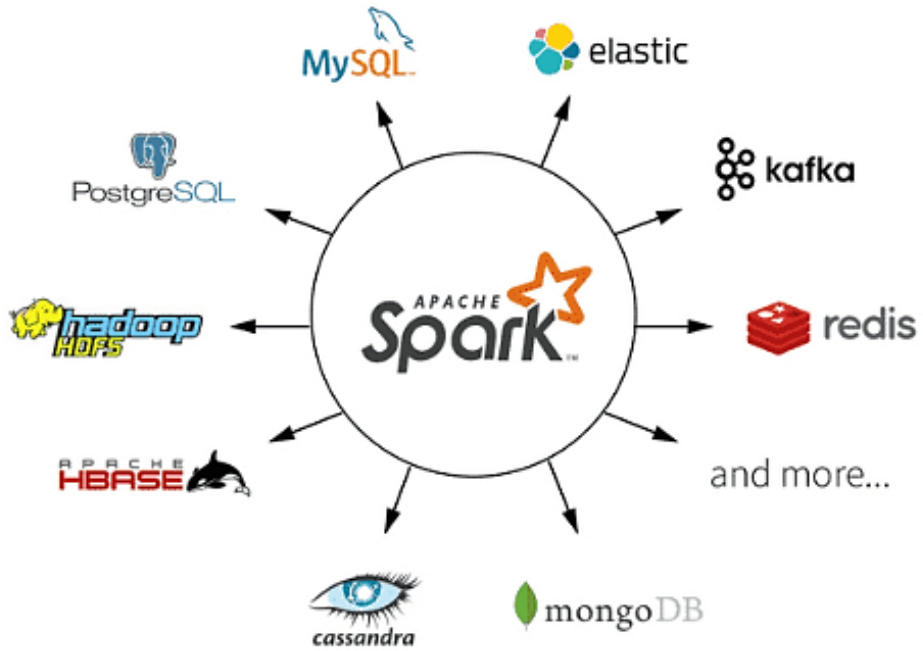
- базы данных
  - ✓ sqoop
- файлы
  - ✓ Hive SerDe

Что дает Spark

- универсальный **простой механизм** ввода-вывода
- параллелизм с простой настройкой
- совместную работу с **Hive**
- эффективное **встраивание** в конвейер обработки

Виды источников

- **базы данных**
  - ✓ JDBC, Hive
- **файлы** (в **HDFS**)
  - ✓ структурированные
    - ➢ CSV, ORC, Parquet, Avro, text
  - ✓ семиструктурированные
    - ➢ JSON, XML, текст
  - ✓ двоичные



Простой универсальный механизм ввода-вывода

Загрузка данных

```
df = spark.read.format() \
.option() \
.load()
```

Выгрузка данных

```
df.write.format() \
  .mode() \
    .option() \
.save() или .saveAsTable()
```

Основные параметры операций ввода вывода

format: форматы

- ✓ “CSV”, “JSON”, “Parquet”, “ORC”, “JDBC”

option: опции

- ✓ bool "inferSchema"
- ✓ bool "header" (для CSV)
- ✓ string "path"
- ✓ string "compression"
- ✓ string "url", "dbtable", "user", "password" (для таблиц)

mode: режим

- ✓ “append” (без удаления)/”overwrite” - с удалением

Параллелизм ввода вывода в Spark

чем определяется **параллелизм** обработки

- ✓ экзекьюторы
- ✓ партиции

при работе с источниками – по-разному

- ✓ чтение и запись
- ✓ файлы и базы данных

Как влиять на параллелизм

- **экзекьюторы**: параметр запуска приложения
- **партиции**
  - ✓ **при записи**: из dataframe
  - ✓ **при чтении**: возможности источника

Вопросы?