

Machine Learning with Spark

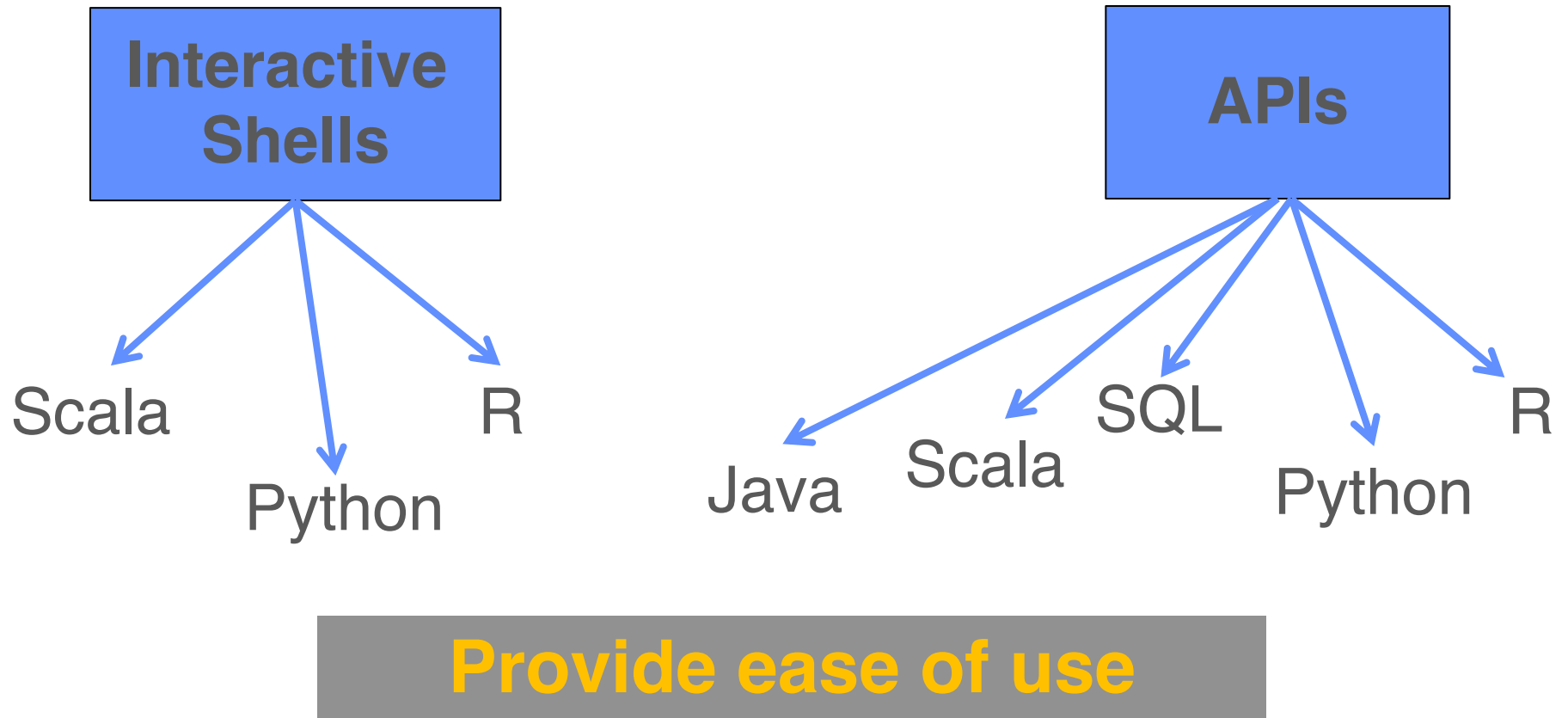
Mai H. Nguyen, Ph.D.

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



Spark Stack



Spark
SQL

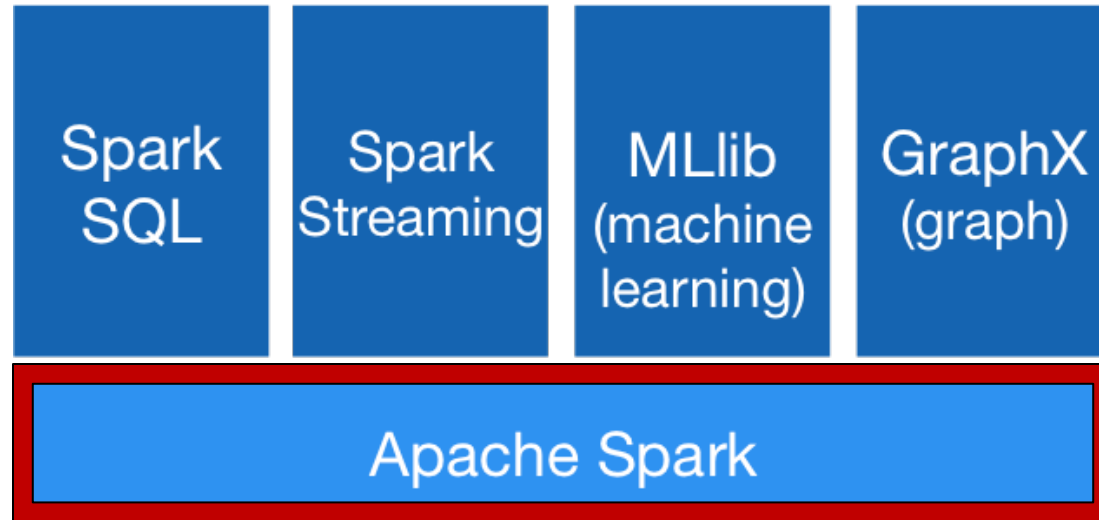
Spark
Streaming

MLlib
(machine
learning)

GraphX
(graph)

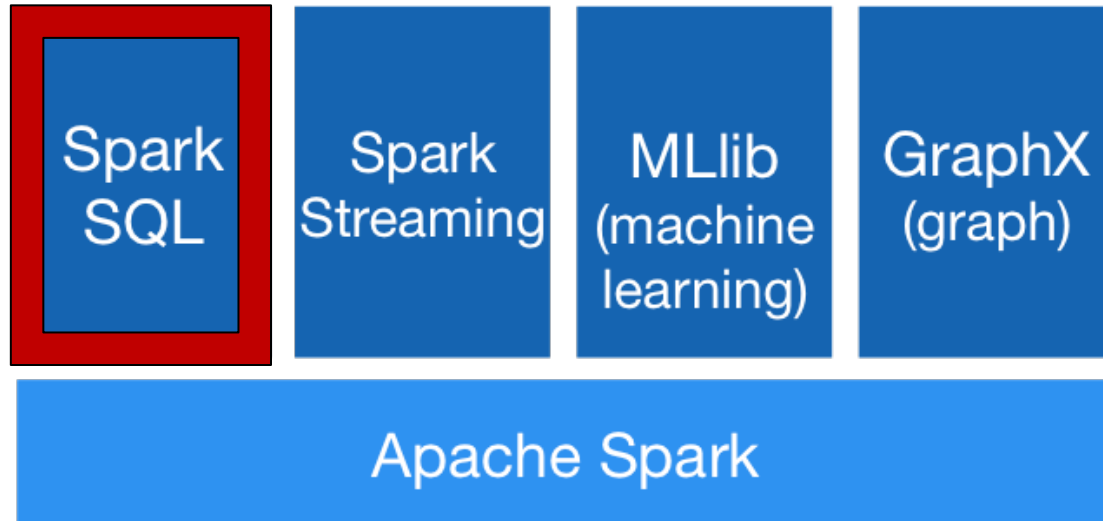
Apache Spark

Spark Stack



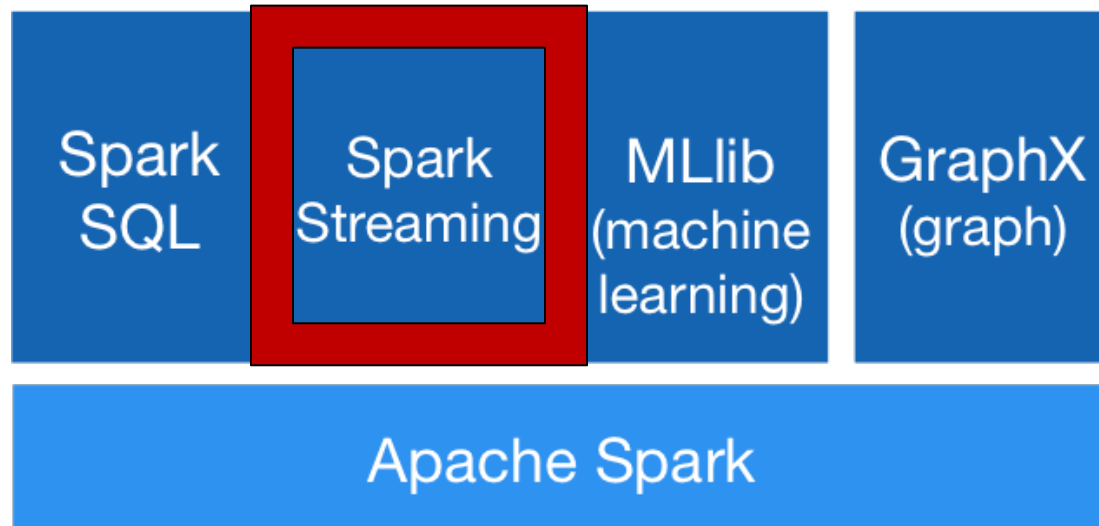
- **Distributed Computing**
 - Distribute tasks across nodes
 - Scheduling
 - Resource management
 - Fault tolerance

Spark Stack



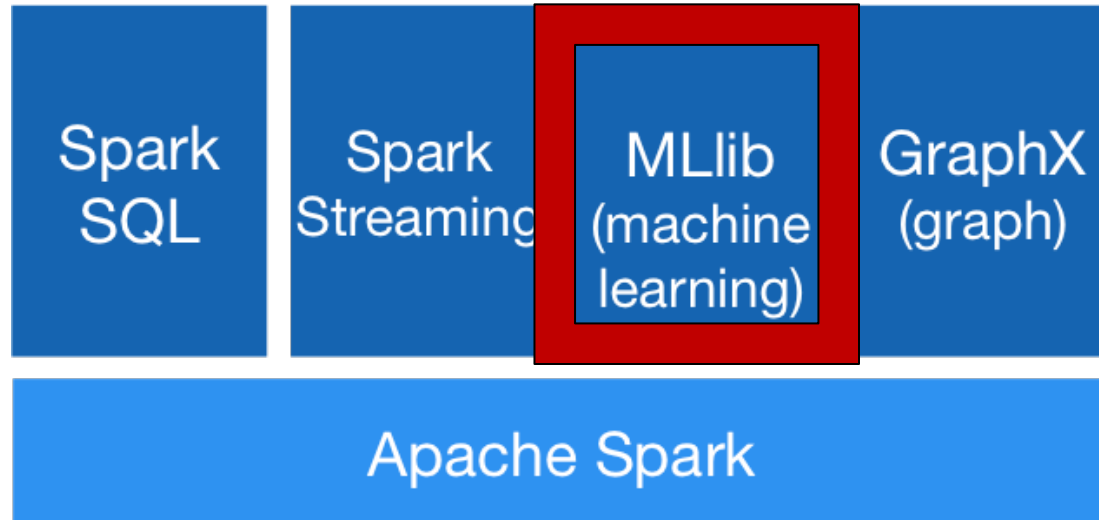
- **Structured Data Processing**
 - Enables querying structured processing
 - Can use SQL and Hive Query Language
 - Can embed SQL queries in Spark code
 - Has APIs for Scala, Java, Python, and R

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

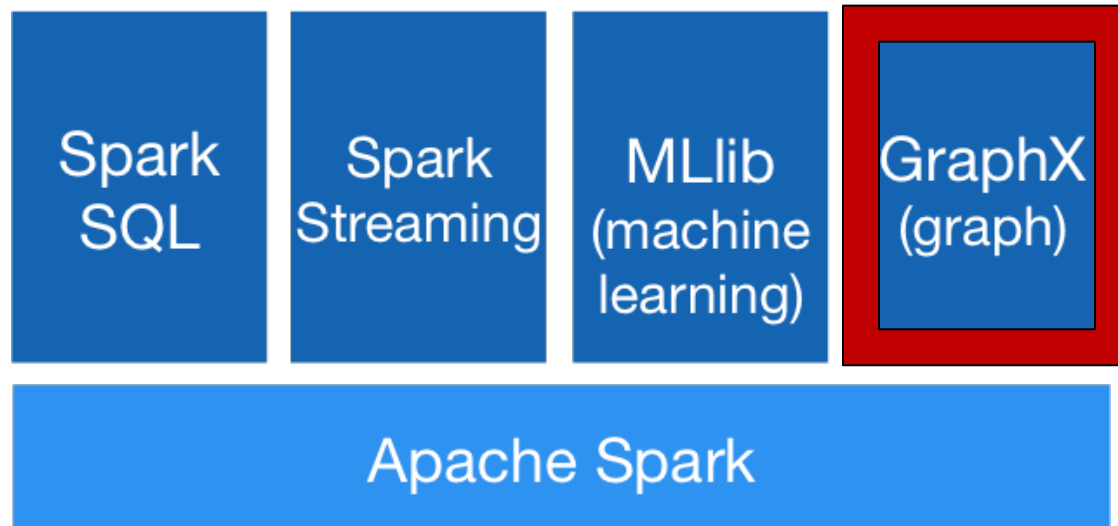
Spark Stack



- **Machine Learning**

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

Spark Stack



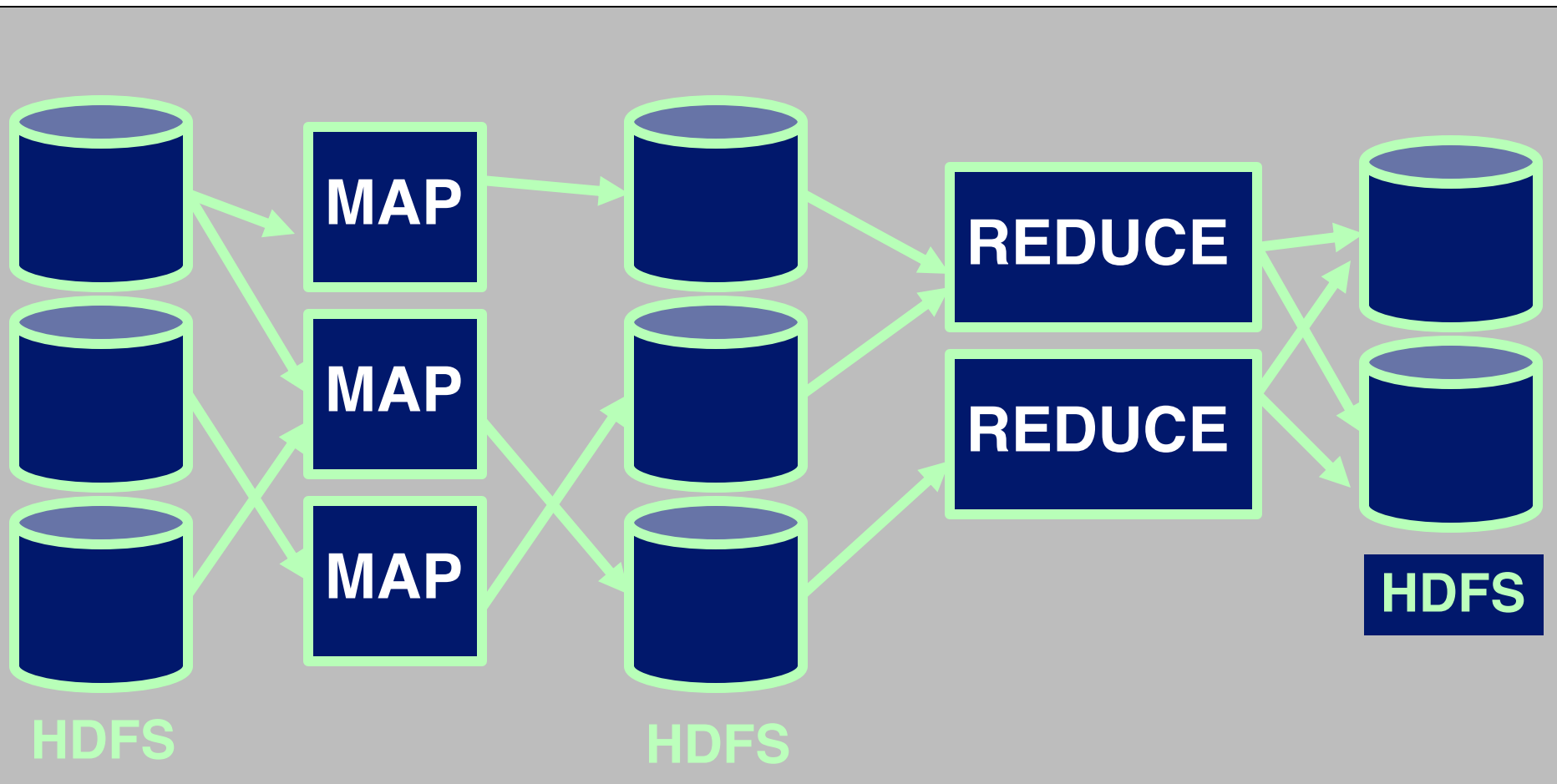
- **Graph Computation**

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

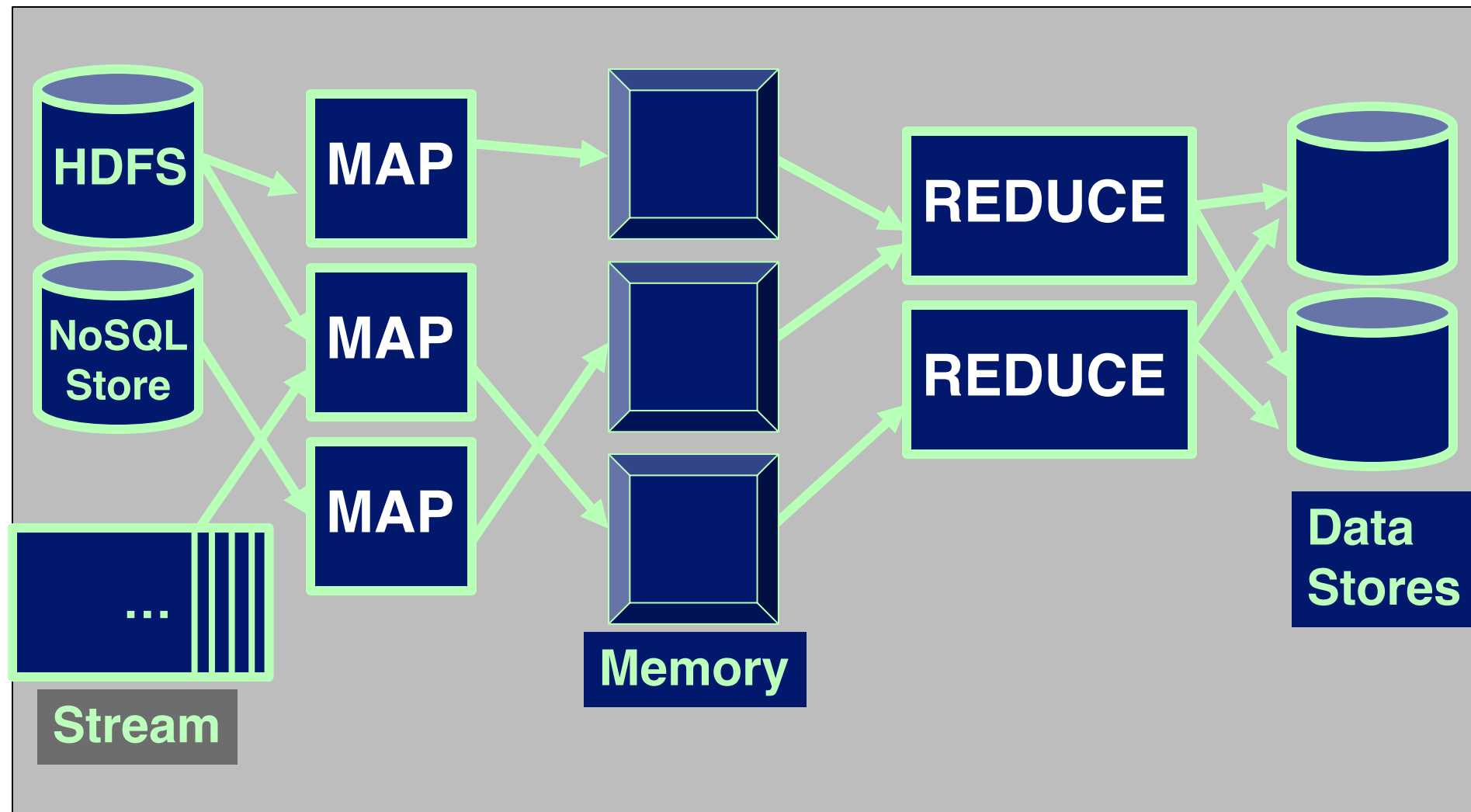
In-Memory Processing

- 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

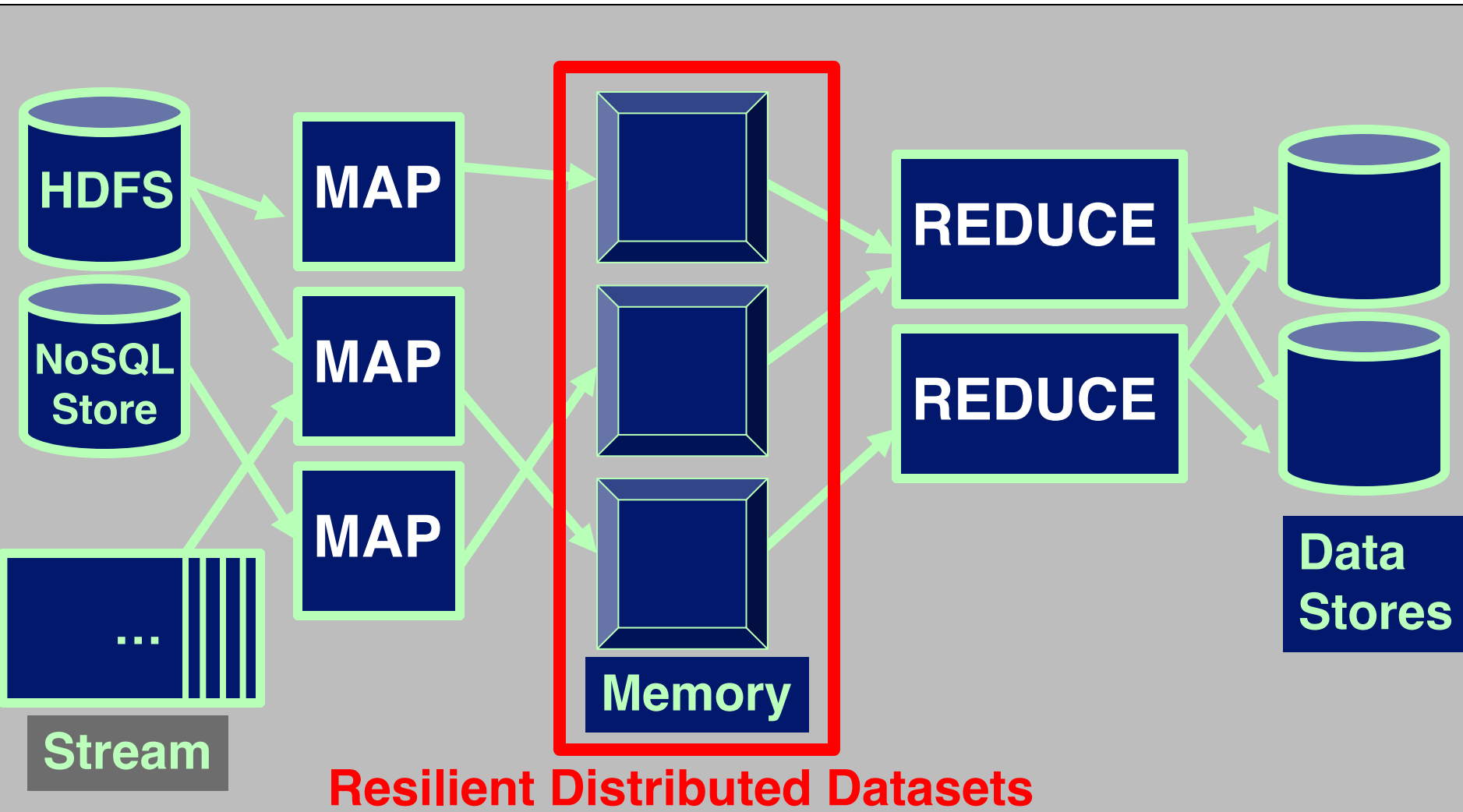
MapReduce



Spark



Spark



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*

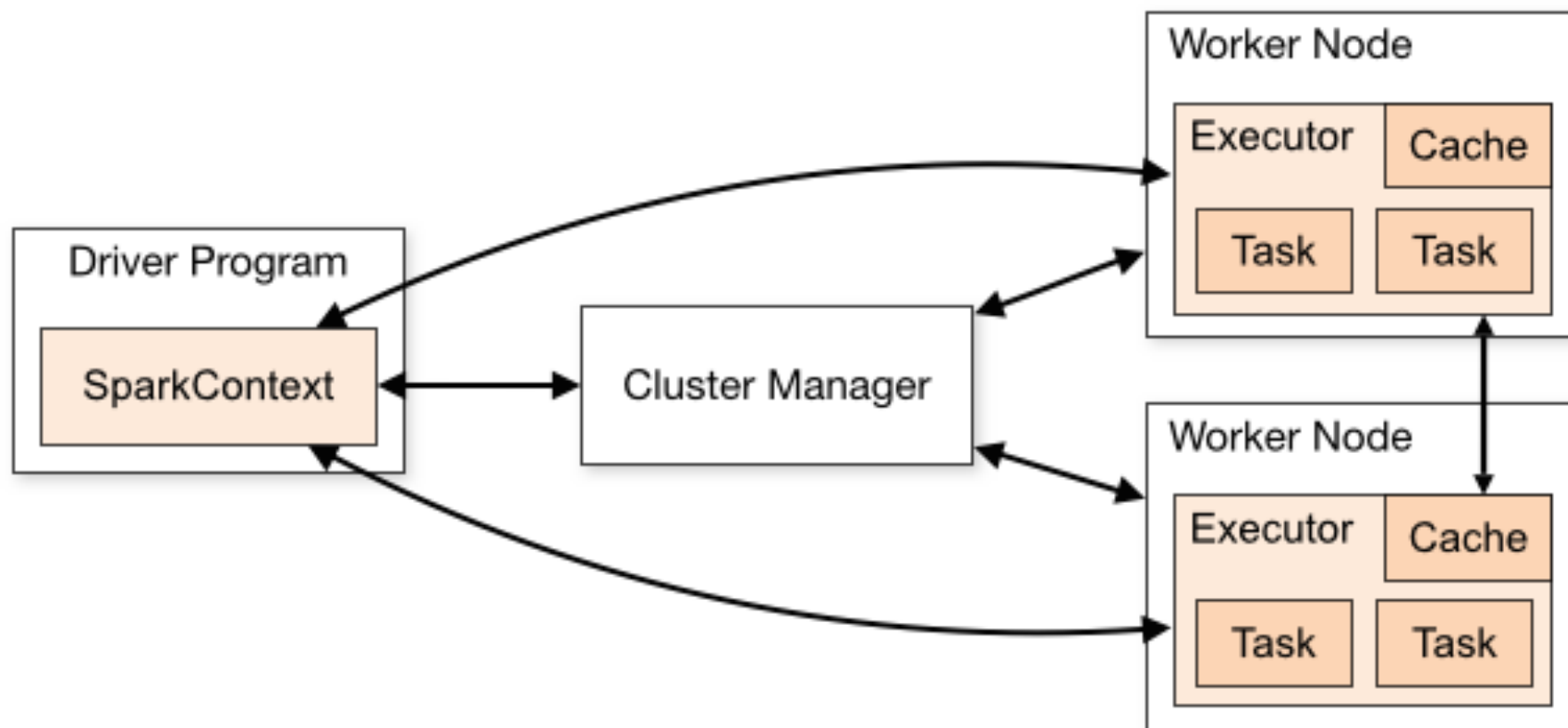
DataFrames & DataSets

DataFrame

DataSet

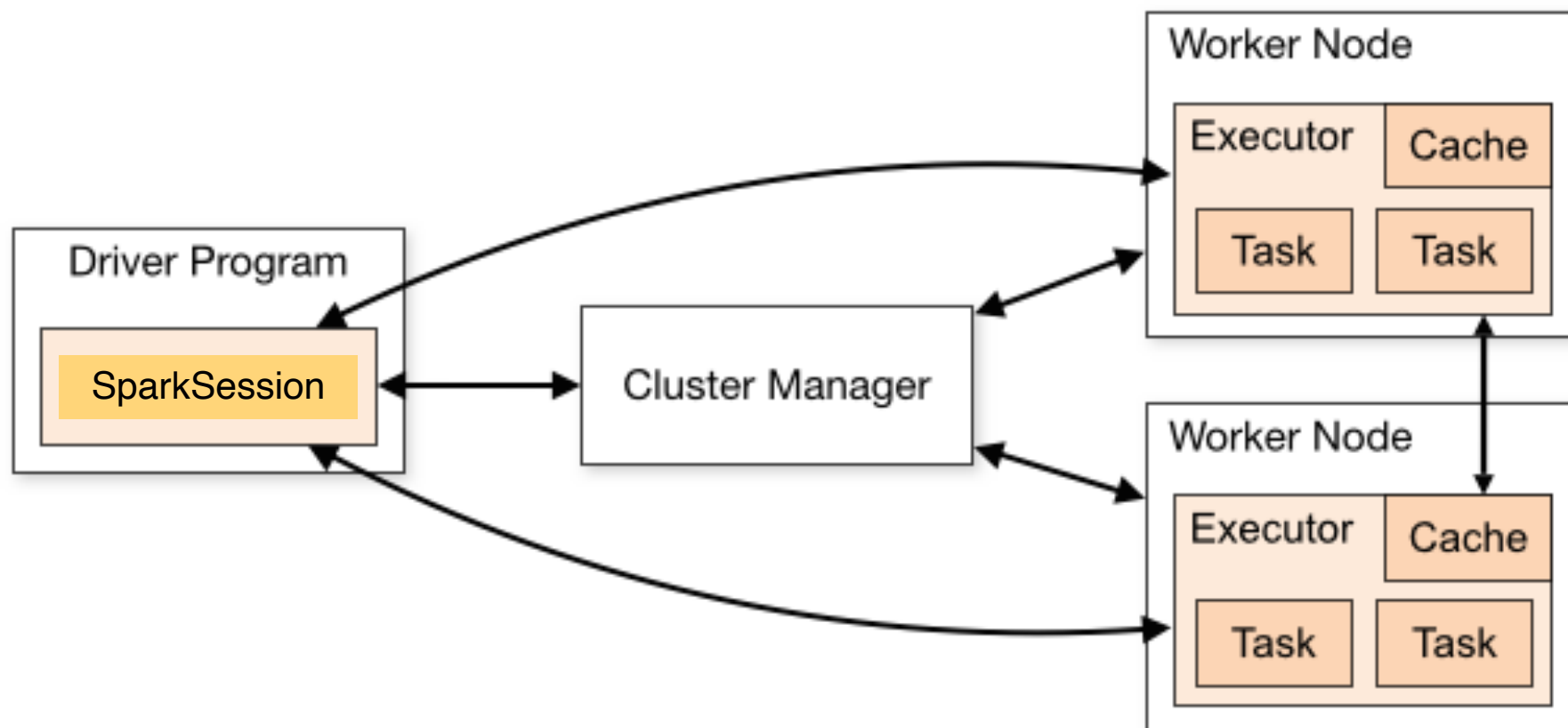
- Extensions to RDDs
- Provide higher-level abstractions, improved performance, better scalability
- Can convert to/from RDDs and use with RDDs

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)
```

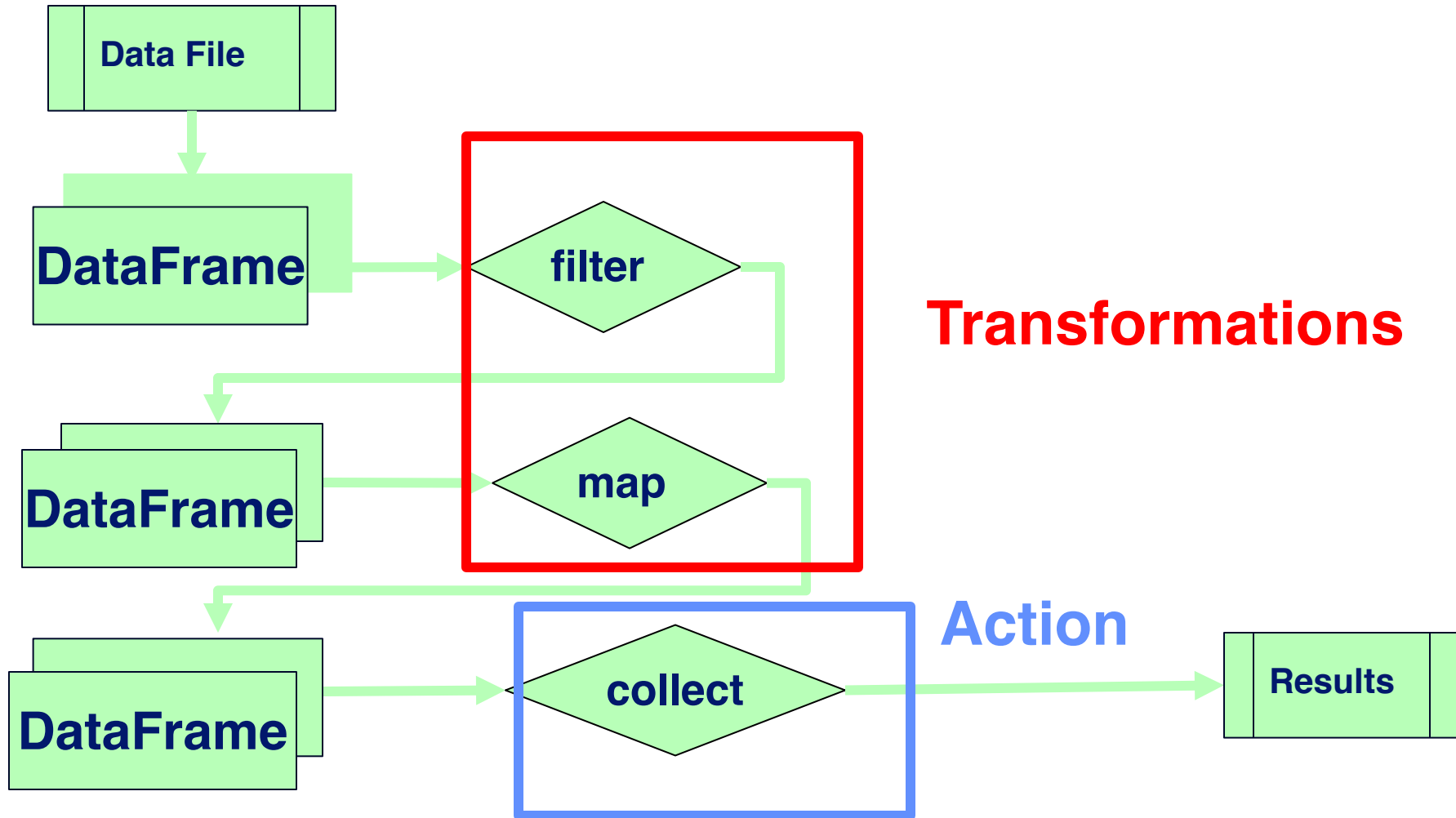
```
df = spark.read.csv\n("hdfs:///.../data.csv")
```

```
empl_0 = Row(id='123',name='John')\nempl_1 = Row(id='456',name='Mary')\nemployees = [empl_0, empl_1]\ndf = spark.createDataFrame(employees)
```

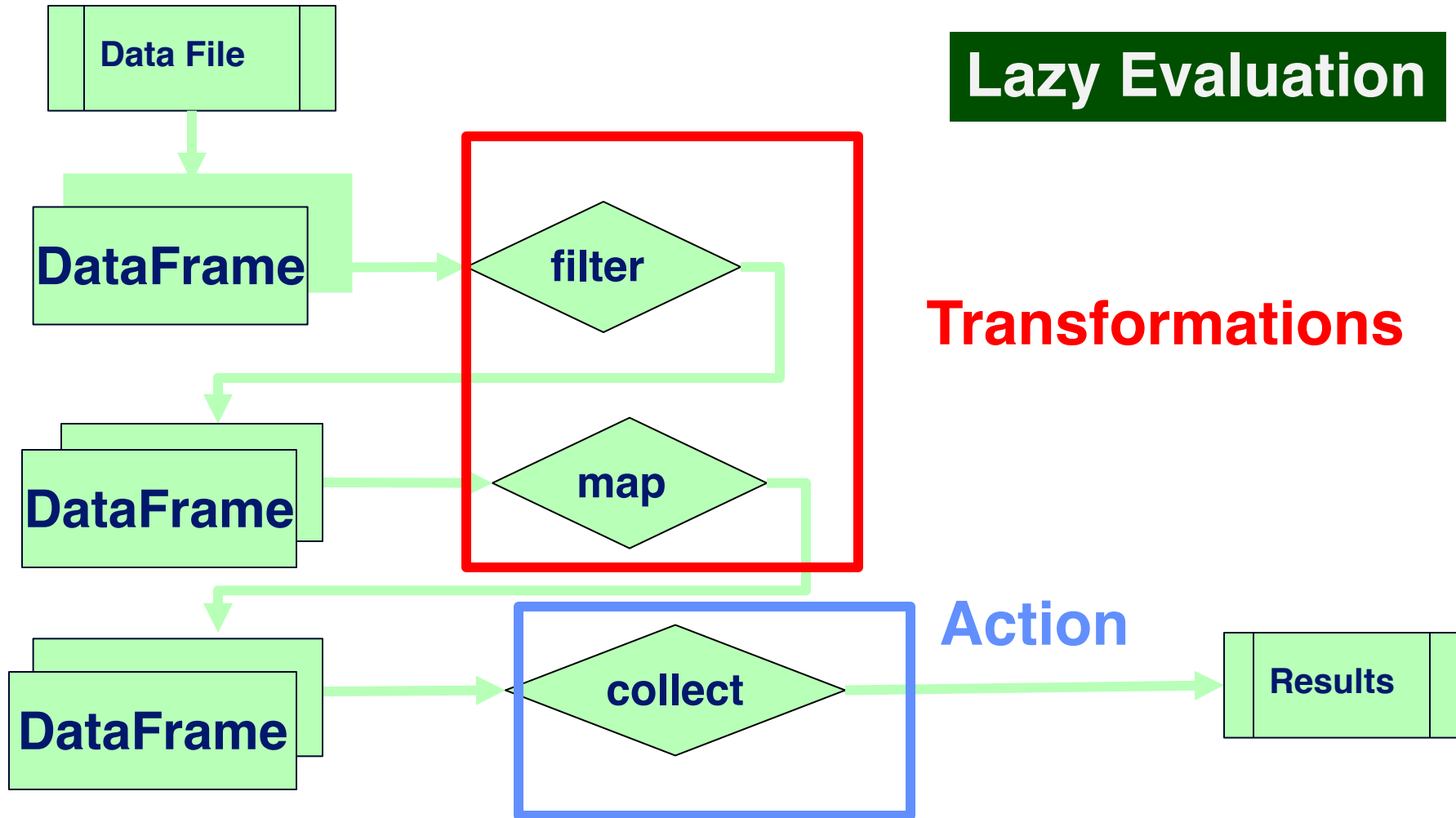
Data Persistence

- **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

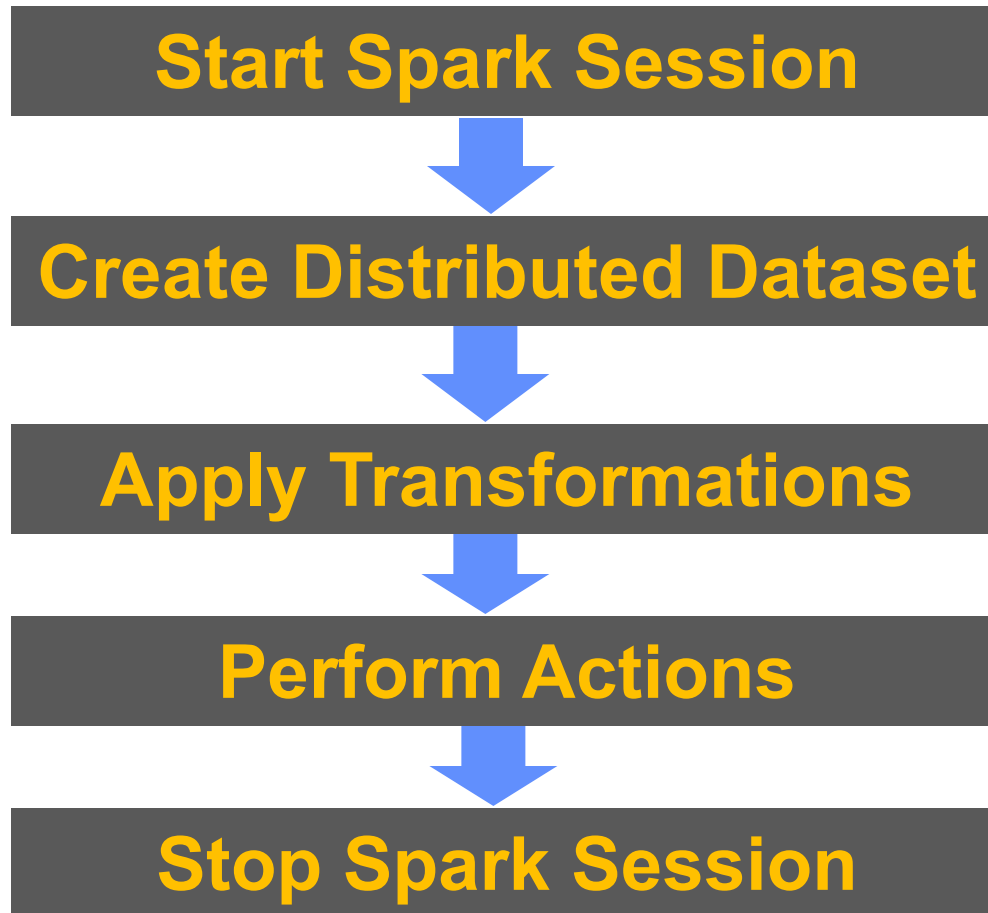
- Transformations not immediately processed
- Plan of transformations is built
- Transformations executed when action is performed.
- Allows for efficient physical plan to be generated

Stop Spark Session

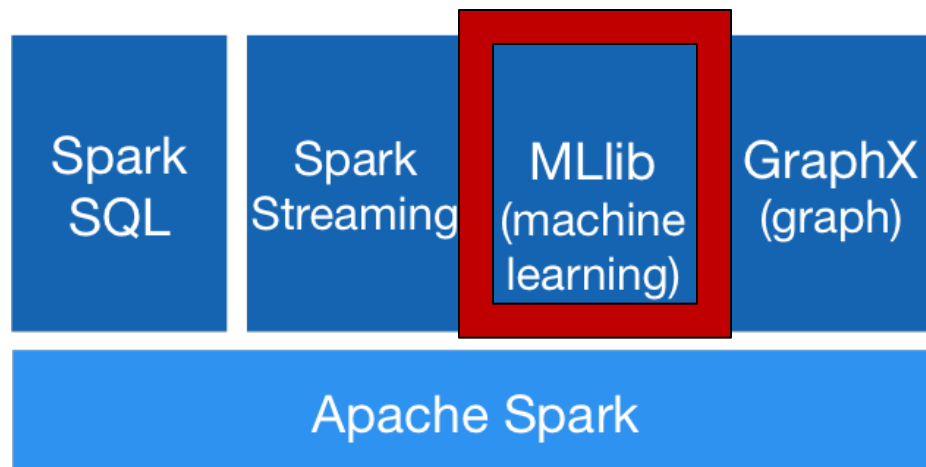
*Driver
Program*

```
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.

MLlib 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)

The screenshot shows the Apache Spark MLlib Guide page. A red arrow points from the 'ml' text to the 'MLlib: Main Guide' section. A green arrow points from the 'mllib' text to the 'MLlib: RDD-based API Guide' section.

MLlib: Main Guide

- Basic statistics
- Data sources
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

MLlib: RDD-based API Guide

- Data types
- Basic statistics
- Classification and regression
- Collaborative filtering
- Clustering
- Dimensionality reduction
- Feature extraction and transformation
- Frequent pattern mining
- Evaluation metrics
- PMML model export
- Optimization (developer)

Machine Learning Library (MLlib) Guide

MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. At a high level, it provides tools such as:

- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featureization: feature extraction, transformation, dimensionality reduction, and selection
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

Announcement: DataFrame-based API is primary API

The MLlib RDD-based API is now in maintenance mode.

As of Spark 2.0, the [RDD](#)-based APIs in the `spark.mllib` package have entered maintenance mode. The primary Machine Learning API for Spark is now the [DataFrame](#)-based API in the `spark.ml` package.

What are the implications?

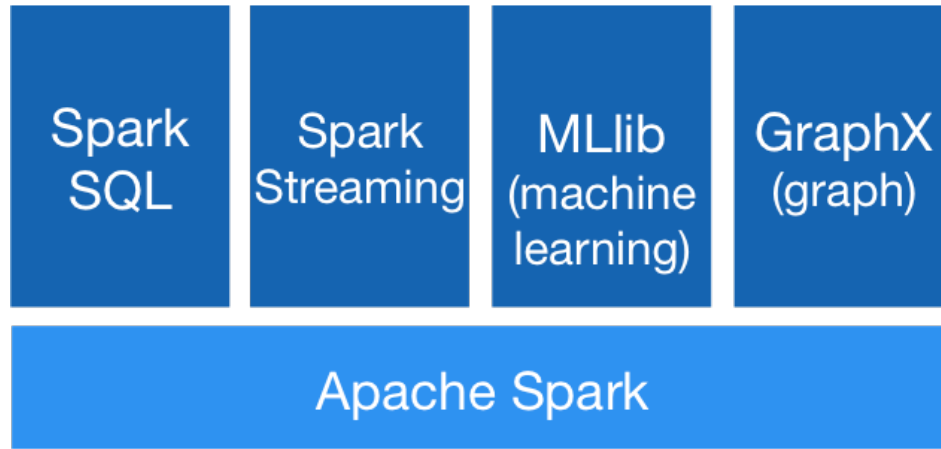
- MLlib will still support the RDD-based API in `spark.mllib` with bug fixes.
- MLlib will not add new features to the RDD-based API.
- In the Spark 2.x releases, MLlib will add features to the DataFrames-based API to reach feature parity with the RDD-based API.
- After reaching feature parity (roughly estimated for Spark 2.3), the RDD-based API will be deprecated.
- The RDD-based API is expected to be removed in Spark 3.0.

Why is MLlib switching to the DataFrame-based API?

- DataFrames provide a more user-friendly API than RDDs. The many benefits of DataFrames include Spark Datasources, SQL/DataFrame queries, Tungsten and Catalyst optimizations, and uniform APIs across languages.
- The DataFrame-based API for MLlib provides a uniform ML algorithms and across multiple languages.
- DataFrames facilitate practical ML Pipelines, particularly feature transformations. See the [Pipelines guide](#) for details.

What is "Spark ML"?

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/>