Apache Spark

Apache Spark

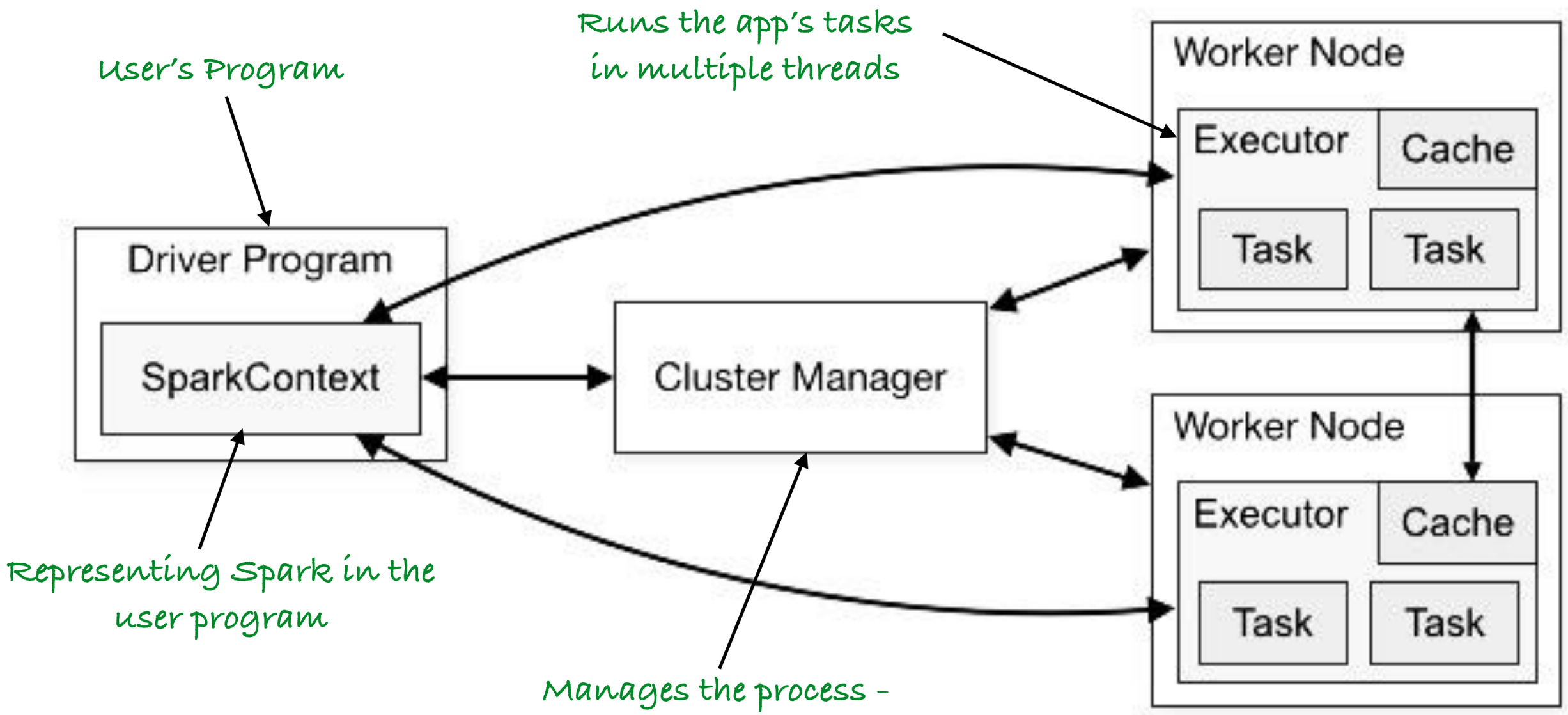
Open source data processing framework for performing Big data analytics on distributed computing cluster.

Spark uses HDFS and replaces M/R with other advanced processing.

Spark Vs. Hadoop

- In-Memory use. Good for iterations. (Vs. single path and disk writes)
- Wide variety of operations like SQL, Streaming, Graph actions. (Vs. map/reduce)
- Richer API in multiple languages

Spark Architecture



Spark Architecture

The main object in Spark is a Resilient Distributed Dataset (RDD).

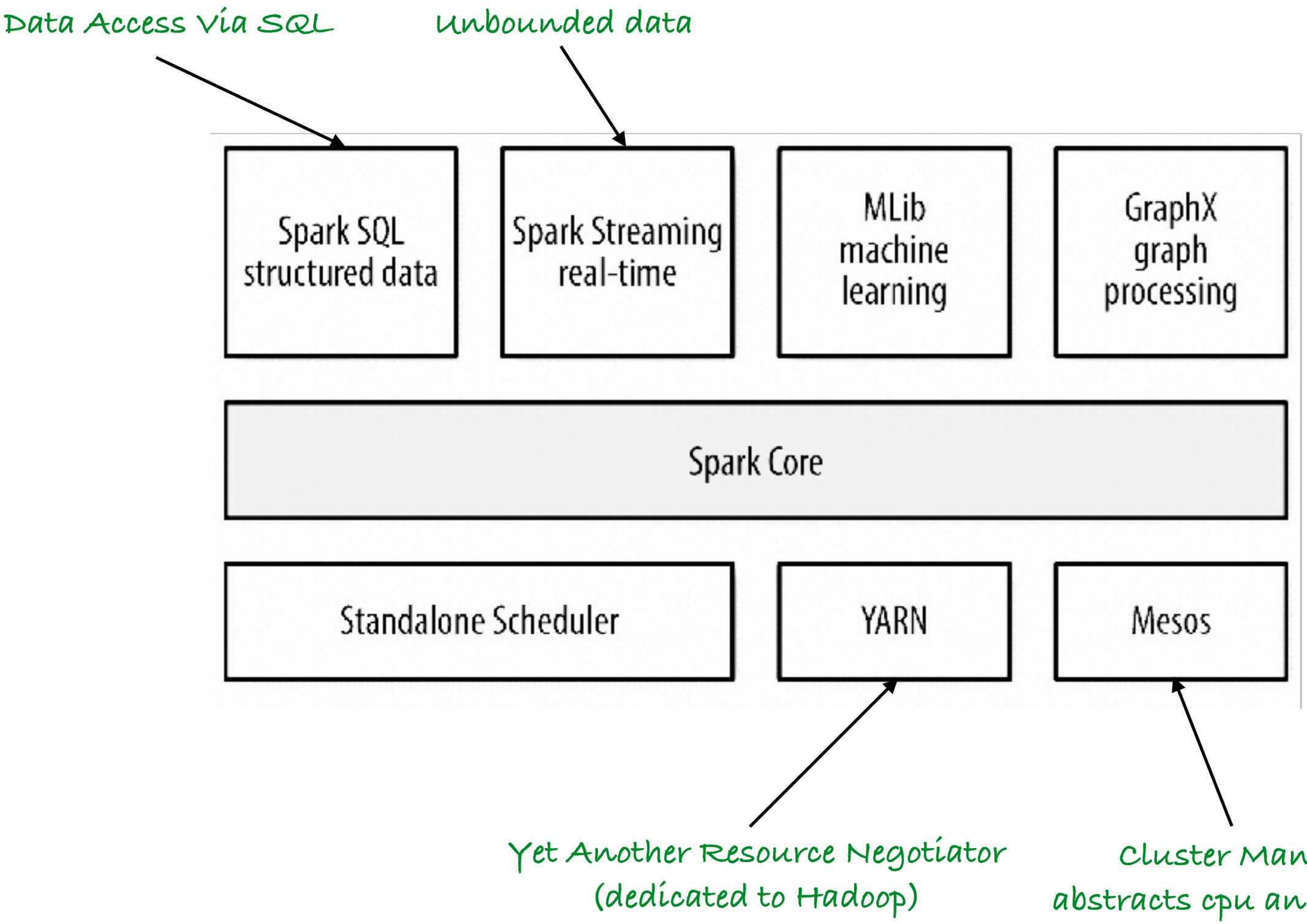
RDDs have:

- Actions that return values, like Reduce or Collect.
- Transformations that return pointers to new RDDs, like Map or Filter

Spark Architecture

A ‘Driver’ program starts parallel operations on any RDD by passing a function to Spark’s core, which further schedules the function's execution in parallel on the cluster.

Spark Echo System



Work with Spark Shell

- Setup Spark…
- Run shell:
  - **./bin/pyspark**
- Python examples:
  - **readMeFile = sc.textFile(“README.md")**
  - **readMeFile.count()**
  - **readMeFile.first()**
  - **linesWSpark = readMeFile.filter(lambda line: "Spark" in line)**
  - **linesWSpark.count()**
  - **readMeFile.filter(lambda line: "Spark" in line).count()**

Work with Spark Shell

- Map/Reduce:
  - **readMeFile.map(lambda line:**

    len(line.split())).reduce(lambda a, b: a if (a > b) else b)
- Using a function:
  - **def max(a, b):**

    if a > b:

    return a

    else:

    return b
  - **readMeFile.map(lambda line: len(line.split())).reduce(max)**

Work with Spark Shell

- More Map/Reduce:
  - **wordCounts = readMeFile.flatMap(lambda line: line.split()).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)**
  - **wordCounts.collect()**

Work with Spark Shell

- Caching (impact is seen over large number of nodes):
  - **linesWSpark.cache()**
  - **linesWSpark.count()**

- Stages - long tasks are broken into execution stages
  - **largeFile = sc.textFile(“english.200MB")**
  - **largeFile.count()**

Work with Spark Shell

- Shuffling:
  - Redistributing data across partitions.
  - Data transfer between stages

Writing apps

```
#app.py
from pyspark import SparkContext

longFile = "./english.50MB"
sc = SparkContext("local", "Demo App")
longData = sc.textFile(longFile).cache()

numAs = longData.filter(lambda s: 'a' in s).count()
numBs = longData.filter(lambda s: 'b' in s).count()

print("Lines with a: %i, lines with b: %i" % (numAs, numBs))

sc.stop()
```

Spark SQL

- Working with data:
  - **df = spark.read.json("examples/src/main/resources/people.json")**
  - **df.show()**
  - **df.select("name").show()**
  - **df.printSchema()**

- Running SQL Queries
  - **df.createOrReplaceTempView("people")**
  - **sqlDF = spark.sql("SELECT * FROM people")**
  - **sqlDF.show()**

Spark ML

- Spark Machine Learning :
  - **./bin/spark-submit /usr/local/bigdata/spark/ml_clustering.py**
  - **./bin/spark-submit /usr/local/bigdata/spark/ml_classification.py**