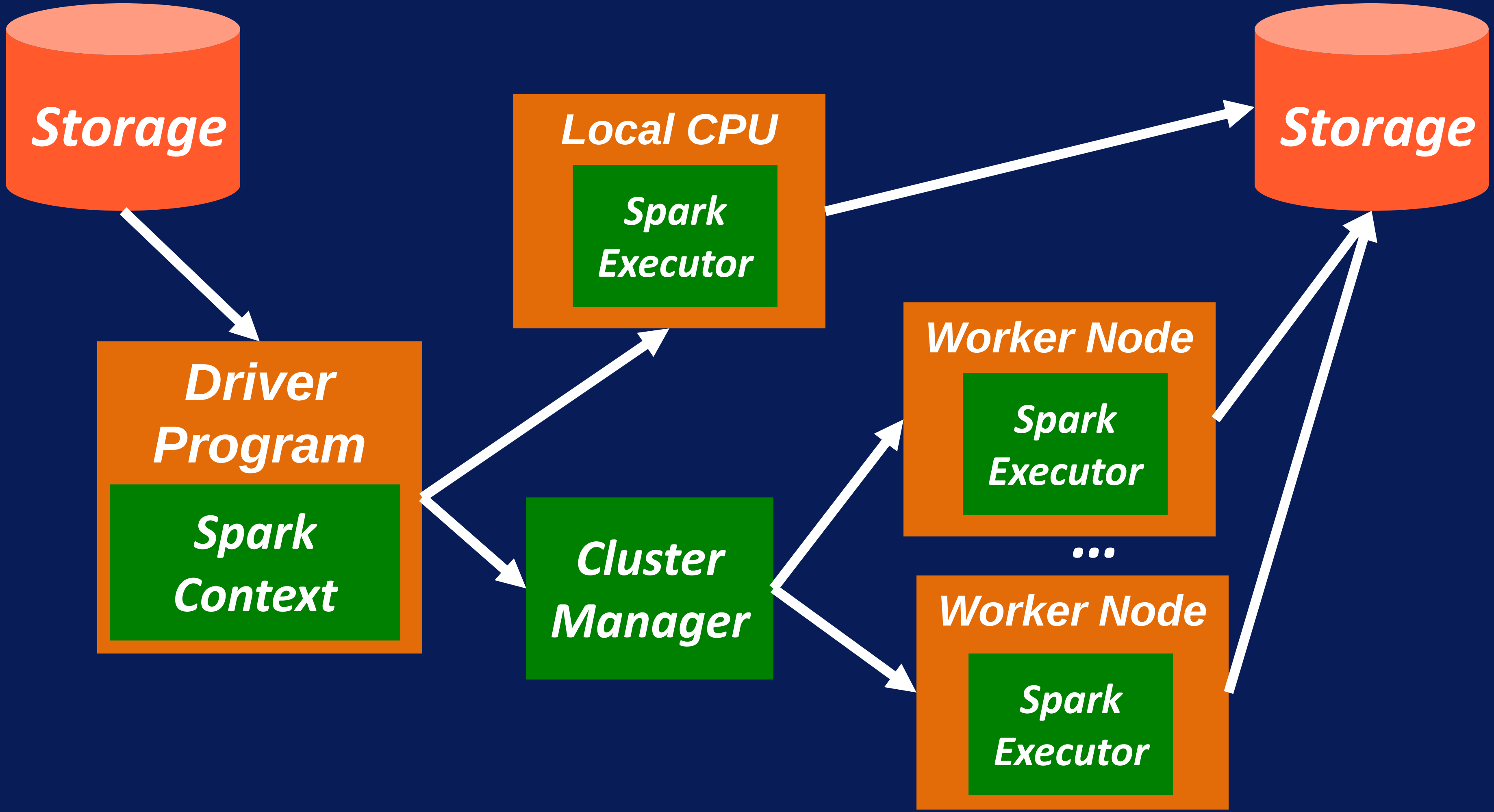
Spark Core:
Programming In Spark

After this video you will be able to..

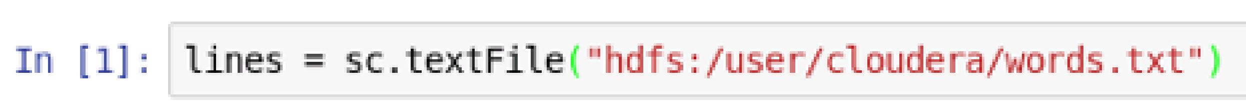
- Use two methods to create RDDs in Spark
- Explain what immutable means
- Interpret a Spark program as a pipeline of transformations and actions
- List the steps to create a Spark program



Storage
Local CPU
Spark Executor
Storage
Driver Program
Spark Context
Worker Node
Spark Executor
Cluster Manager
...
Worker Node
Spark Executor

Creating RDDs

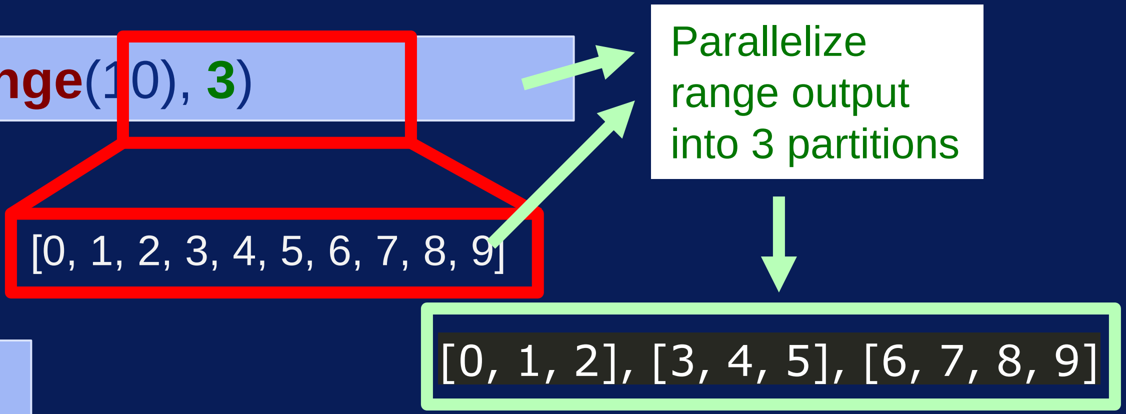
Driver Program



```
In [1]: lines = sc.textFile("hdfs:/user/cloudera/words.txt")
```

```
lines = sc.parallelize([“big”, ”data”])
```

```
numbers = sc.parallelize(range(10), 3)
```



[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

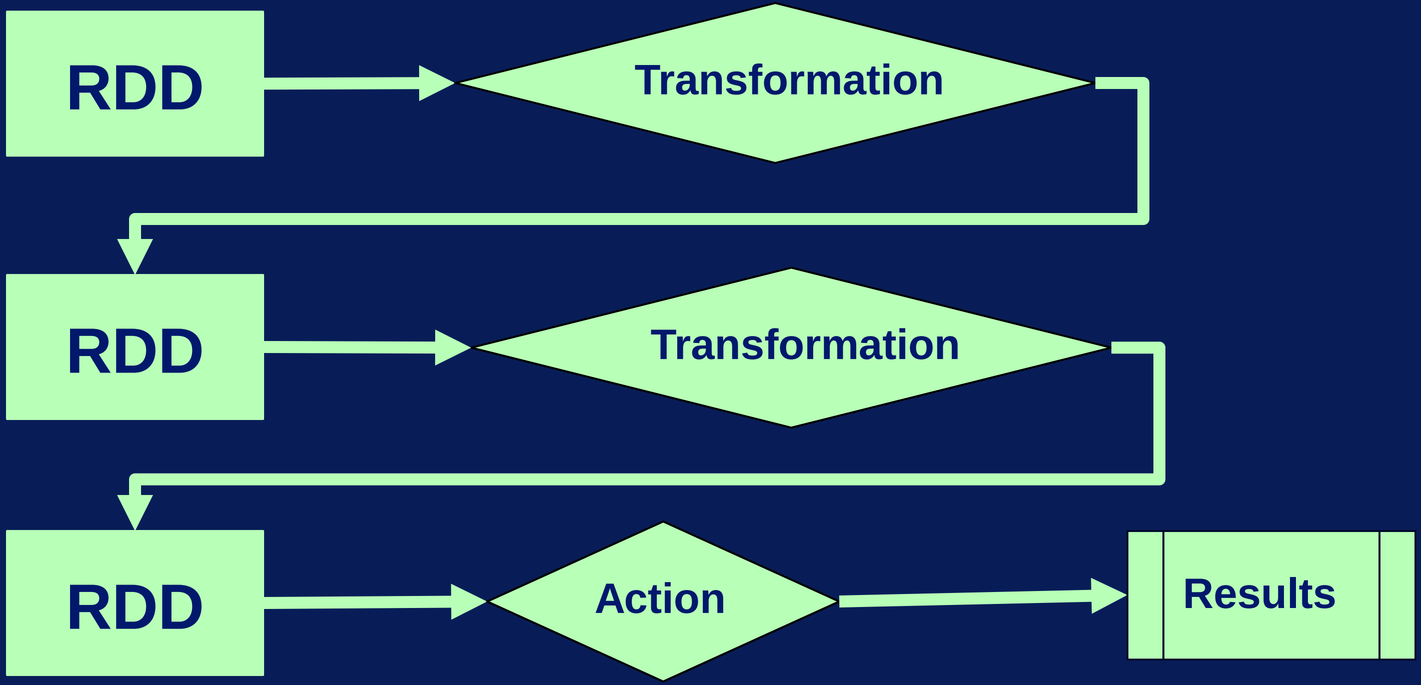
Parallelize range output into 3 partitions

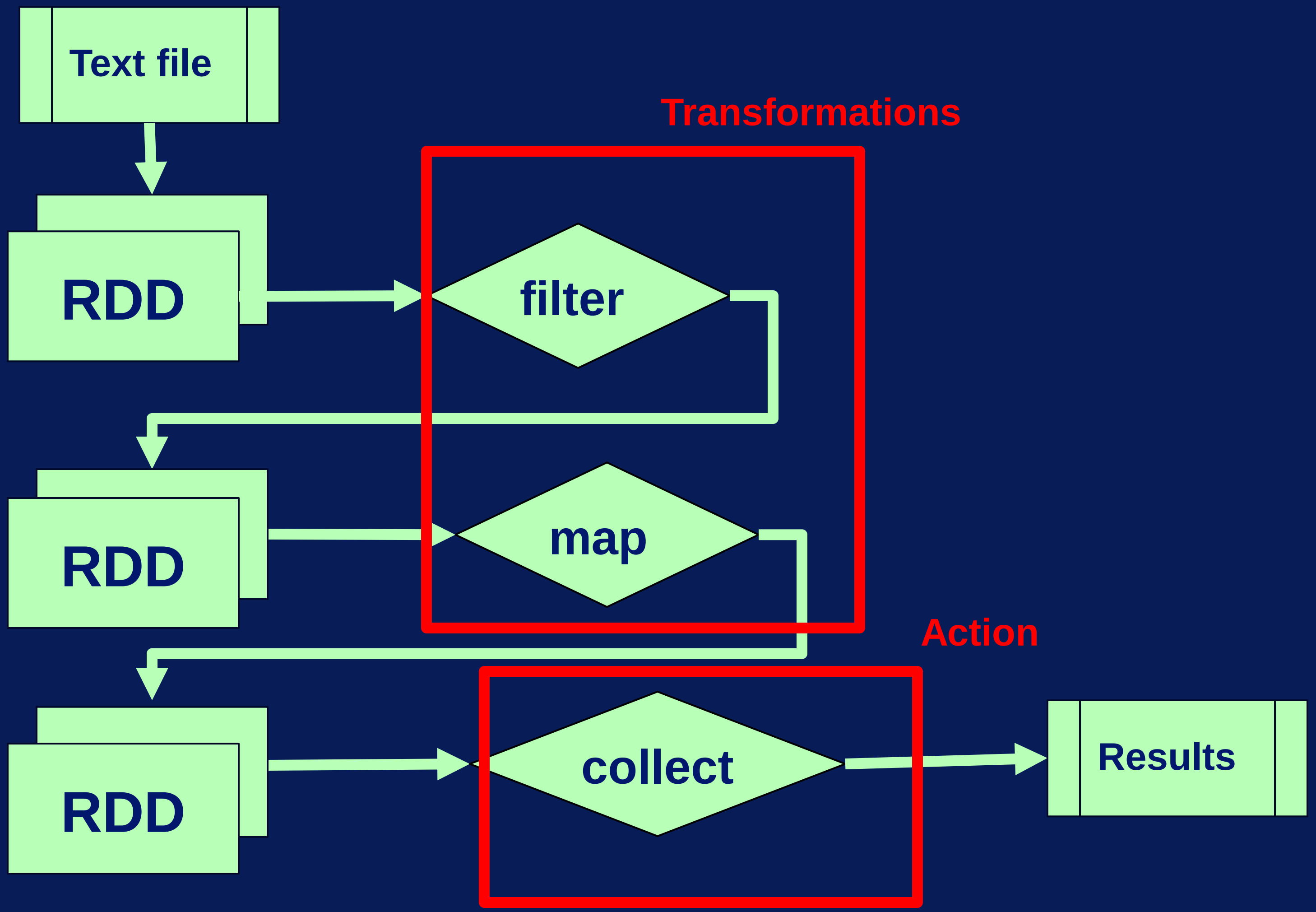
[0, 1, 2], [3, 4, 5], [6, 7, 8, 9]

```
numbers.collect()
```

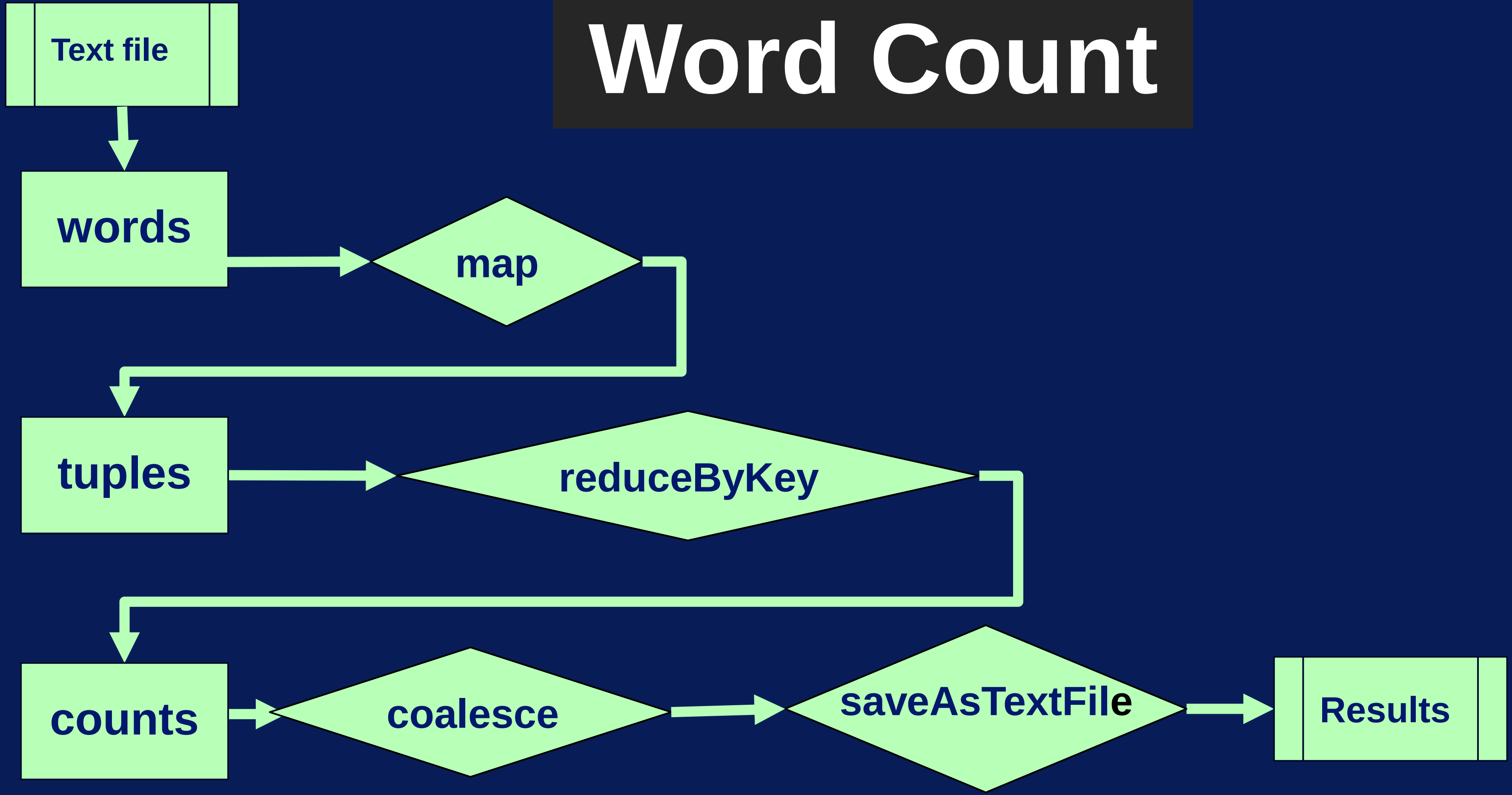
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Processing RDDs

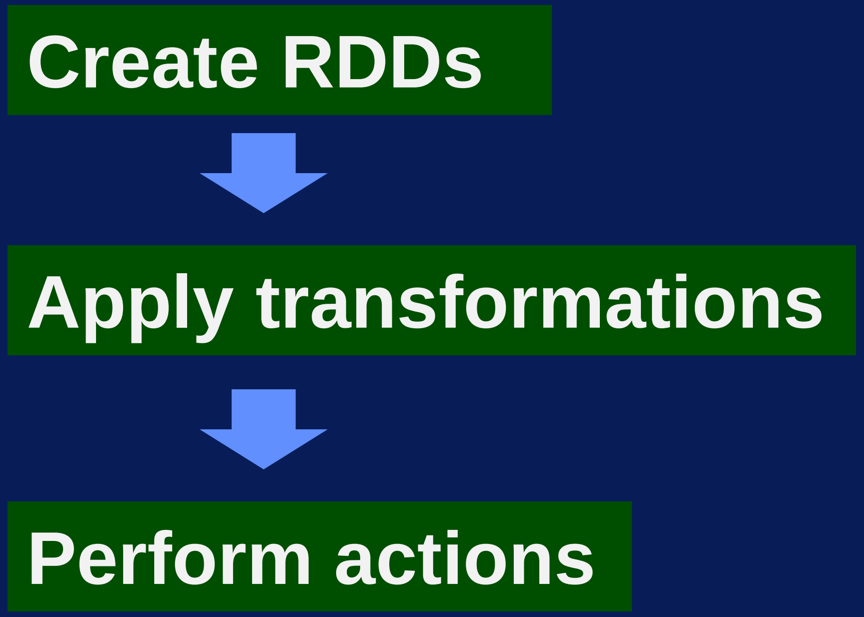



Text file
RDD
Transformations
filter
RDD
map
Action
RDD
collect
Results

Word Count


Text file
words
map
tuples
reduceByKey
counts
coalesce
saveAsTextFile
Results

Programming in Spark



Create RDDs

Apply transformations

Perform actions

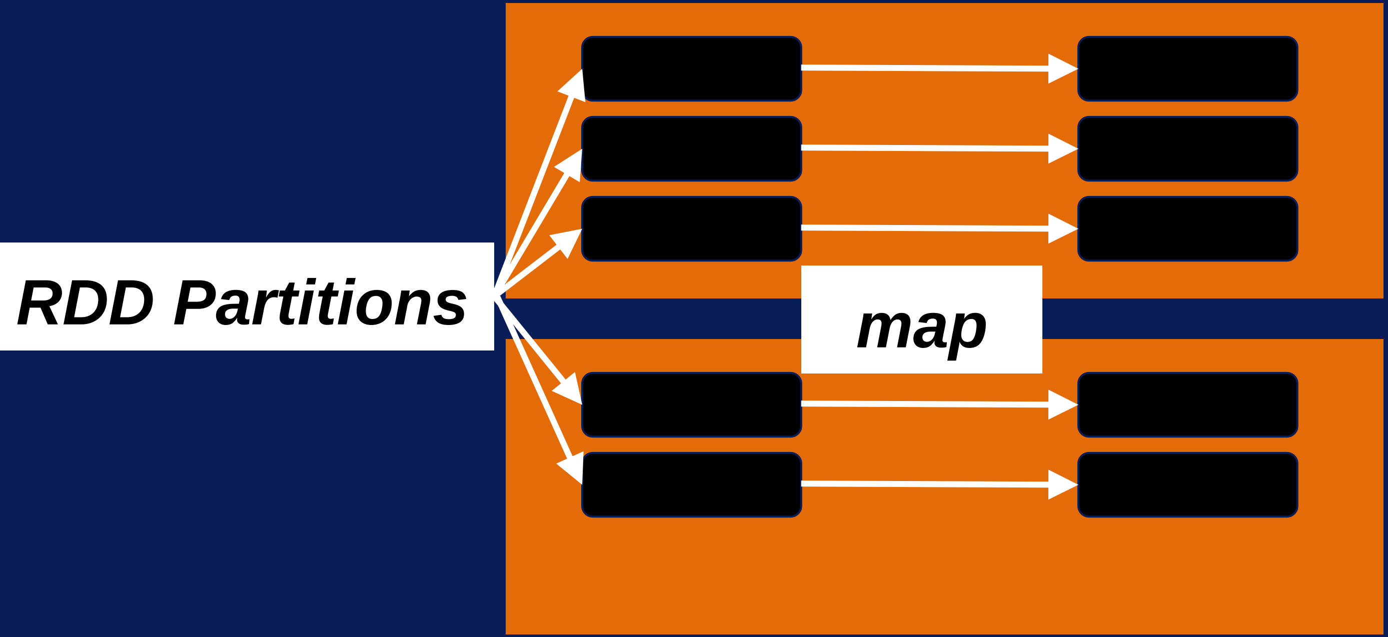
Spark Core: Transformations

After this video you will be able to..

- Explain the difference between a narrow transformation and wide transformation
- Describe map, flatmap, filter and coalesce as narrow transformations
- List two wide transformations

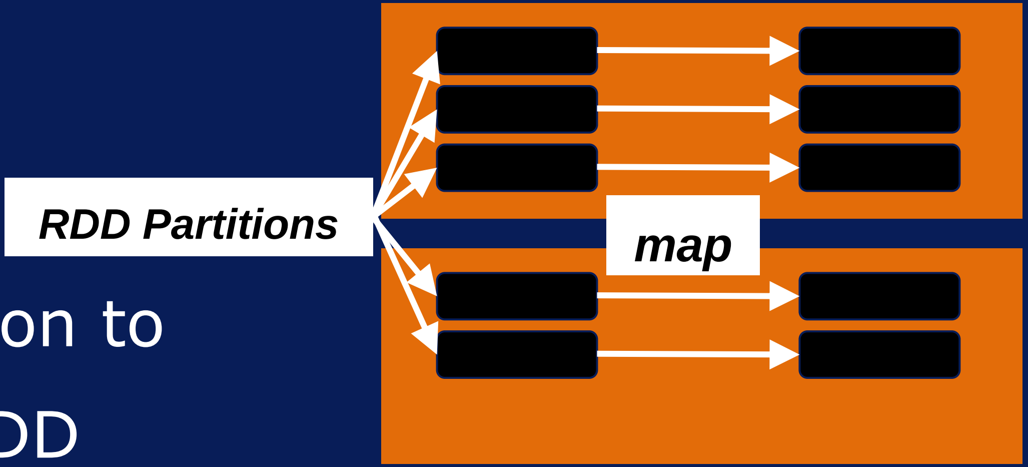
map

map : apply function to each element of RDD



map

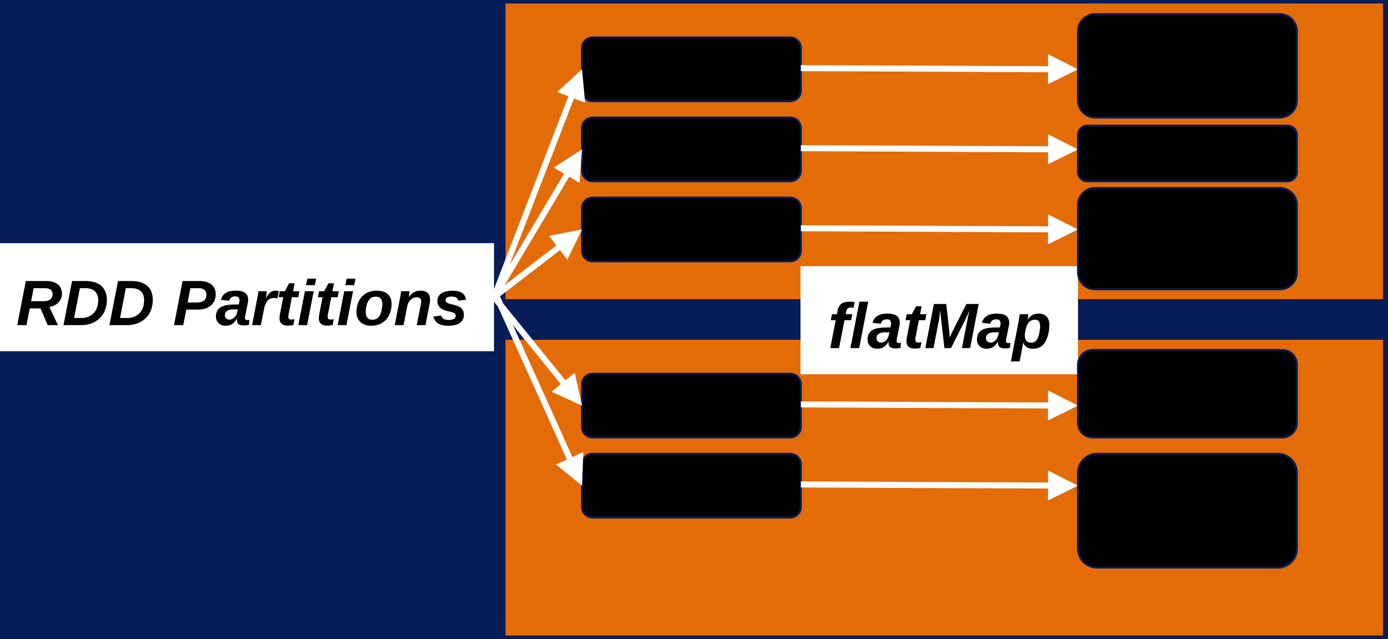
map : apply function to each element of RDD



```
def lower(line):
    return line.lower()
lower_text_RDD = text_RDD.map(lower)
```

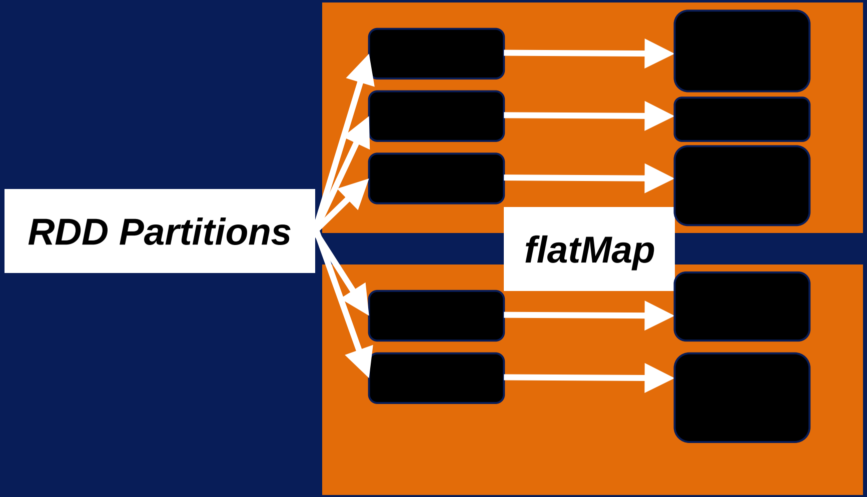
flatMap

flatMap : map then flatten output



flatMap

flatMap : map then flatten output

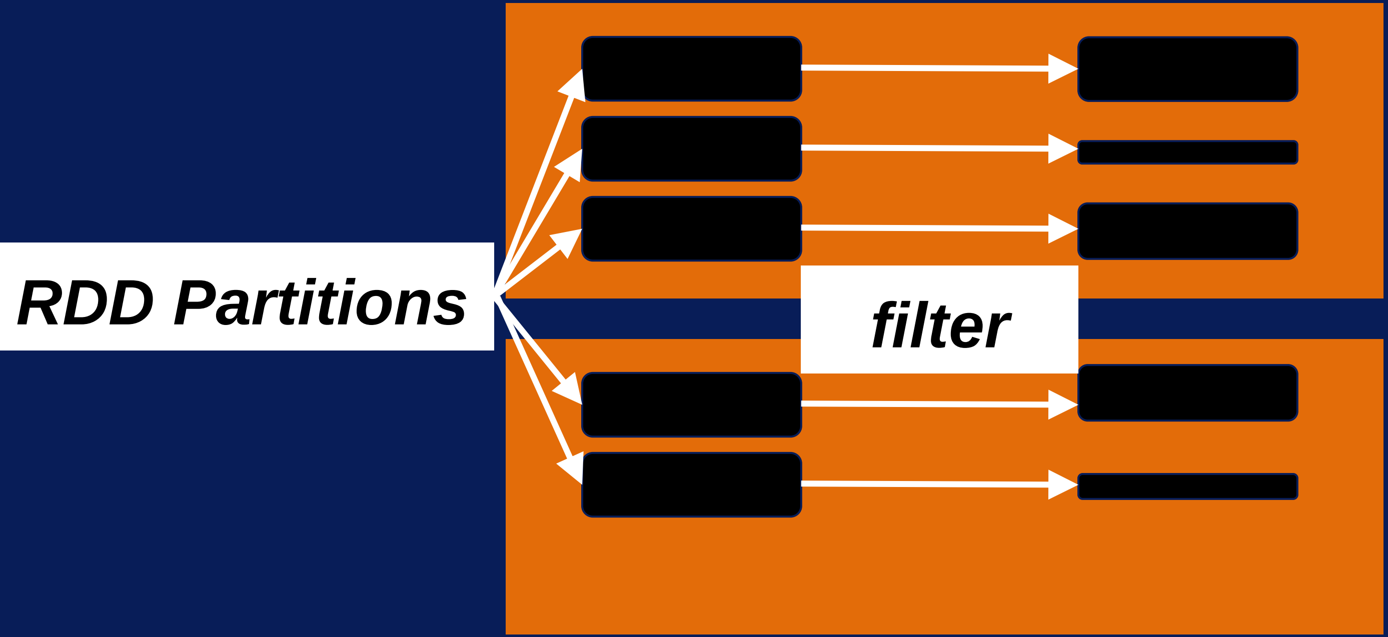


```
def split_words(line):
    return line.split()

words_RDD = text_RDD.flatMap(split_words)
words_RDD.collect()
```

filter

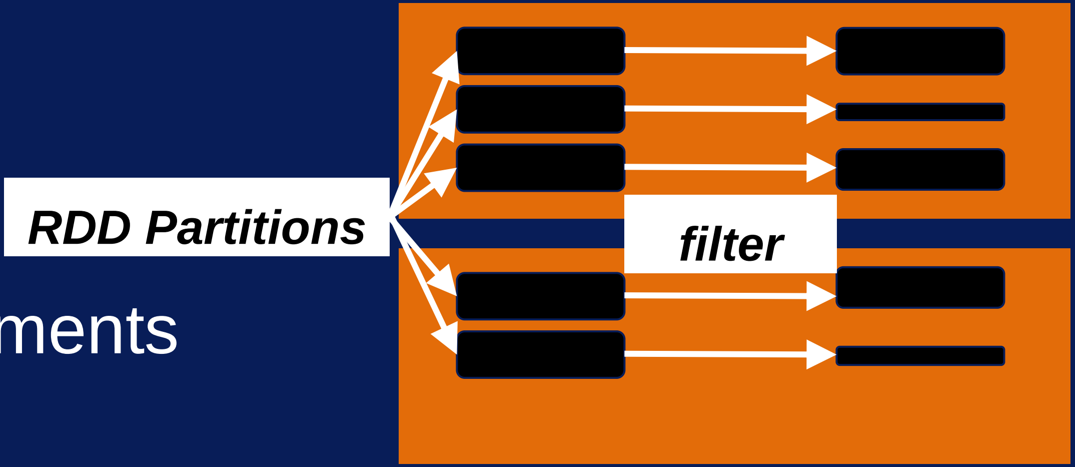
filter : keep only elements where function is true



filter

filter : keep only elements

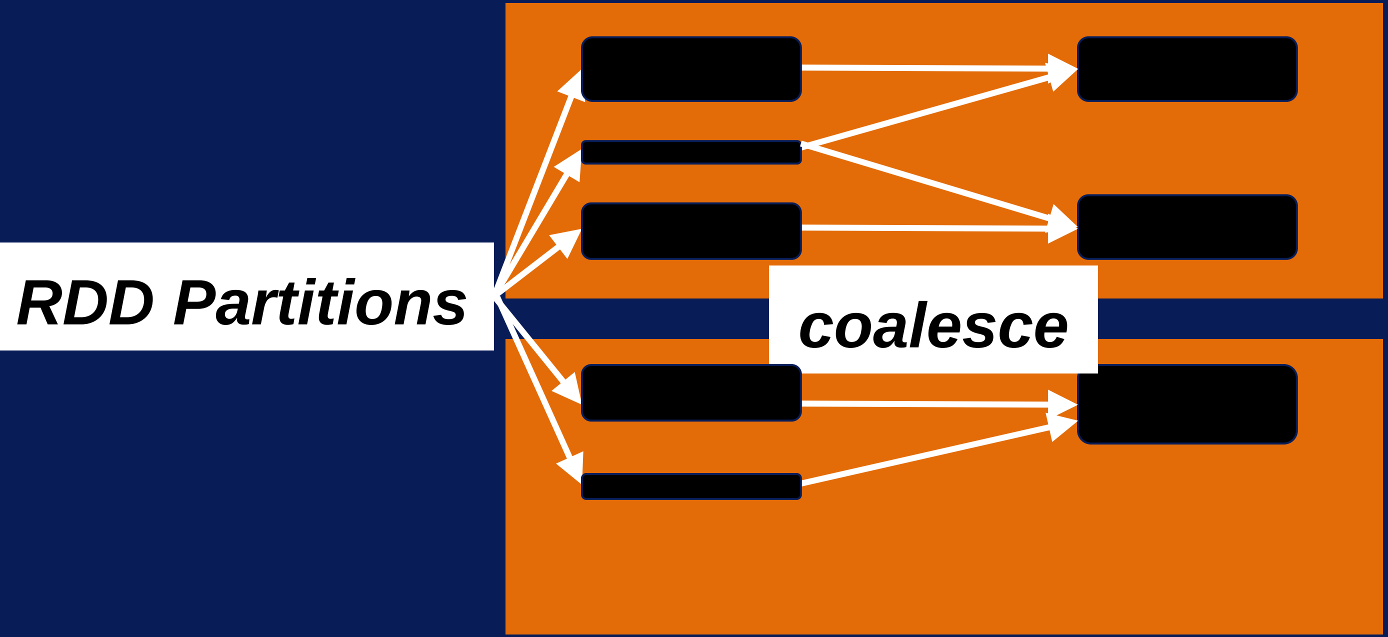
where function is true



```
def starts_with_a(word):
    return word.lower().startswith("a")
words_RDD.filter(starts_with_a).collect()
```

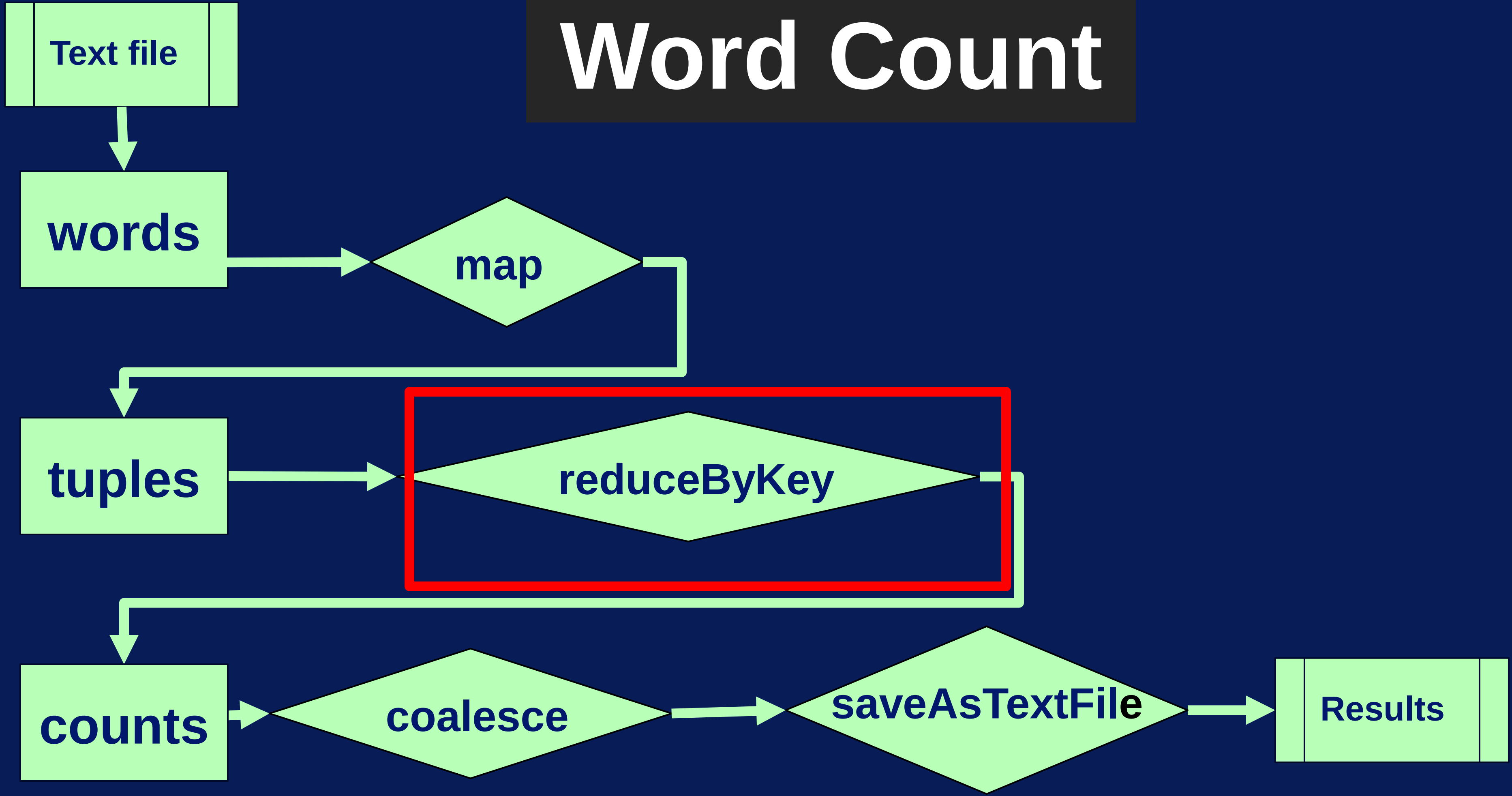
coalesce

coalesce : reduce the number of partitions



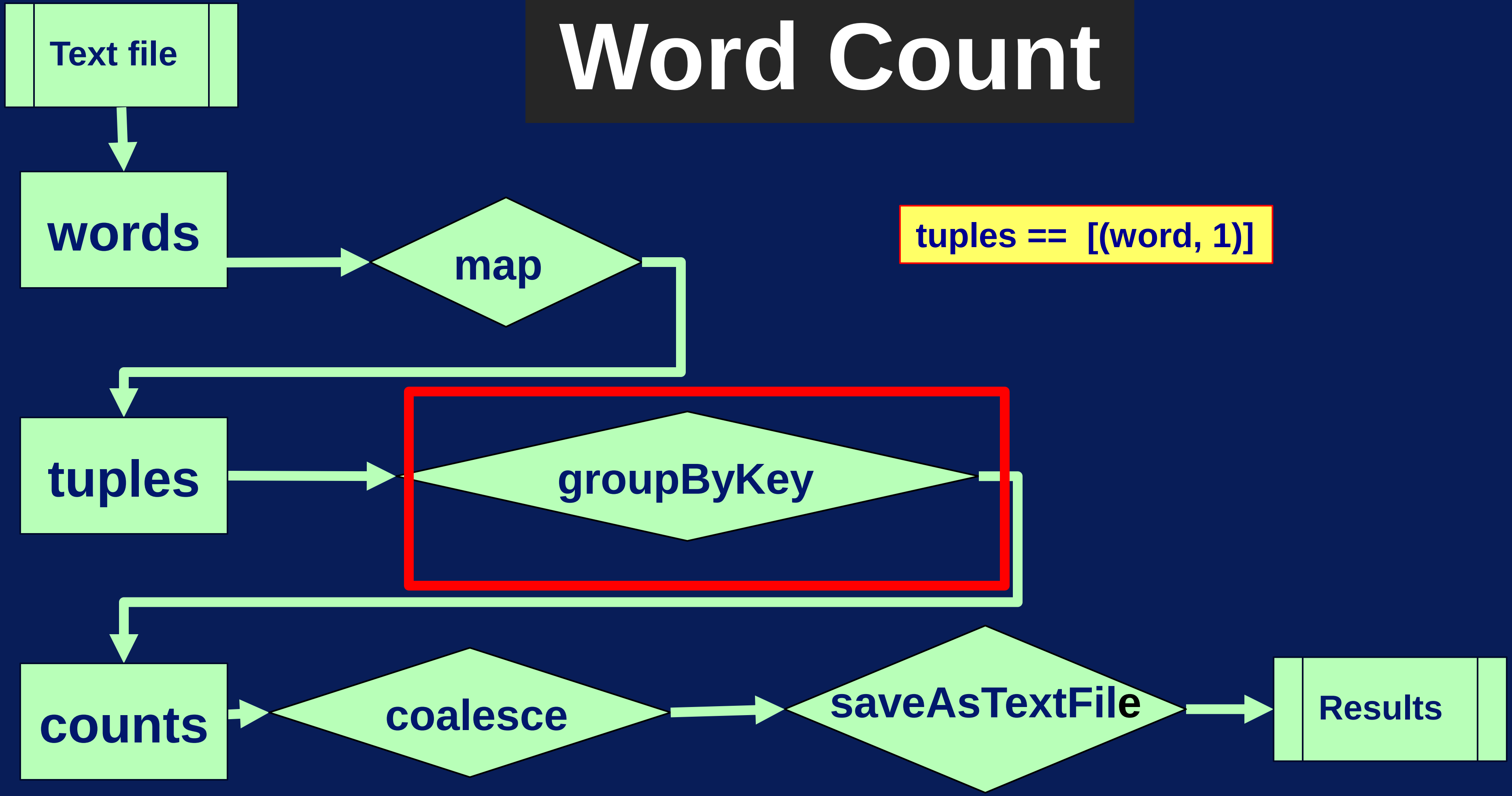
Wide Transformations

Word Count



Text file
words
map
tuples
reduceByKey
counts
coalesce
saveAsTextFile
Results

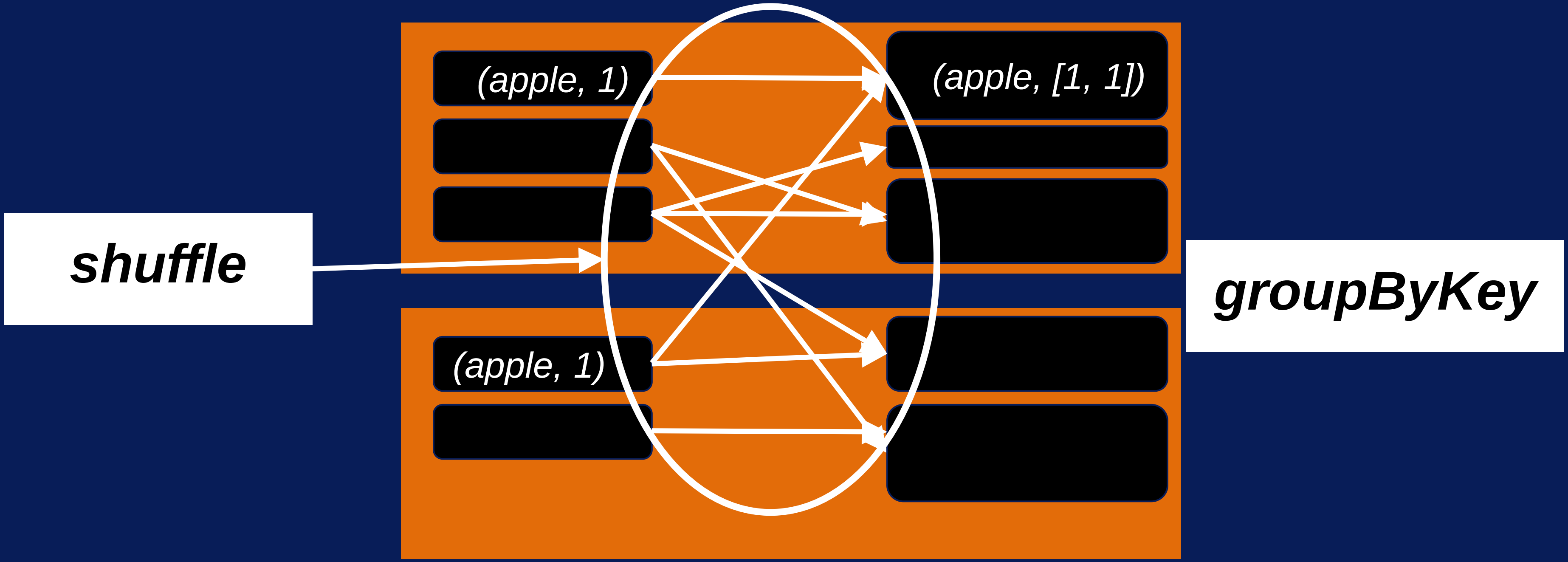
Word Count



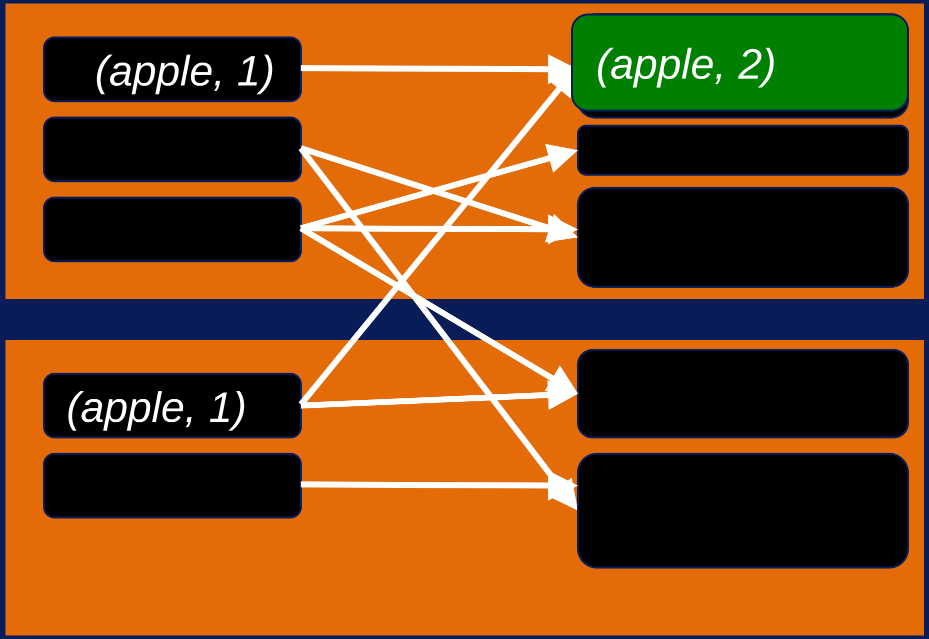
tuples == [(word, 1)]

groupByKey

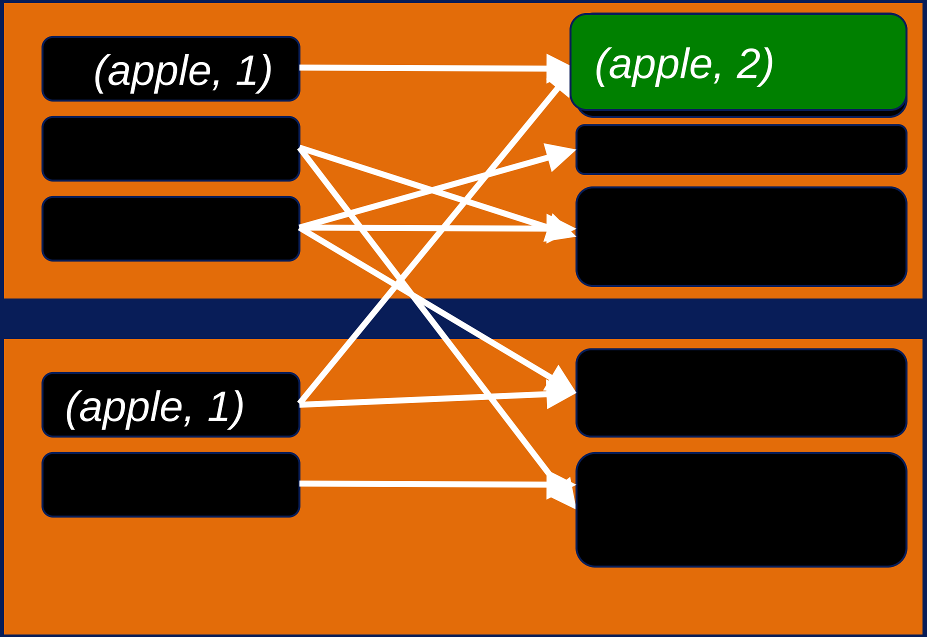
groupByKey : (K, V) pairs => (K, list of all V)



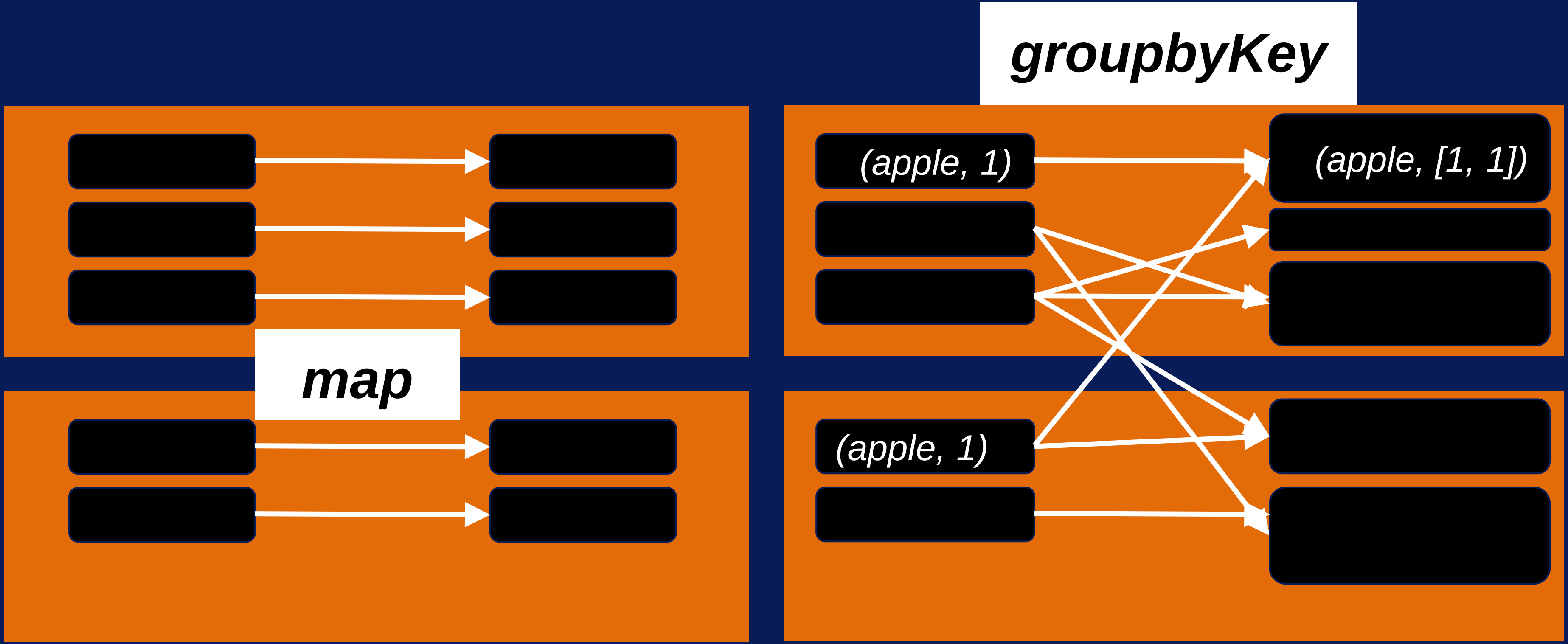
groupByKey + reduce



reduceByKey



Narrow vs Wide



Many more transformations…

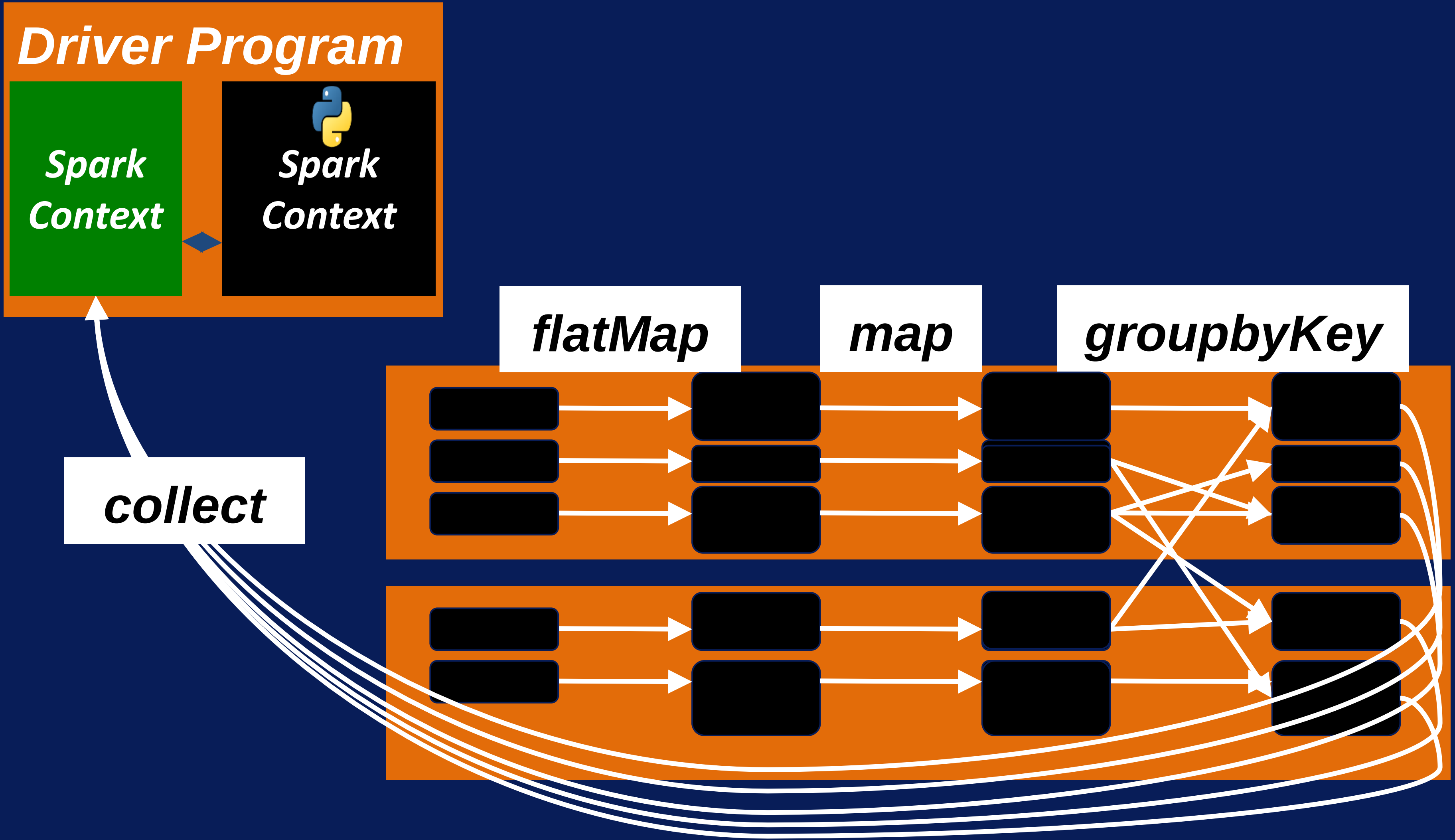
Full list of transformations at:

https://spark.apache.org/docs/1.2.0/programming-guide.html#transformations

Spark Core:
Actions

After this video you will be able to..

- Explain the steps of a Spark pipeline ending with a collect action
- List four common action operations in Spark



Driver Program

Spark Context

Spark Context

flatMap

map

groupbyKey

collect

Some Common Actions

Action	Usage
collect()	Copy all elements to the driver
take(n)	Copy first n elements
reduce(func)	Aggregate elements with func (takes 2 elements, returns 1)
[redacted]	Save to local file or HDFS

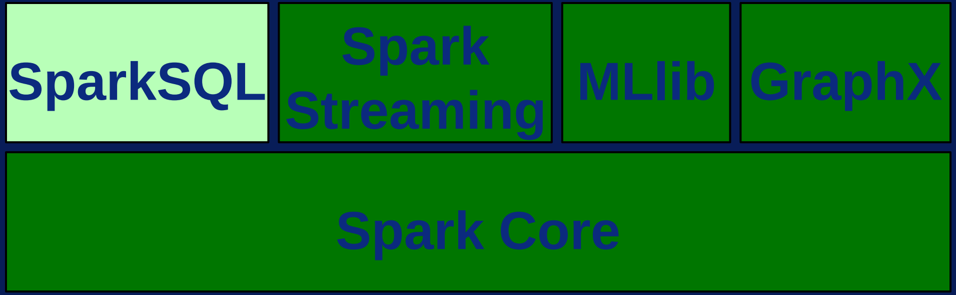
Spark SQL

After this video you will be able to..

- Process structured data using Spark’s SQL module

- Explain the numerous benefits of Spark SQL

Spark SQL



- Enables querying structured and unstructured data through Spark
- Provides a common query language
- Has APIs for Scala, Java and Python to convert results into RDDs

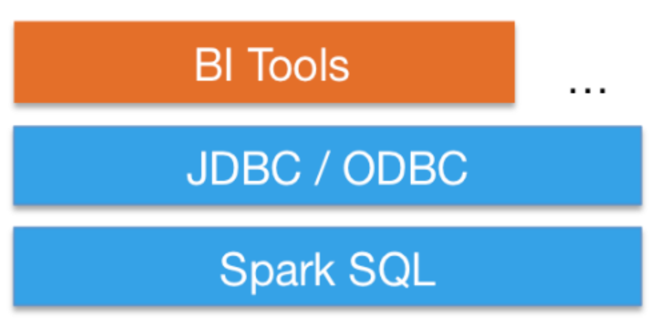
Relational Operations

Perform Relational Processing such as Declarative Queries

Embed SQL queries inside Spark Programs

Business Intelligence Tools

Spark SQL connects to all BI tools that support JDBC or ODBC standard



http://spark.apache.org/

DataFrames

Distributed Data organized as named columns

Look just like a table in relational databases

How to go Relational in Spark ?

Step 1: Create a SQLContext

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

http://spark.apache.org/

How to go Relational in Spark ?

Create a DataFrame from

- an existing RDD
- a Hive table
- data sources

JSON → DataFrame

```
# Read
df = sqlContext.read.json("/filename.json")

# Display
df.show()
```

RDD of Row objects → DataFrame

```
# Read
from pyspark.sql import SQLContext, Row
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a Row.
lines = sc.textFile("filename.txt")
cols = lines.map(lambda l: l.split(","))
data = cols.map(lambda p: Row(name=p[0], zip=int(p[1])))

# Create DataFrame
df = sqlContext.createDataFrame(data)

# Register the DataFrame as a table
df.registerTempTable(”table")

# Run SQL
Output = sqlContext.sql("SELECT * FROM table WHERE …")
```

http://spark.apache.org/

DataFrames are just like tables

http://spark.apache.org/

```
# Show the content of the DataFrame
df.show()

# Print the schema
df.printSchema()

# Select only the "X" column
df.select("X").show()

# Select everybody, but increment the discount by 5%
df.select(df["name"], df["discount"] + 5).show()

# Select people height greater than 4.0 ft
df.filter(df["height"] > 4.0).show()

# Count people by zip
df.groupBy("zip").count().show()
```

Spark SQL

Relational on Spark

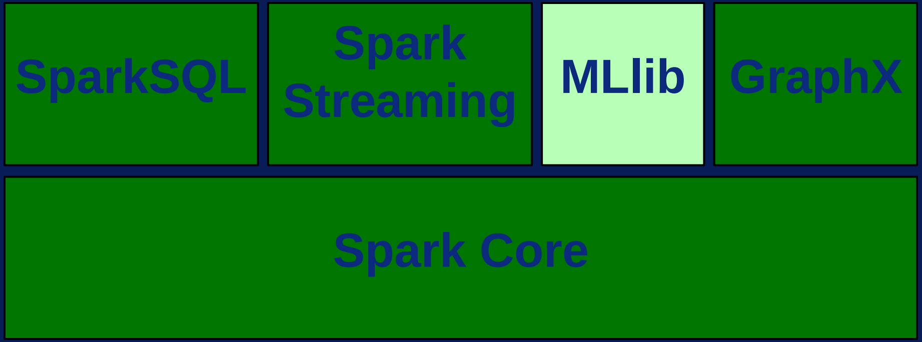
Connect to variety of databases

Deploy business intelligence tools over Spark

Spark MLlib

After this video you will be able to..

- Describe what MLlib is
- List main categories of techniques available in MLlib.
- Explain code segments containing MLlib algorithms.



Spark MLlib

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

MLlib Algorithms & Techniques

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

MlLib Example – Summary Statistics

- Compute column summary statistics

```
from pyspark.mllib.stat import Statistics
```

1

```
# Data as RDD of Vectors
dataMatrix = sc.parallelize([ [1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12] ])
```

2

```
# Compute column summary statistics.
summary = Statistics.colStats(dataMatrix)
```

3

```
print(summary.mean())
print(summary.variance())
print(summary.numNonzeros())
```

4

MLlib Example – Classification

- Build decision tree model for classification

```
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel   1
from pyspark.mllib.util import MLUtils
                                                               2
# Read and parse data
data = sc.textFile(“data.txt”)        3


# Decision tree for classification
model = DecisionTree.trainClassifier                   4
              (parsedData, numClasses=2)
print(model.toDebugString())                               5
model.save(sc, “decisionTreeModel”)                    6
```

MLlib Example – Clustering

- Build k-means model for clustering

```
from pyspark.mllib.clustering import KMeans, KMeansModel
```

1

```
from numpy import array
```

2

```
# Read and parse data
data = sc.textFile(“data.txt”)
parsedData = data.map(lambda line:
            array([float(x) for x in line.split(‘ ‘)]))
```

3

```
# k-means model for clustering
clusters = Kmeans.train (parsedData, k=3)
```

4

```
print(clusters.centers)
```

5

Main Take-Aways

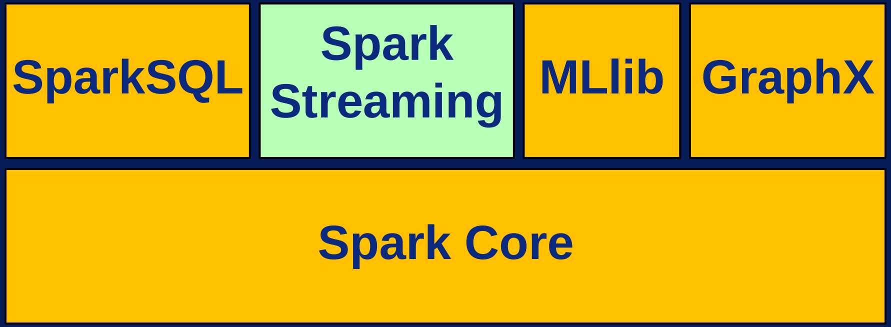
- MLlib is Spark’s machine learning library.
  - Distributed implementations
- Main categories of algorithms and techniques:
  - Machine learning
  - Statistics
  - Utility for ML pipeline

Spark Streaming

After this video you will be able to..

- Summarize how Spark reads streaming data
- List several sources of streaming data supported by Spark
- Describe Spark’s sliding windows

Spark Streaming

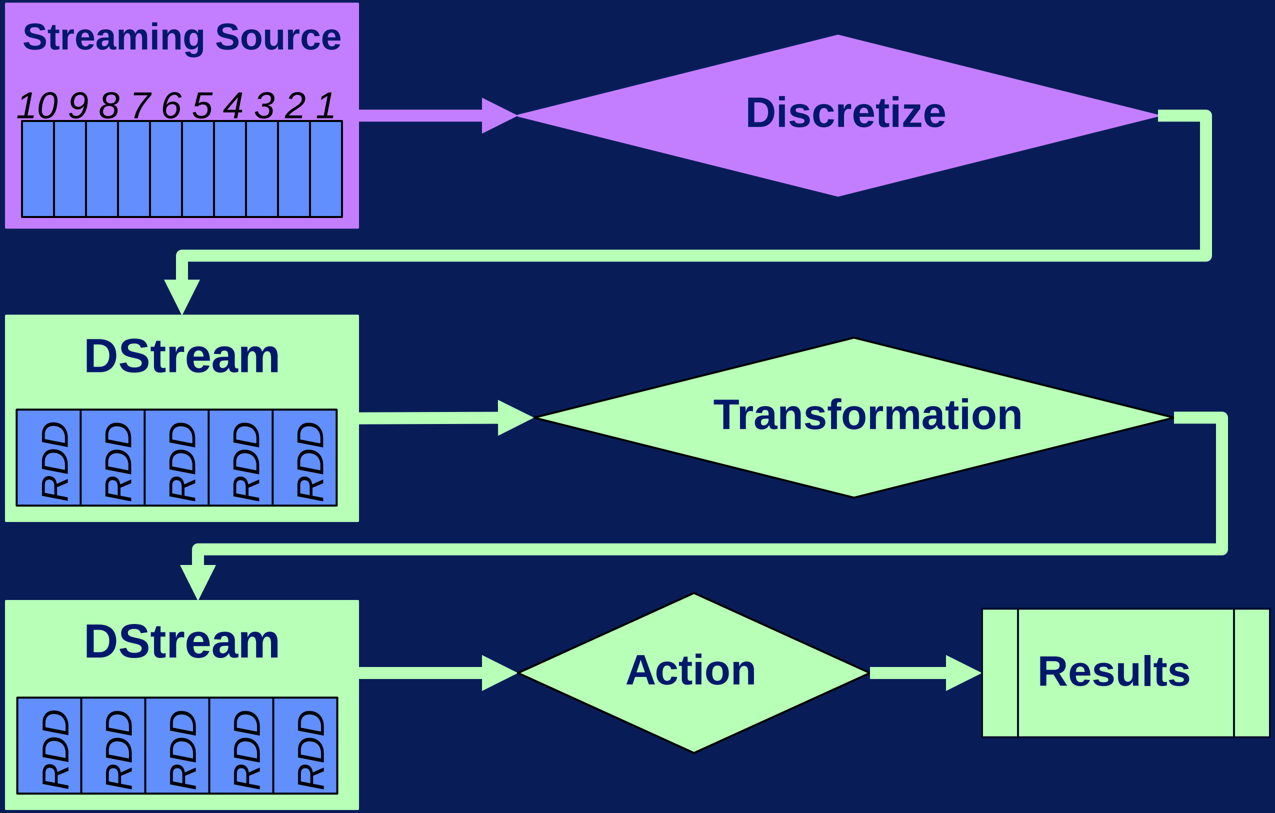


- Scalable processing for real-time analytics
- Data streams converted to discrete RDDs
- Has APIs for Scala, Java, and Python

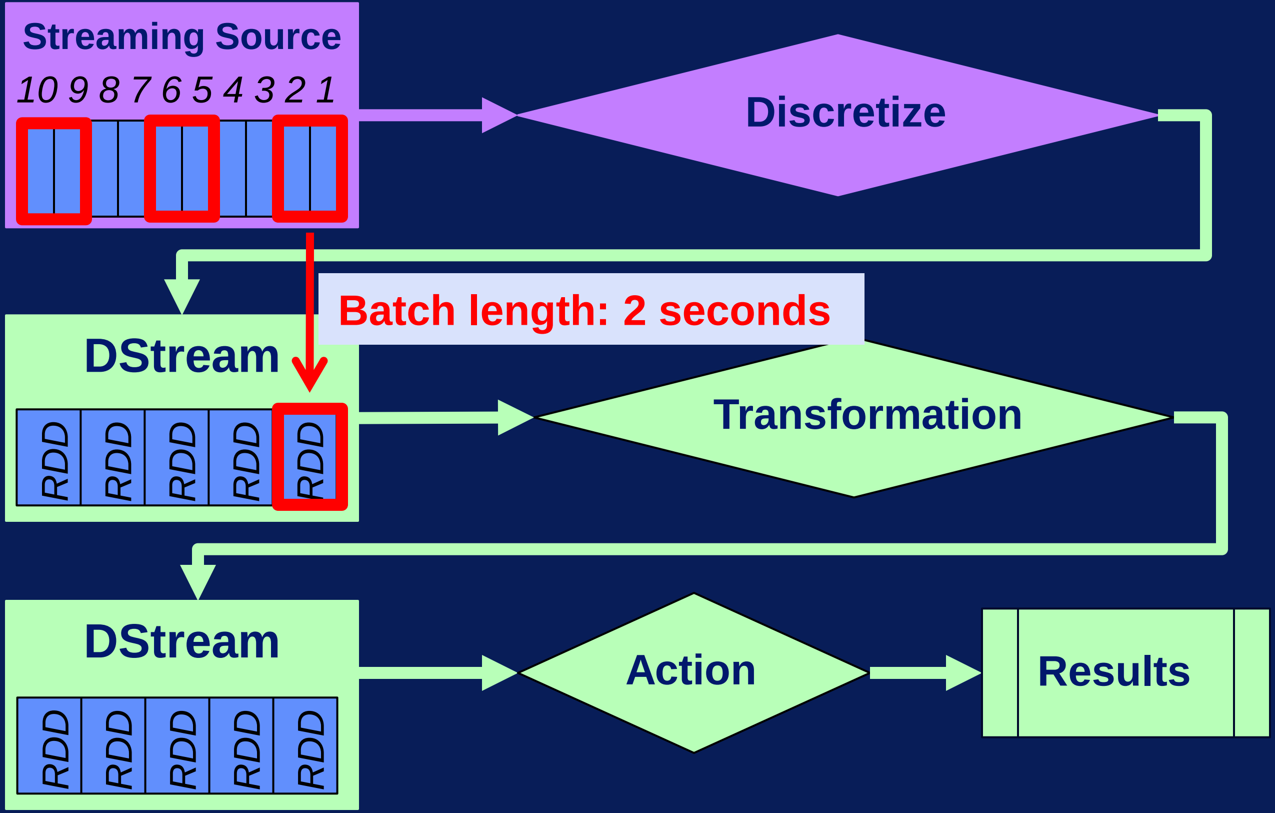
Spark Streaming Sources

- Kafka
- Flume
- HDFS
- S3
- Twitter
- Socket
- …etc.

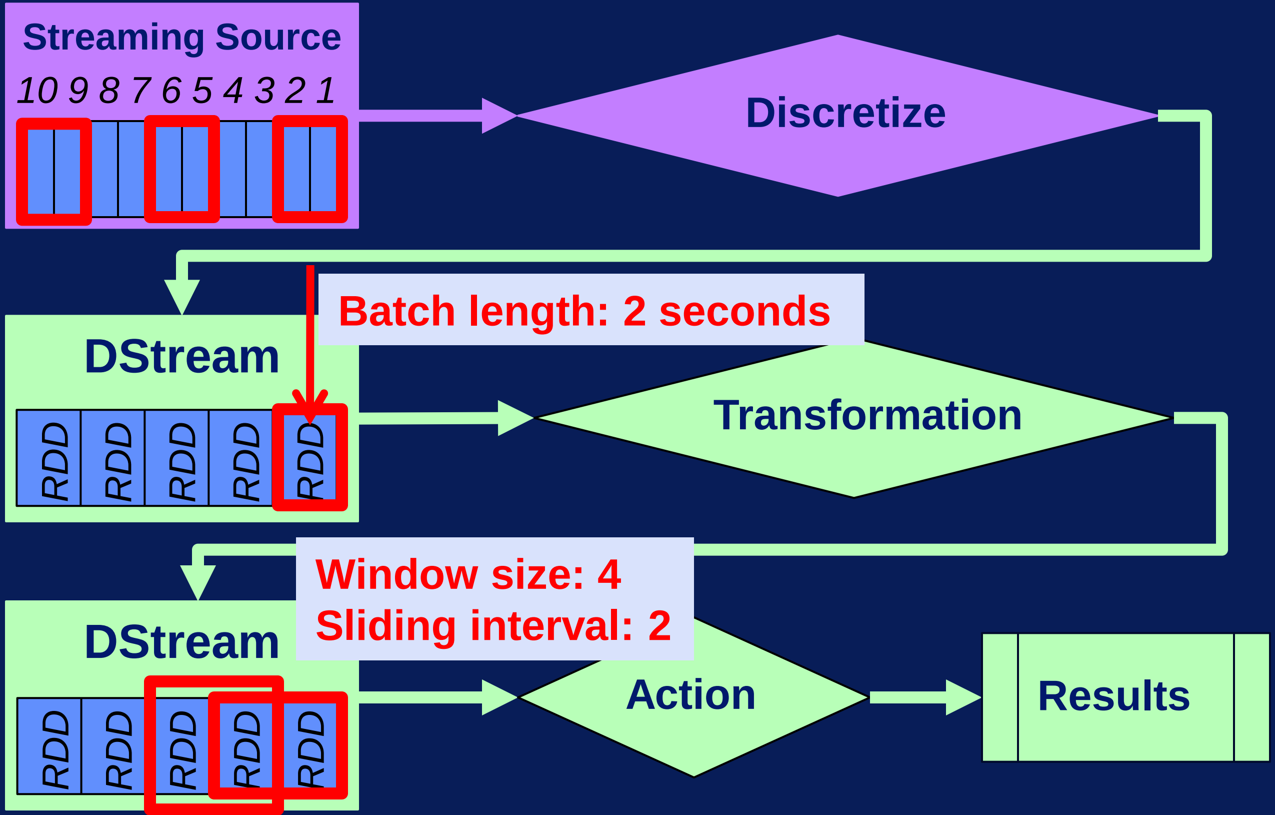
Creating and Processing DStreams



Creating and Processing DStreams



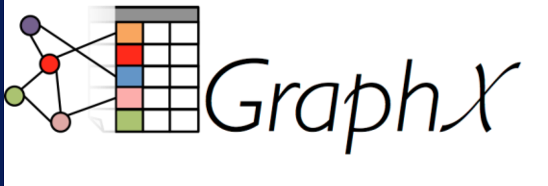
Creating and Processing DStreams



Main Take-Aways

- Spark uses DStreams to make discrete RDDs from streaming data.
  - Same transformations and calculations applied to batch RDDs can be applied
- DStreams can create a sliding window to perform calculations on a window of time.

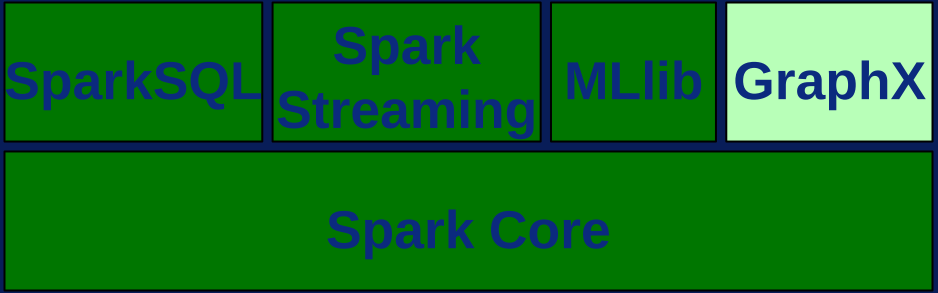
Spark GraphX



After this video you will be able to..

- Describe what GraphX is
- Explain how Vertices and Edges are stored
- Describe how Pregel works at a high level

Spark GraphX



SparkSQL

Spark Streaming

MLlib

GraphX

Spark Core

GraphX is Apache Spark's API for graphs and graph-parallel computation.

GraphX uses a property graph model.

Both Nodes and Edges can have attributes and values

Properties → Tables

Vertex Table

Node properties

Edge Table

Edge properties

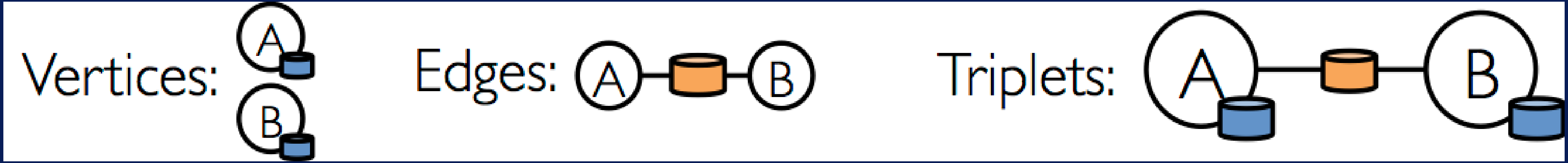
GraphX uses special RDDs

VertexRDD[A] extends RDD[(VertexID, A)]

EdgeRDD[ED, VD] extends RDD[Edge[ED]]

Triplets

The triplet view logically joins the vertex and edge properties.



http://spark.apache.org/docs/latest/img/triplet.png

Pregel API

Bulk-synchronous parallel messaging mechanism

Constrained to the topology of the graph

GraphX

Graph Parallel Computations

Special RDDs for storing Vertex and Edge information

Pregel operator works in a series of super steps