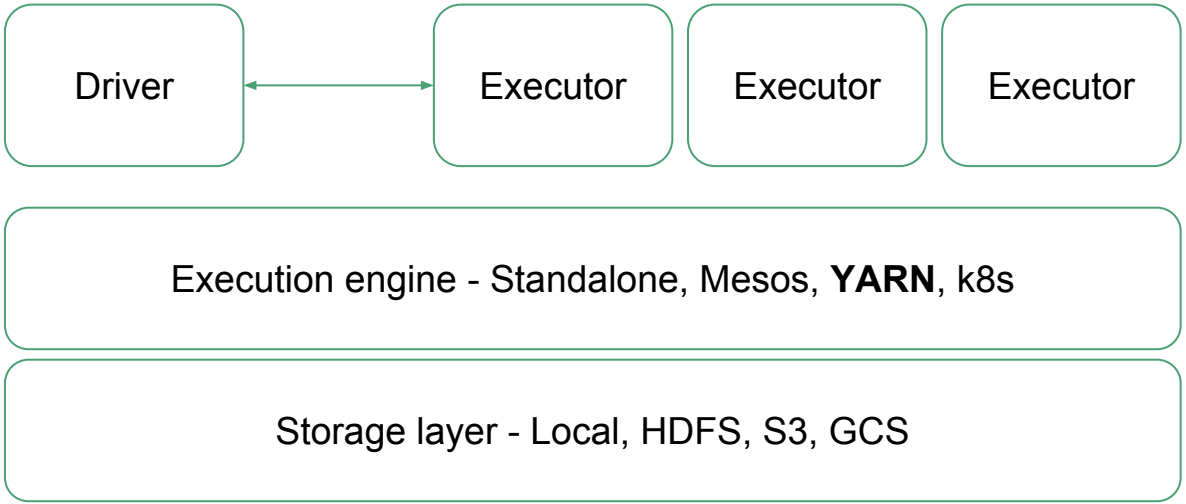
Metody przetwarzania danych

Apache Spark

Agenda

1. Zasada działania
2. Wykorzystanie z poziomu języków programowania
3. Spark SQL
4. Przykładowe zadanie w Sparku (część warsztatowa)

Apache Spark



Apache Spark

- Framework
- RDD (resilient distributed dataset)
- RDD -> Dataset -> DataFrame
- Transformacje
- Akcje
- Lazy evaluation
- SQL (katalog w Hive)
- Aktywny development
- Dynamic Resource Allocation
- Łatwa integracja z zewnętrznymi systemami

Apache Spark

- Java
- Scala
- Python
- R

Apache Spark

- Spark Streaming
- MLlib
- GraphX

Apache Spark

```scala
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SparkApp {
 def main(args: Array[String]) {

        // Create SparkContext
        val conf = new SparkConf().setAppName("SparkApp")
        val sc = new SparkContext(conf)

        val distFile = sc.textFile("data.txt")

       //cleanly shutdown
       sc.stop()
 }
}
```

Apache Spark

```java
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.*;

public final class SparkApp {

    public static void main(String[] args) throws Exception {

       //Create SparkContext
        JavaSparkContext sc = new JavaSparkContext(
                    new SparkConf().setAppName("SparkApp")
                    );

        JavaRDD<String> distFile = sc.textFile("data.txt");


       //cleanly shutdown
       sc.stop();
       System.exit(0);
    }
}
```

Apache Spark

```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setAppName(appName).setMaster(master)
spark = SparkContext(conf=conf)
distFile = spark.textFile("data.txt")
```

Na co zwracać uwagę ?

- GC
- Limity pamięci (executor-memory, driver-memory)
- Czasy wykonania
- Spark History Server