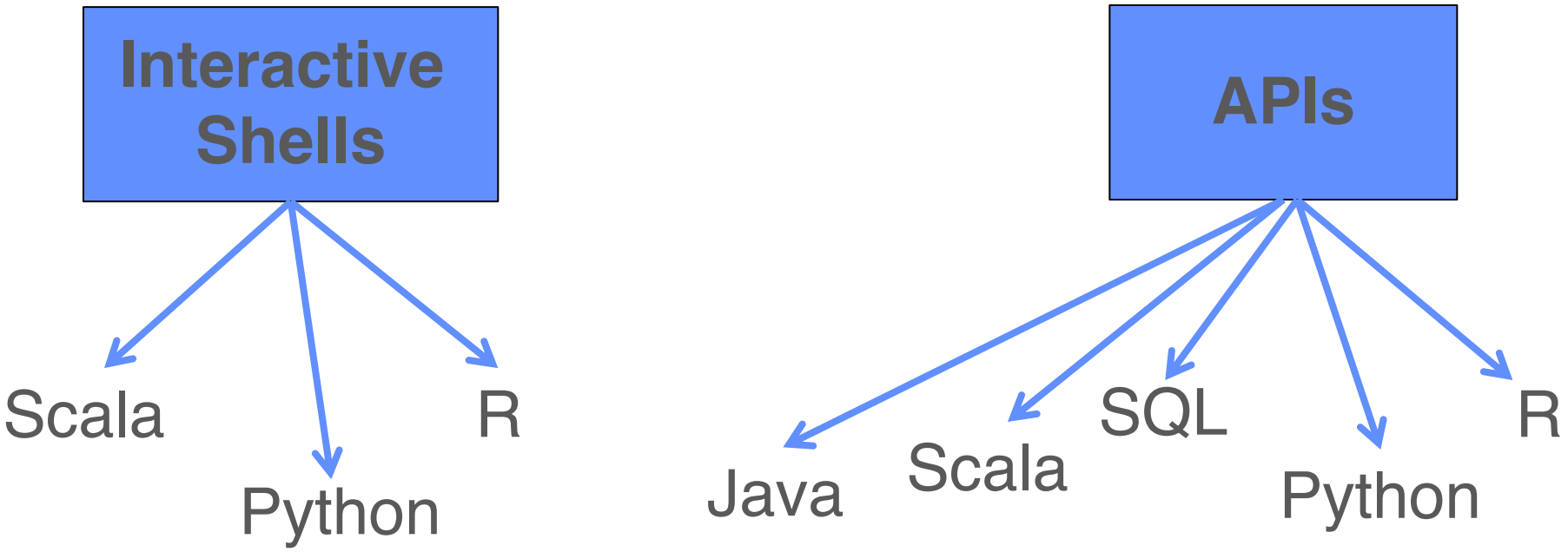
Machine Learning with Spark

Mai H. Nguyen, Ph.D.

Spark

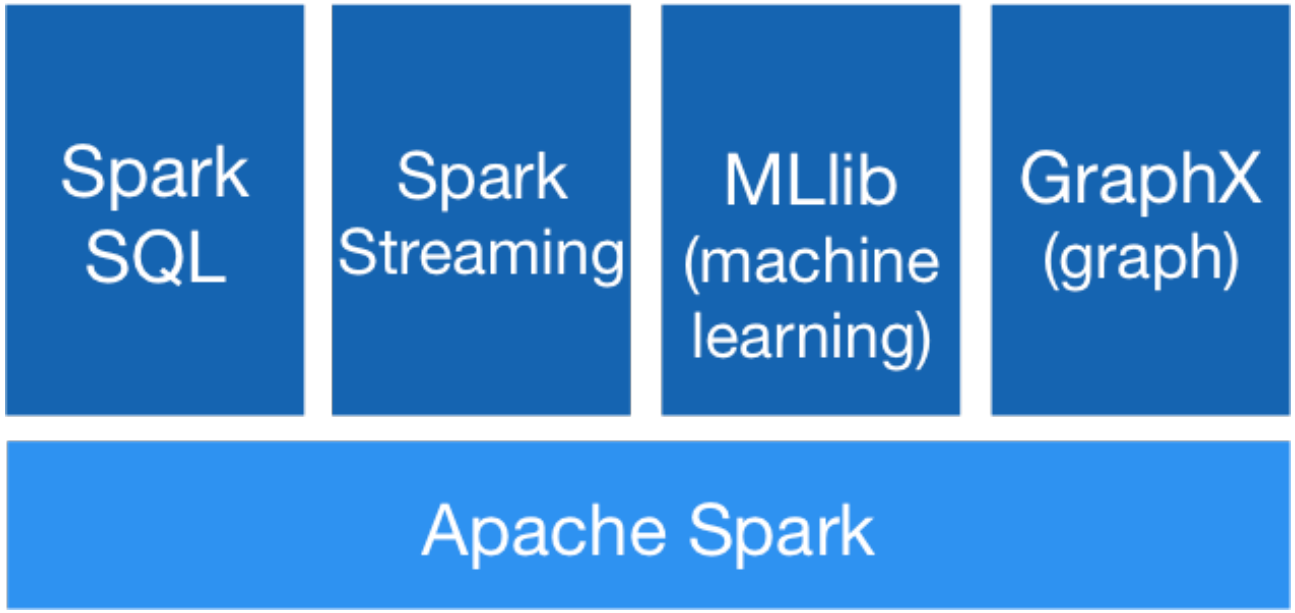
- **Computing platform for distributed processing**
- **Provides built-in data parallelism & fault-tolerance for big data processing on cluster**
- **Goals:**
  - Speed
  - Ease of use
  - Generality
  - Unified platform
- **Open-source**

Spark Interface

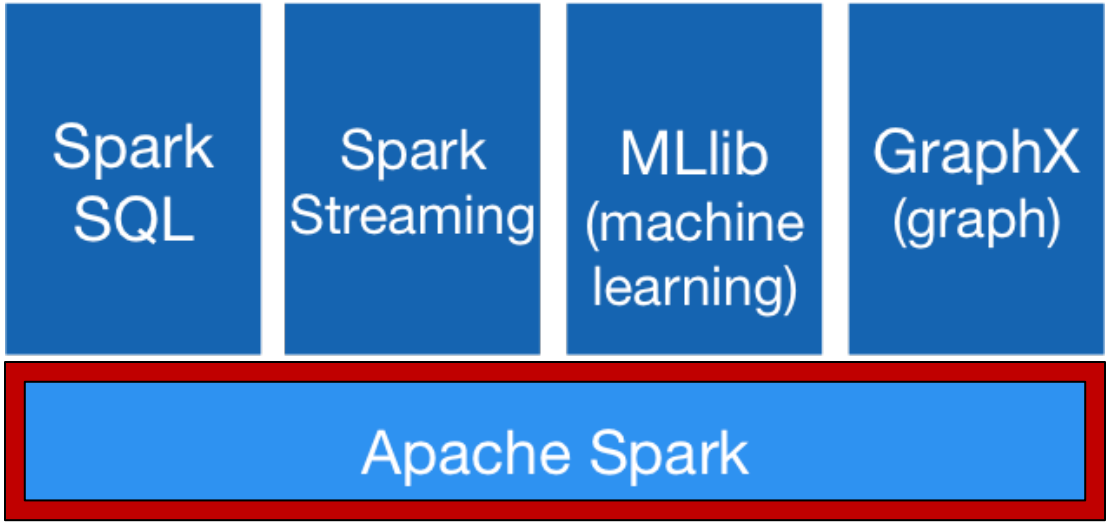


Provide ease of use

Spark Stack

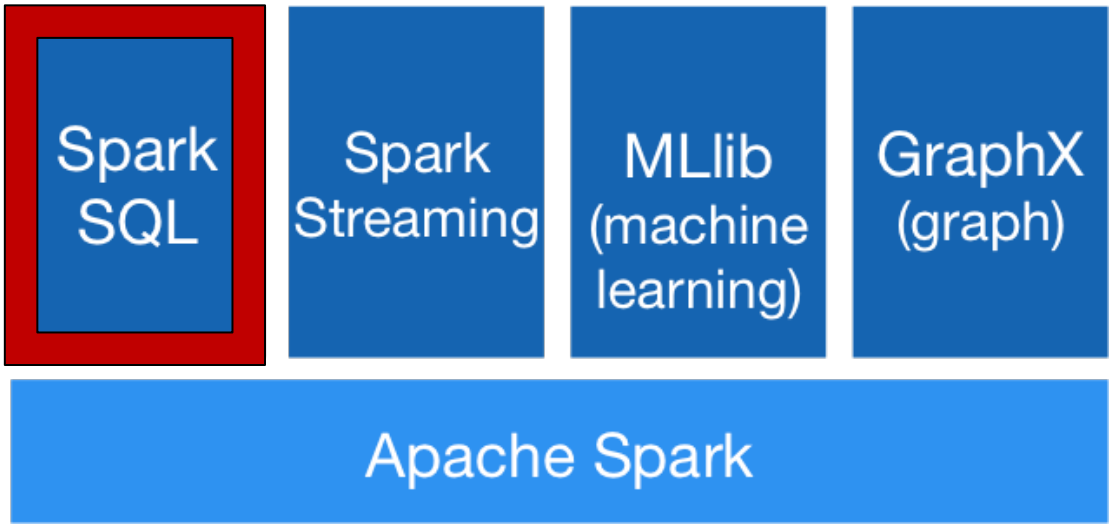

Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

Spark Stack



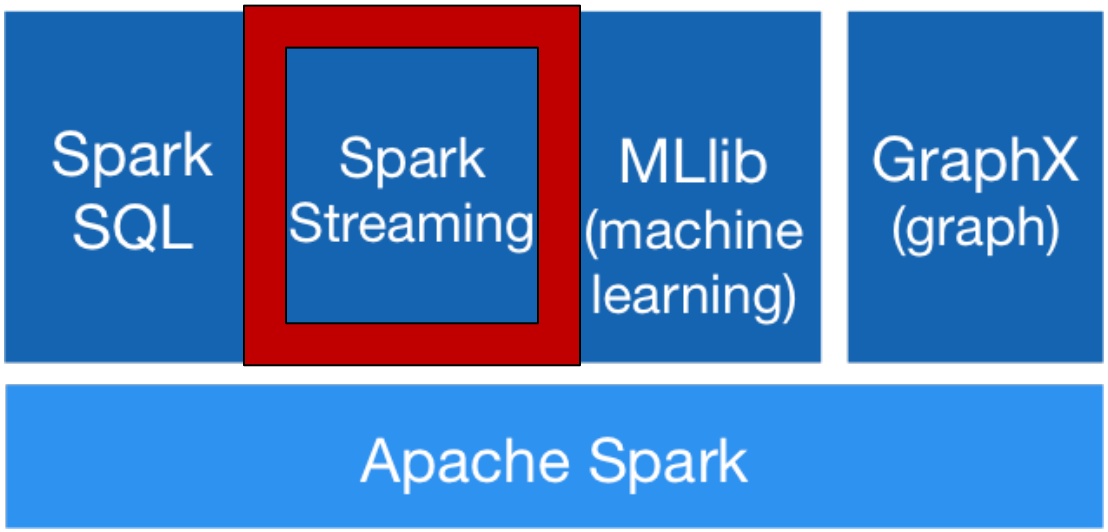
- **Distributed Computing**
  - Distribute tasks across nodes
  - Scheduling
  - Resource management
  - Fault tolerance

Spark Stack



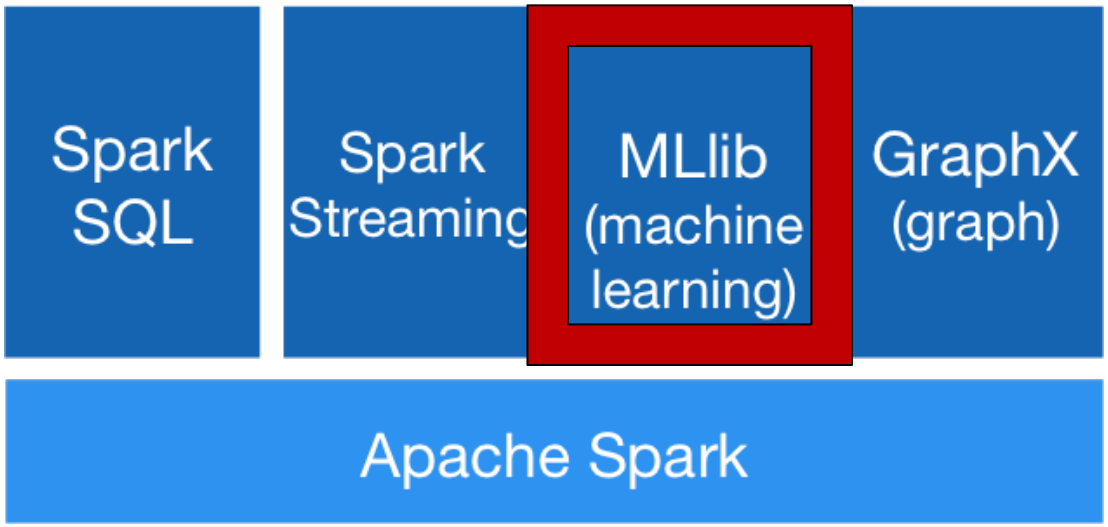
- **Structured Data Processing**
  - Enables querying structured processing
  - Can use SQL and Hive Query Language
  - Can embed SQL queries in Spark code
  - Has APIs for Scala, Java, Python, and R

Spark Stack



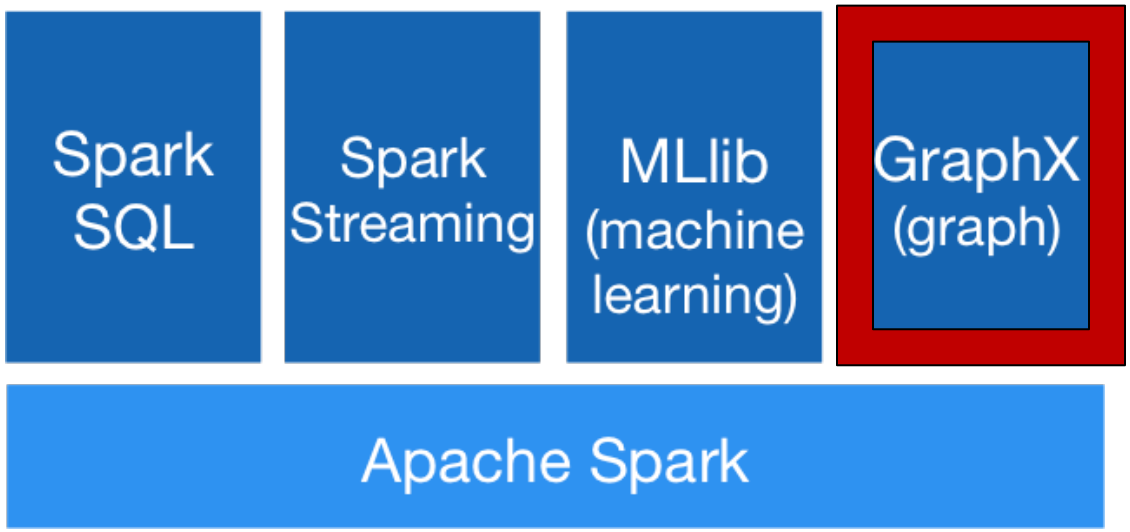
- **Streaming Data Processing**
  - Scalable processing for real-time analytics
  - Data stream is divided into micro-batches of data
  - Has APIs for Scala, Java, and Python

Spark Stack



- **Machine Learning**
  - Scalable machine learning library
  - Distributed implementations of machine learning algorithms and utilities
  - Has APIs for Scala, Java, Python, and R

Spark Stack

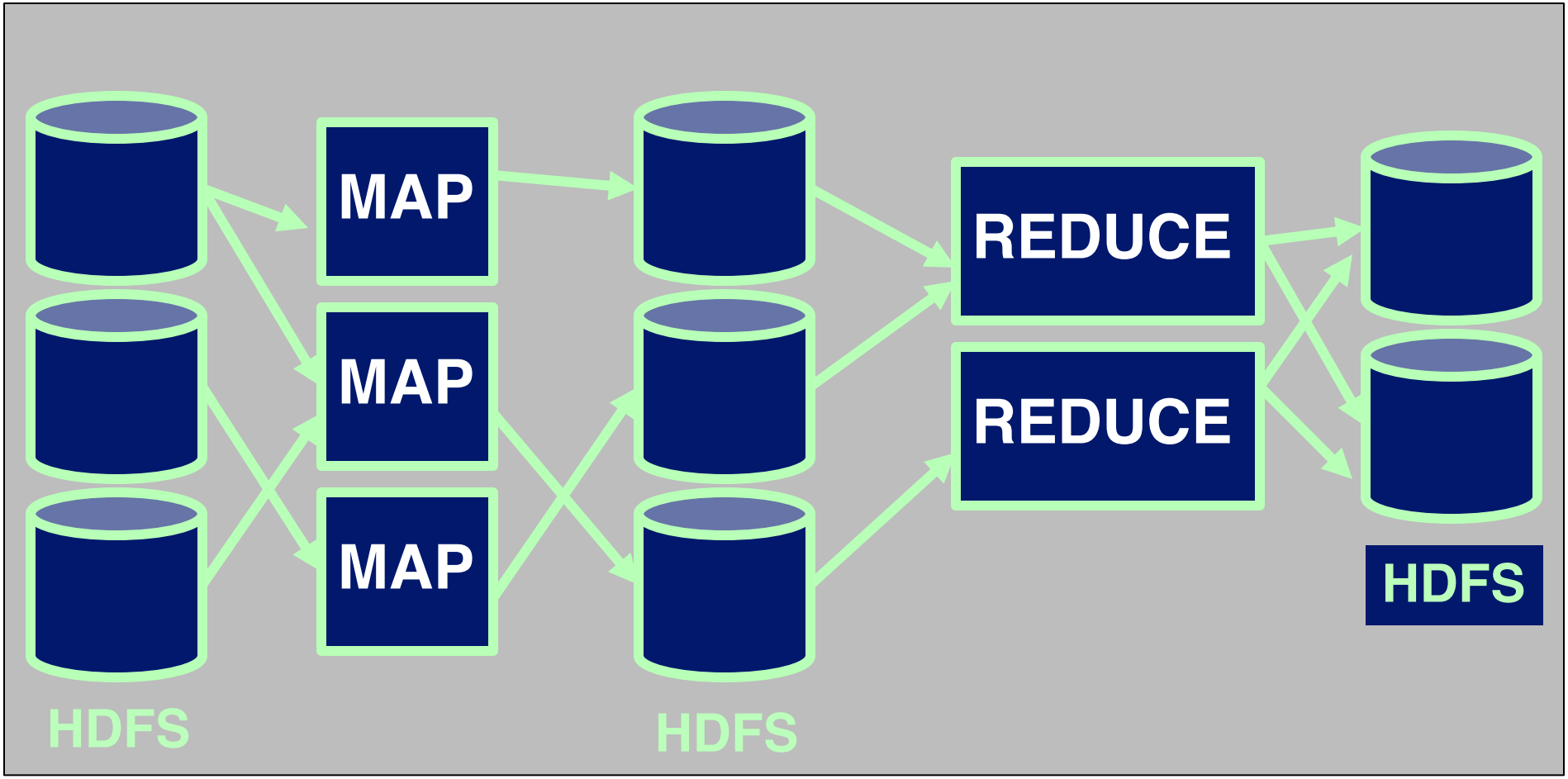


- **Graph Computation**
  - Enables distributed graph processing
  - Special structures for storing vertex and edge information & operations for manipulating graphs
  - Has APIs for Scala and Java (alpha)

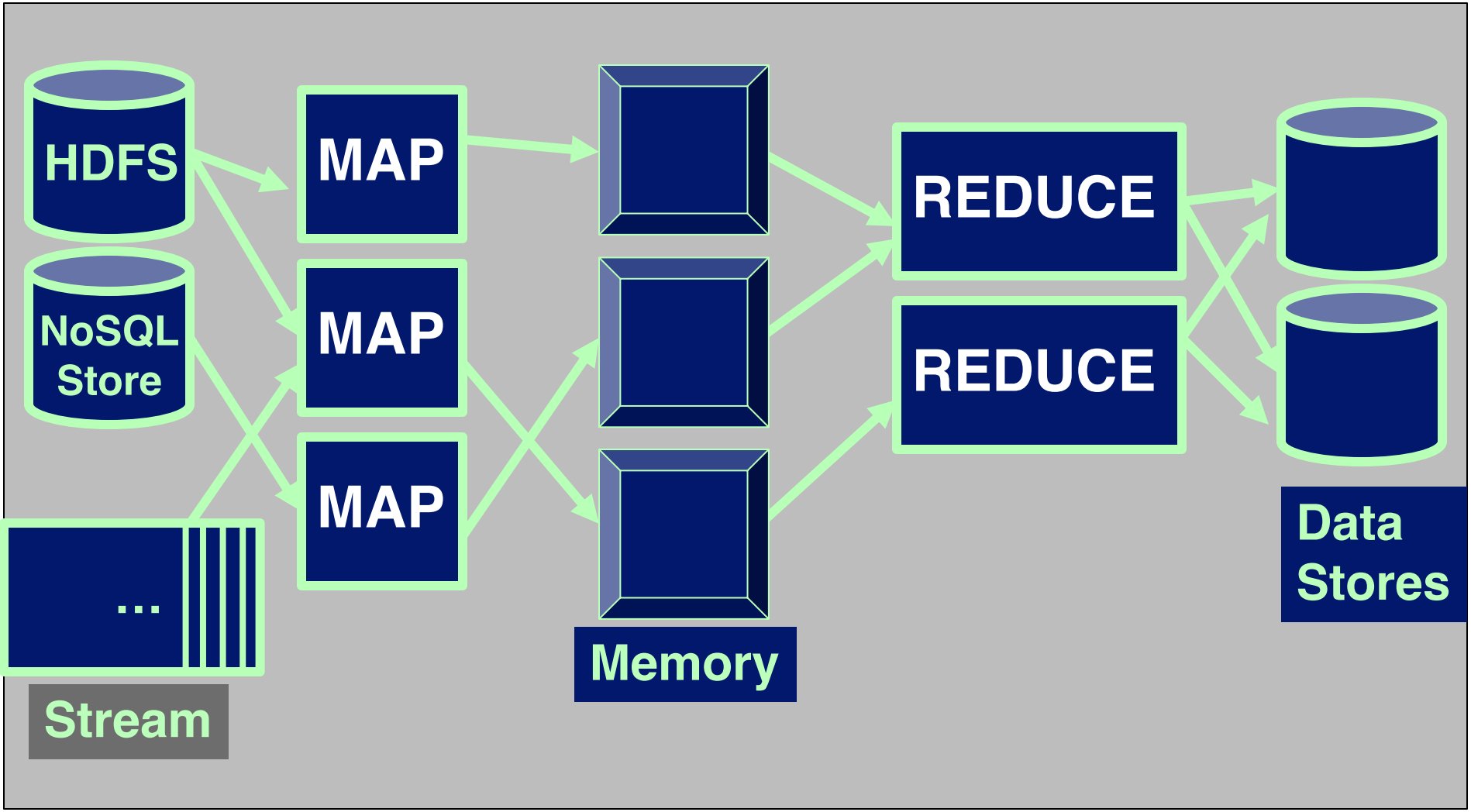
In-Memory Processing

- Provides speed
- Important for iterative operations (e.g., machine learning algorithms) and interactive queries
- Needed for real-time or near real-time analytics
- Spark can run up to 100x faster on some workloads

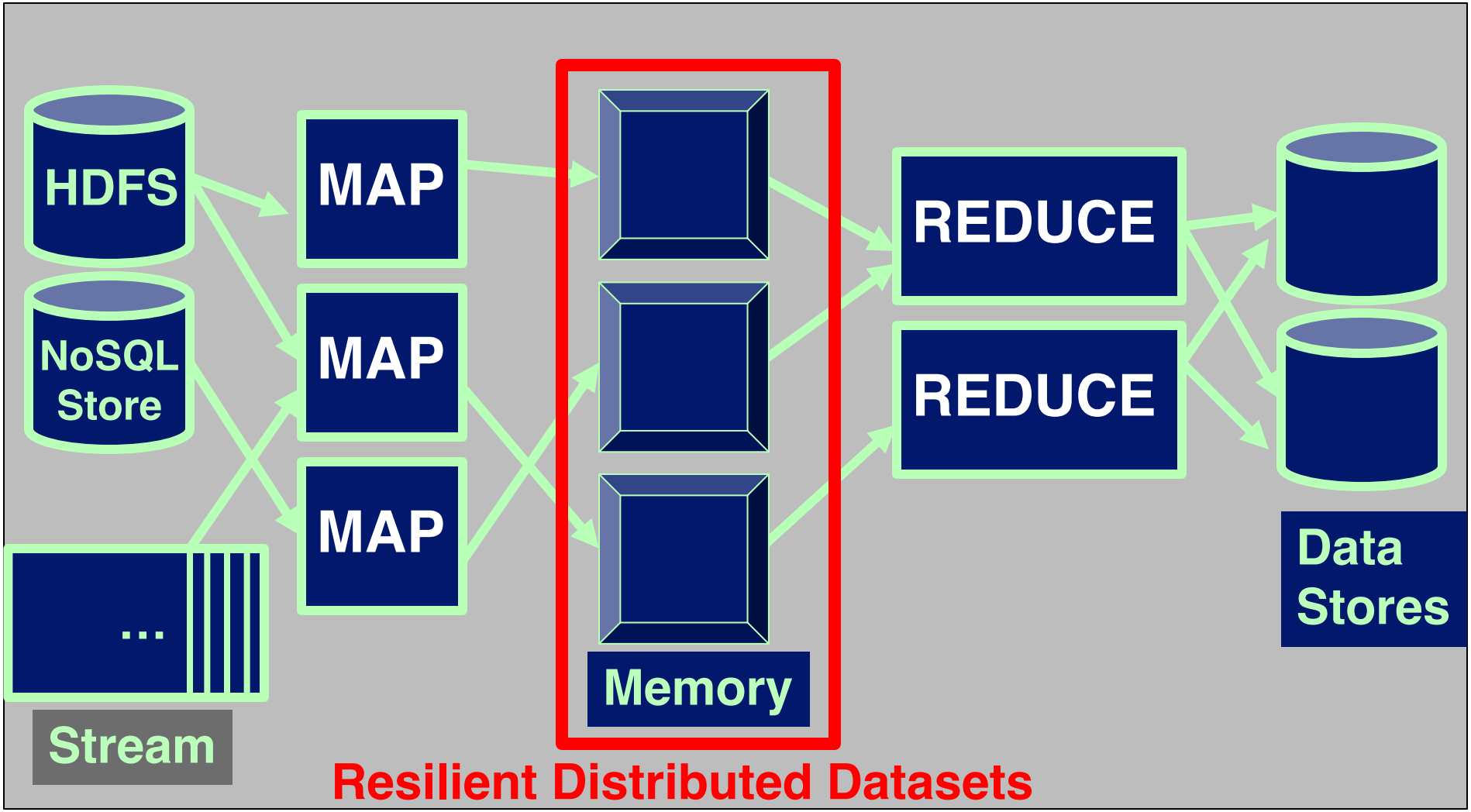
MapReduce



Spark


HDFS
NoSQL Store
...
Stream
MAP
MAP
MAP
Memory
REDUCE
REDUCE
Data Stores

Spark


HDFS
NoSQL Store
...
Stream
MAP
MAP
MAP
Memory
REDUCE
REDUCE
Data Stores
Resilient Distributed Datasets

Resilient Distributed Datasets

Dataset

Data storage created from: HDFS, S3, HBase, JSON, text, Local hierarchy of folders

Or created transforming another RDD

Resilient Distributed Datasets

Distributed

Distributed across the cluster of machines

Divided in partitions, atomic chunks of data

Resilient Distributed Datasets

Resilient

Recover from errors, e.g. node failure, slow processes

Track history of each partition, re-run

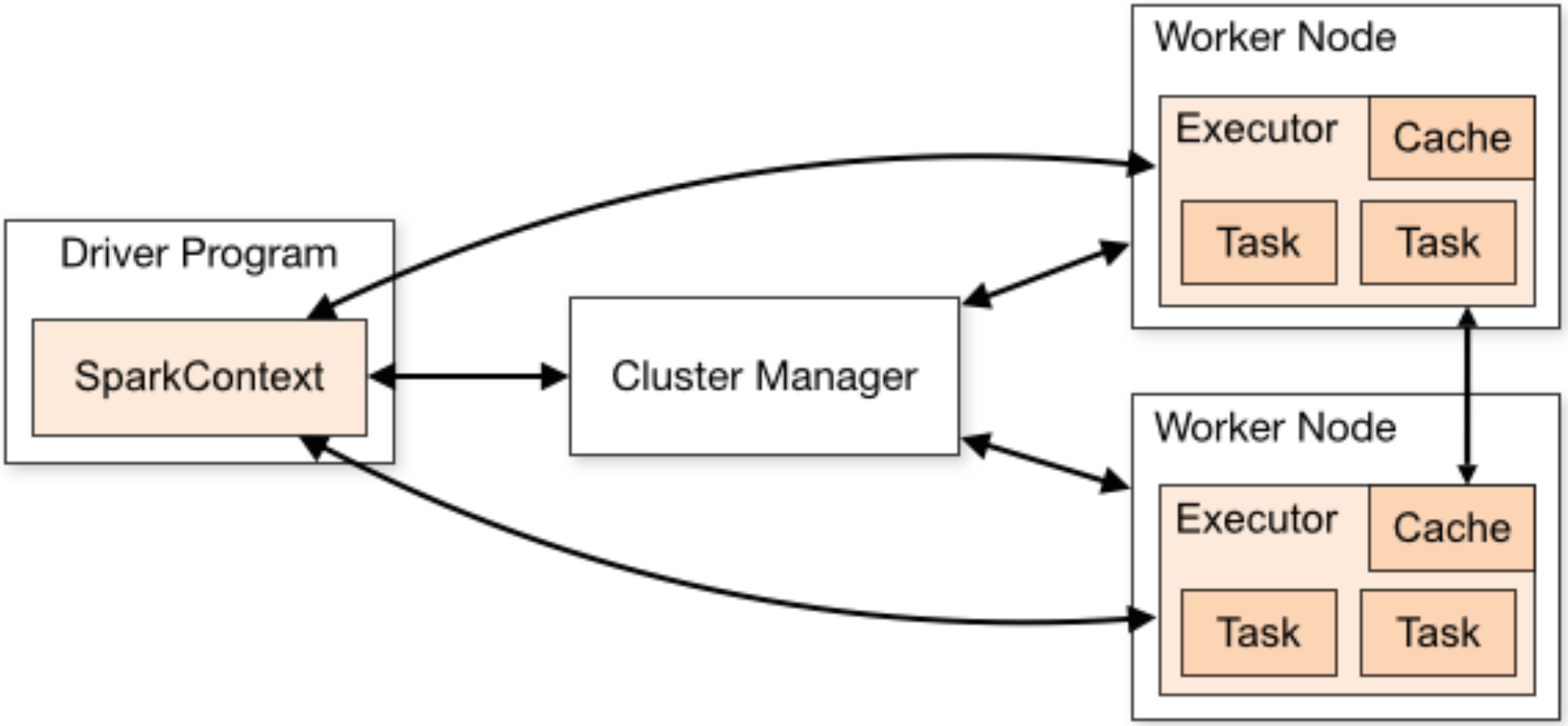
DataFrames & DataSets

DataFrame

DataSet

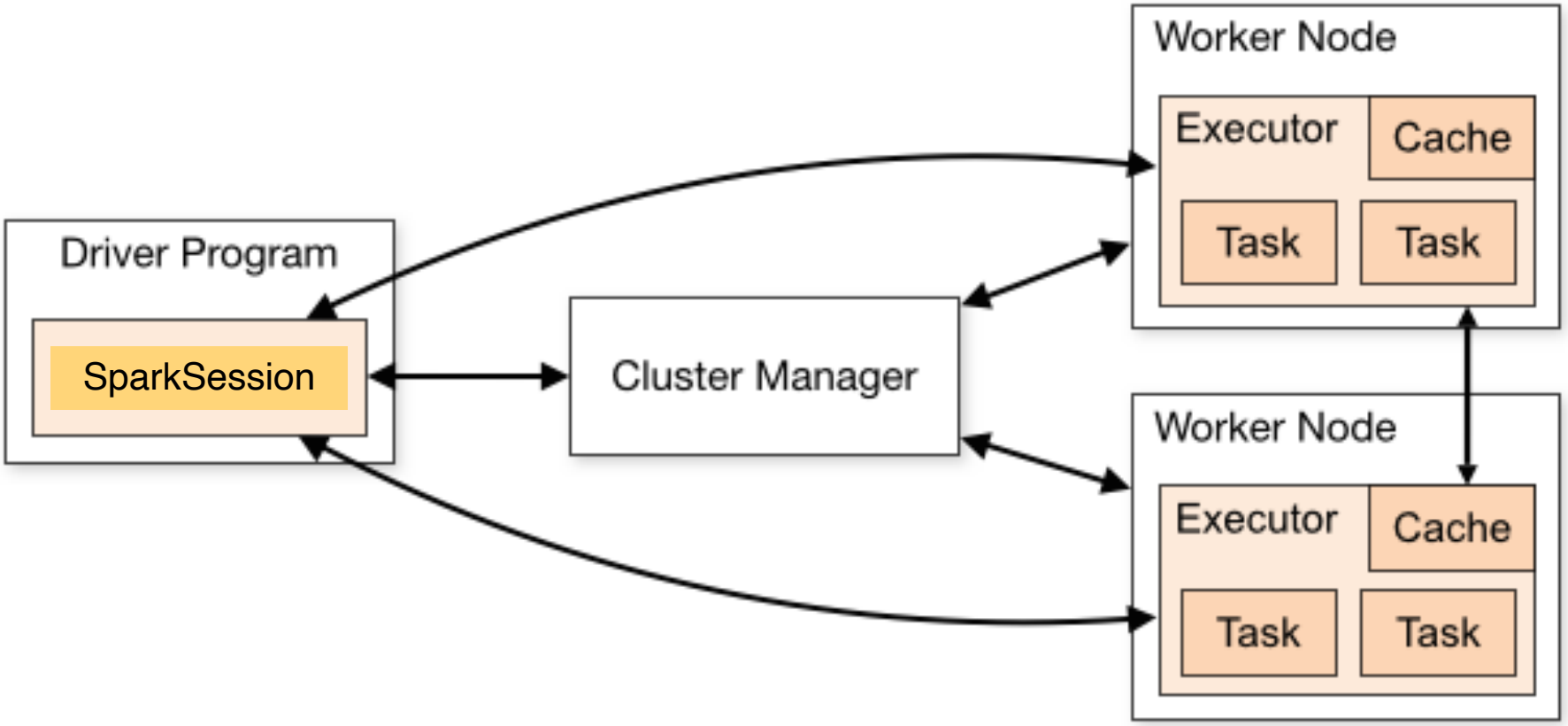
- Extensions to RDDs
- Provide higher-level abstractions, improved performance, better scalability
- Can convert to/from RDDs and use with RDDs

Spark Architecture



http://spark.apache.org/docs/latest/cluster-overview.html

Spark Architecture



http://spark.apache.org/docs/latest/cluster-overview.html

Start Spark Session

Driver Program

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName ("PySpark Example") \
    .config("config.option","config.value") \
    .getOrCreate()
```

Create DataFrame

```
df = spark.read.csv("data.csv"),\
   inferSchema=True,header=True)
```

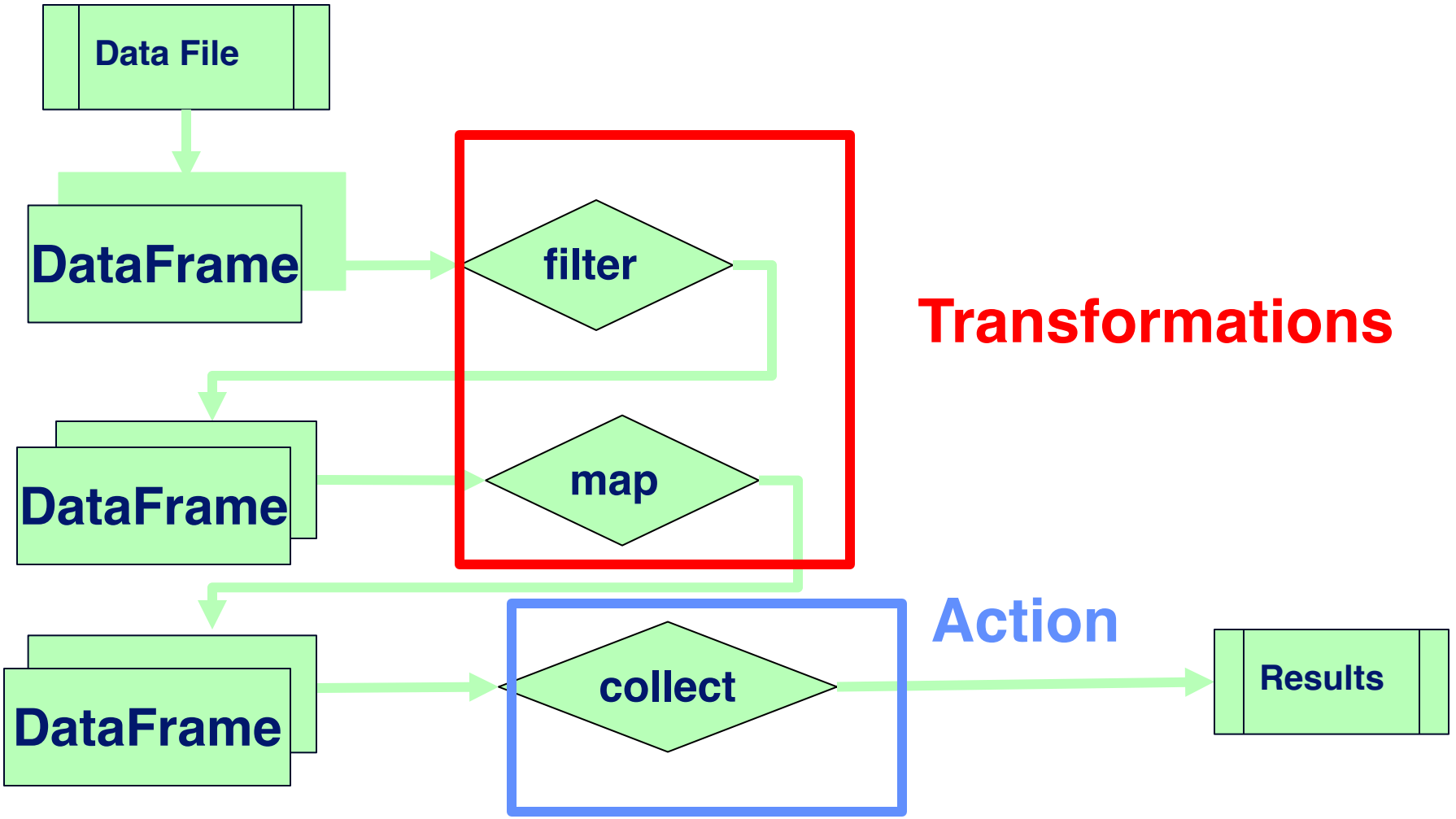
```
df = spark.read.csv
("hdfs:///…/data.csv")
```

```
empl_0 = Row(id='123',name='John')
empl_1 = Row(id='456',name='Mary')
employees = [empl_0, empl_1]
df = spark.createDataFrame(employees)
```

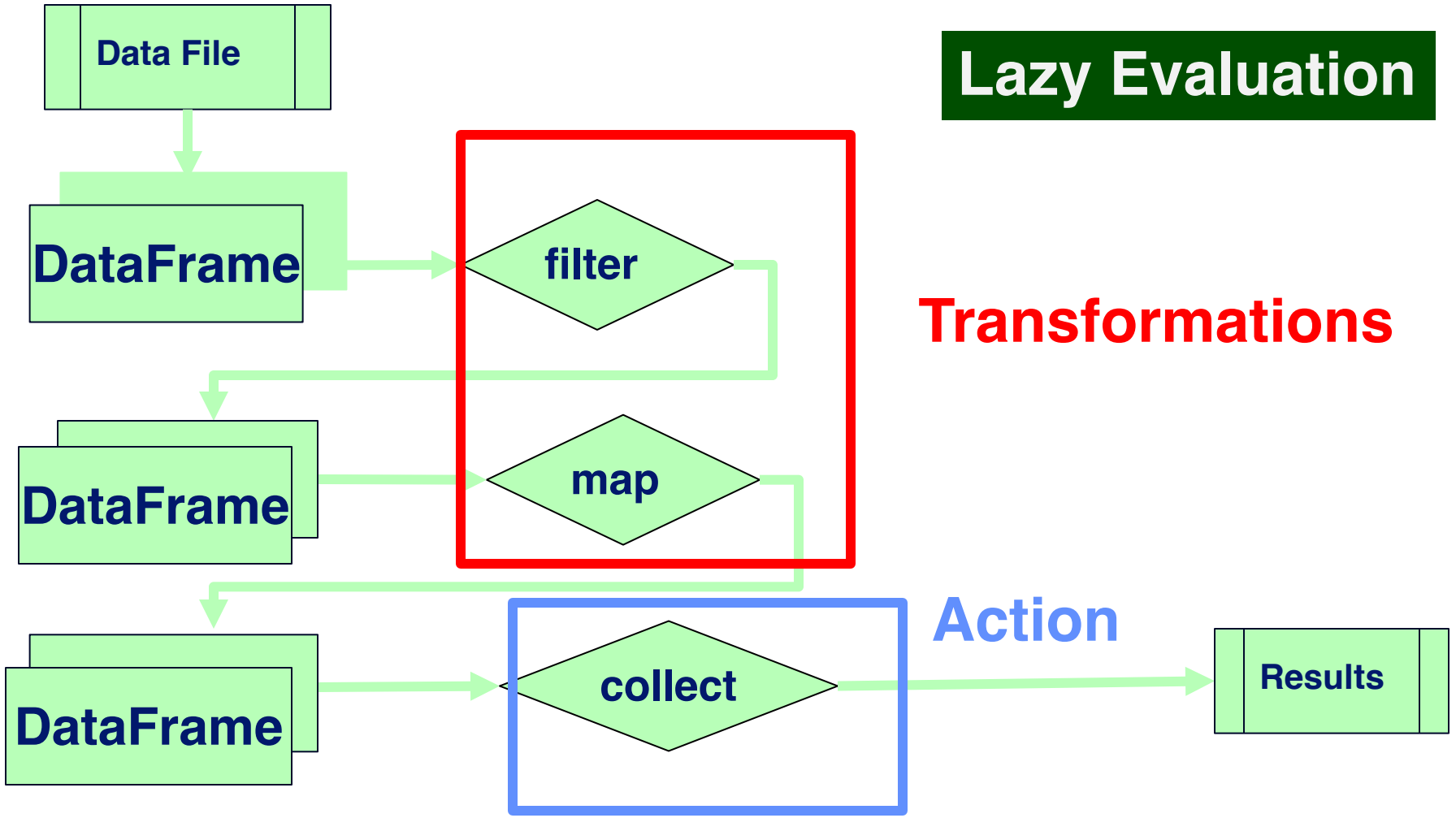
Data Persistence

- **Persist data through caching**
  - Data is stored in memory to avoid re-computing
- **Can specify different storage levels**
  - In memory, on disk, serialized in memory, etc.
- **Examples**
  - df.cache() – MEMORY_ONLY
  - df.persist(MEMORY_ONLY_SER) – Serialized in memory
  - df.unpersist() – Remove from cache

Process Data



Process Data



Lazy Evaluation

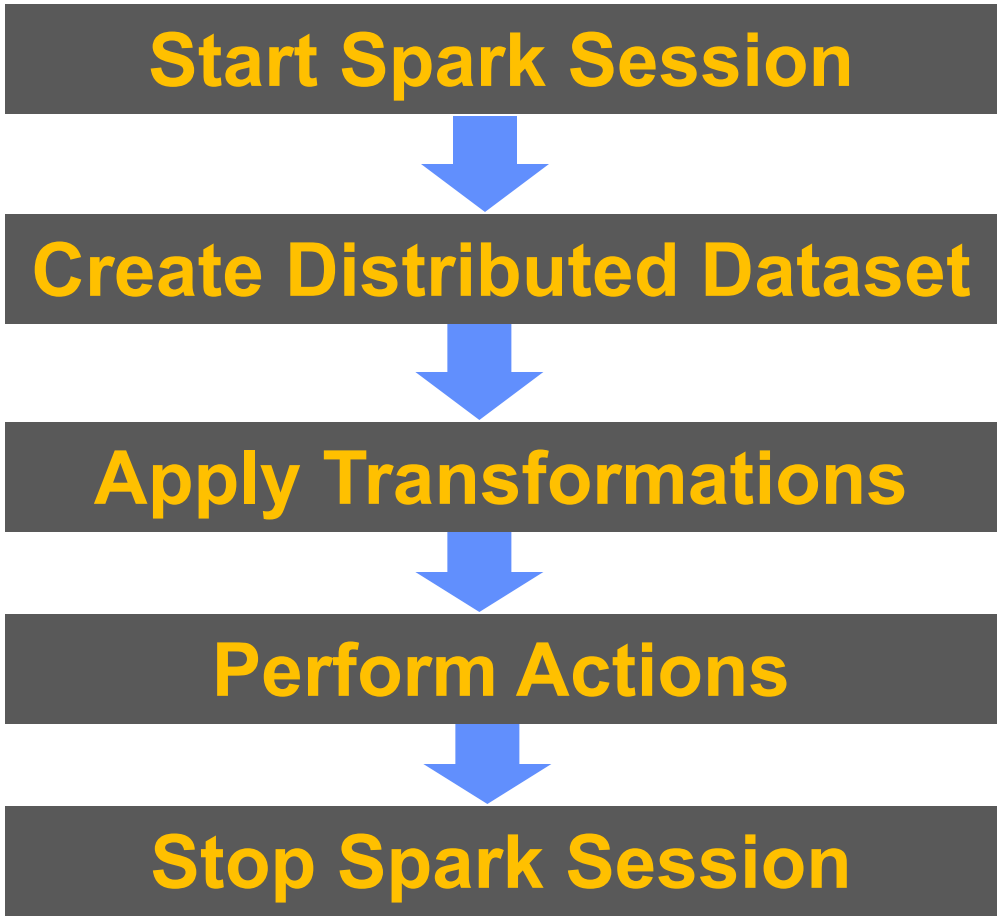
- Transformations not immediately processed
- Plan of transformations is built
- Transformations executed when action is performed.
- Allows for efficient physical plan to be generated

Stop Spark Session

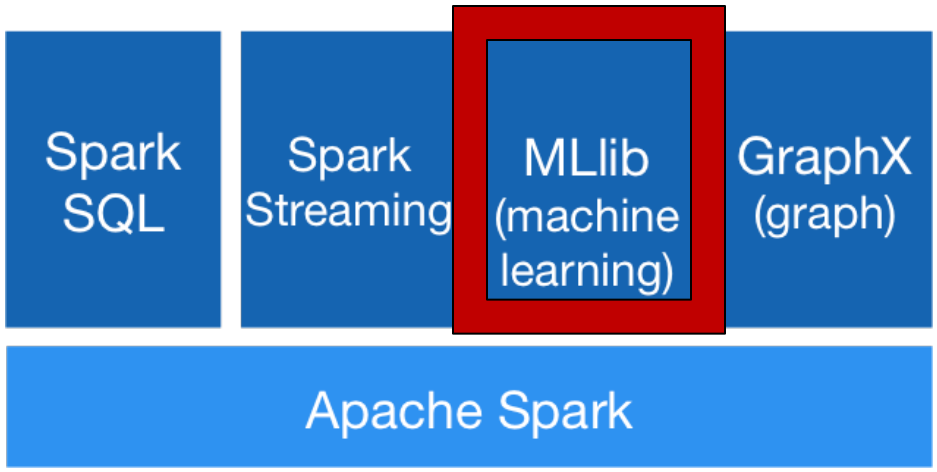
Driver Program

```
spark.stop()
```

Programming in Spark



Spark MLlib



- Machine Learning
  - Classification, regression, clustering, etc.
  - Evaluation metrics
- Statistics
  - Summary statistics, sampling, etc.
- Utilities
  - Dimensionality reduction, transformation, etc.

MILib Example – Statistics

```
from pyspark.sql.functions import rand

# Generate random numbers
df = sqlContext.range(0,10)
                .withColumn("rand1", rand(seed=10))
                .withColumn("rand2", rand(seed=27))

# Show summary statistics
df.describe().show()

# Compute correlation
df.stat.corr("rand1","rand2")
```

MLlib Example – Clustering

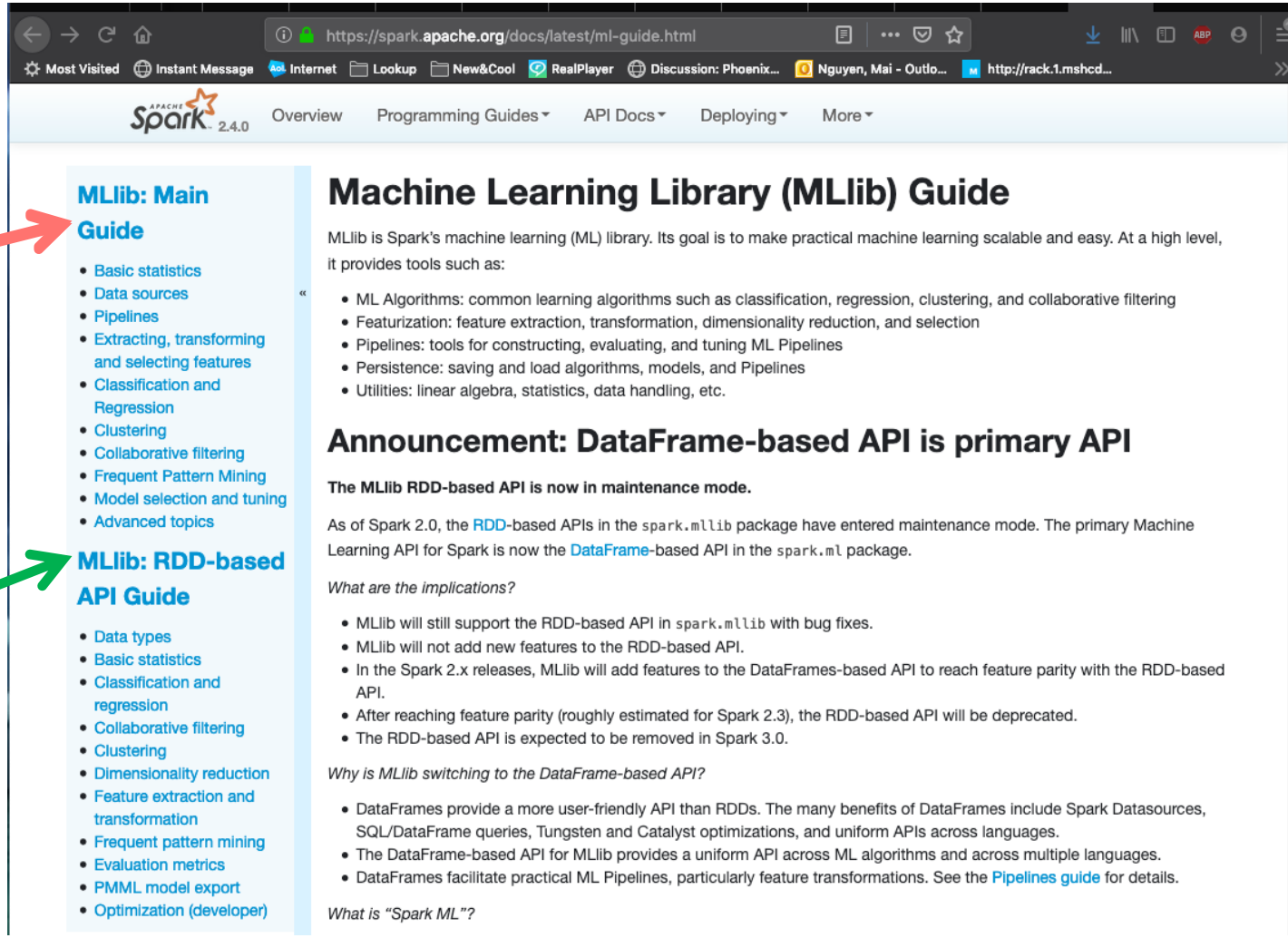
```
from pyspark.ml.clustering import KMeans

# Read and parse data
data = spark.read.csv("data.csv",inferSchema="true",
                      header="true")

# k-means model for clustering
kmeans = Kmeans().setK(3).setSeed(123)
model = kmeans.fit (data)
for center in model.clusterCenters()
   print (center)
```

MLlib vs. ML Libraries

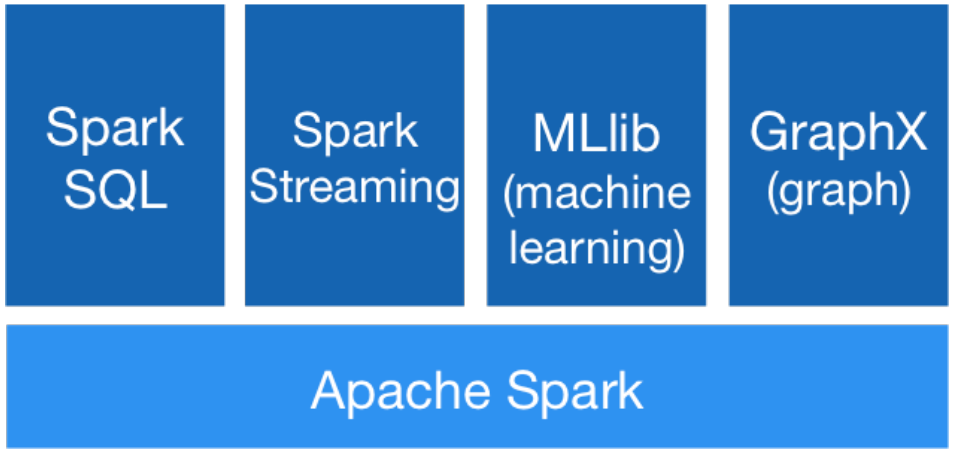
mllib vs. ml

ml
(DataFrame-based)

mllib
(RDD-based)



Spark



- Spark engine provides distributed computing
- Libraries support multiple analytics applications and workloads
- RDD/DF/DS provide data parallelism & fault-tolerance
- MLlib provides scalable machine learning

Spark Resources

- Spark
  - https://spark.apache.org/
- MLlib
  - https://spark.apache.org/mllib/
- Mastering Apache Spark
  - https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/