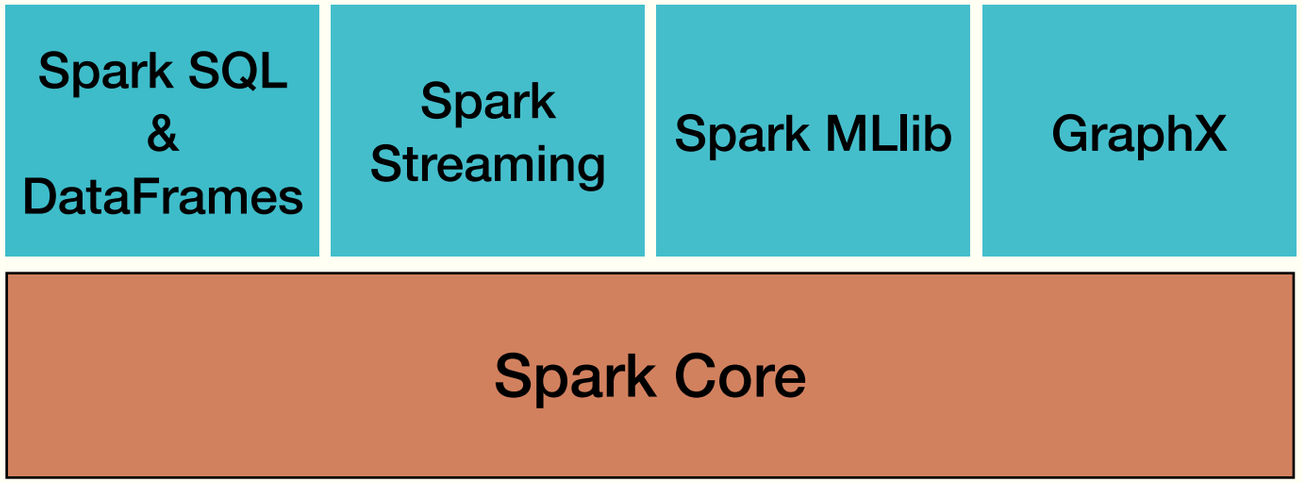
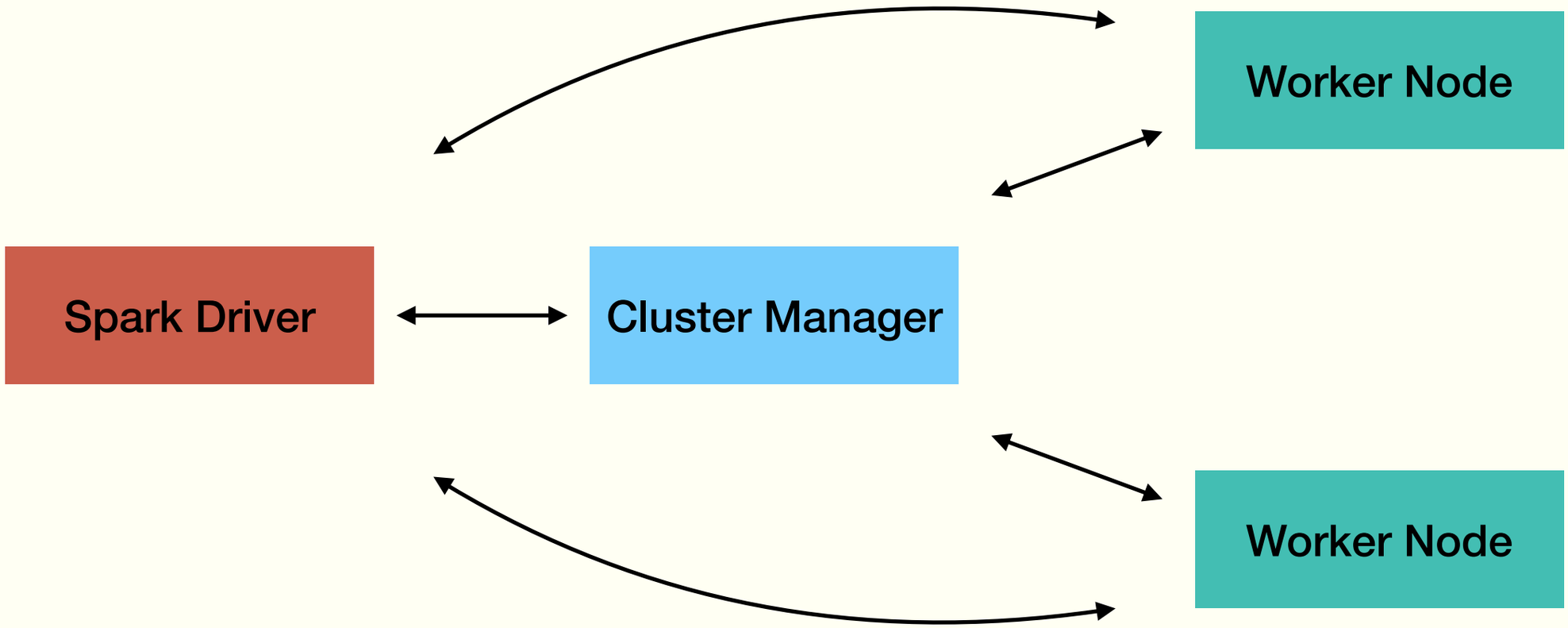
Apache Spark

Spark Ecosystem

- A unified framework
  - simplified installation and setup
- High Performance:
  - Memory-based data processing
- Common API:
  - ease or programming



Spark Architecture Overview



Spark Architecture Overview

- Driver:
  - Entry point for spark application
  - Aggregates results for spark application

- Worker Nodes:
  - runs tasks assigned to them in parallel
  - each worker node has an executor that executes the tasks and interacts with the driver

- Cluster Manager:
  - allocates and manages resources for the cluster of nodes
  - Spark provides built-in standalone cluster manager
  - Apache Hadoop YARN, Apache Mesos and Kubernetes are supported as alternate cluster manager

RDD

- Resilient Distributed Dataset (RDD):
  - Resiliency: RDD can be reconstructed when needed
  - Distributed: data is partitioned across nodes
  - Dataset: data is abstracted and made available through low-level APIs
- Creating RDDs:
  - Parallelizing existing collection
  - Reading external datasets (HDFS, HBase, etc)
- Operations:
  - RDD supports generic operations on the data through *transformations* and *actions*:
  - Transformation - creates new RDD from existing RDD (eg. map)
  - Action: returns results to the driver (eg. reduce)

Transformations

- Transformations apply functions to RDDs and create new RDDs
- Transformed RDD can be reconstructed if needed
- Transformation operations are applied *lazily,* meaning the results are computed only when needed
- Examples:
  - map(func)
  - filter(func)
  - flatMap(func)
  - mapPartitions(func)
  - groupByKey([numPartitions])
  - reduceByKey(func, [numPartitions])
  - and others..

Actions

- Actions run computation on the RDDs and return result to the driver program
- Examples:
  - reduce(func)
  - collect()
  - count()
  - countByKey()
  - foreach(func)
  - saveAsTextFile(path)
  - and others ...

DataFrame and DataSet

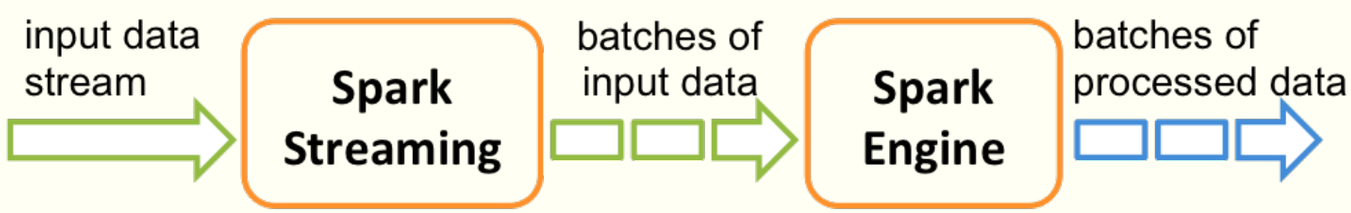
- DataSet:
  - Distributed collection of data
  - Provides optimization using Spark SQL's execution engine.
- DataFrame:
  - Dataset organized into named columns
  - Conceptually similar to R/Python dataframe, but with distributed data management and optimized computation functionality.
  - Constructed from data files, RDDs, Hive tables or external databases.

Spark SQL

- SQL execution engine on top of Spark Core
- Performs optimizations based on the computation information available through the APIs
- Provides SQL and Dataset APIs
- Run SQL queries, and returns results as Datasets/DataFrames

Spark Streaming

- Stream processing of data across the nodes of the cluster
- Extension of Spark Core
- Data source can be Kafka, Kinesis, or TCP sockets.
- Data results can be pushed to filesystems, databases and dashboards
- Support processing of stream data through high-level functions



source: https://spark.apache.org/docs/latest/streaming-programming-guide.html

MLlib

- Spark's Machine Learning Library
- Provides machine learning algorithms for scale
- Algorithms include classification, regression, clustering
- DataFrame based API as the primary API since Spark 2.0

GraphX

- Spark component for graph data abstraction and graph-parallel computation.
- Graph abstraction (Property Graph) is a directed multigraph with vertex and edge properties
- Graph is built on top of RDD, and so is scalable and fault-tolerant
- GraphX provides graph computation through operators and Pregel
- Sample Operators:
  - mapVertices()
  - mapEdges()
  - pageRank()
  - subgraph()

Spark Shell - Scala

- Launch using ./bin/spark-shell
- Interactive shell for running spark programs using Scala programming language

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_271)
Type in expressions to have them evaluated.
Type :help for more information.
```

Example - Spark Shell (Scala)

```
scala> val data = Array(1, 2, 3, 4, 5, 6, 7, 8)
data: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8)

scala> val distData = sc.parallelize(data)
distData: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[1] at parallelize at <console>:26

scala> distData.reduce((a, b) => a+b)
res4: Int = 36
```

Spark Shell - Python

- Launch using ./bin/pyspark
- Interactive shell for running spark programs using Python programming language

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Python version 3.9.5 (default, May 18 2021 12:31:01)
Spark context Web UI available at http://10.0.0.40:4041
Spark context available as 'sc' (master = local[*], app id = local-1635311447563).
SparkSession available as 'spark'.
>>>
```

Example - Spark Shell (Python)

```
>>> data = [1, 2, 3, 4, 5, 6, 7, 8]

>>> distData = sc.parallelize(data)

>>> distData.reduce(lambda a, b: a+b)
36
```

Example - Word Count

- Read data from a text file
- Data consists of text from "*1989* (Taylor Swift album)" Wikipedia page
- Perform Transformation and Action operations to count words in the overview article.
- Write results to local directory

Wiki1989.txt — Edited

```
1989 is the fifth studio album by American singer-songwriter Taylor Swift.
It was released on October 27, 2014, by Big Machine Records. Following the
release of her genre-spanning fourth studio album Red (2012), noted for
pop hooks and electronic elements, the media questioned the validity of
Swift's status as a country artist. Inspired by 1980s synth-pop to create
a record that shifted her sound and image from country to mainstream pop,
Swift enlisted Max Martin as co-executive producer, and titled her fifth
album after her birth year as a symbolic rebirth of her artistry.

The album's synth-pop sound is characterized by heavy synthesizers,
programmed drums, and processed backing vocals. Swift wrote the songs
inspired by her personal relationships, which had been a trademark of her
songwriting. The songs on 1989 express lighthearted perspectives,
departing from her previous hostile attitude towards failed romance. Swift
and Big Machine promoted the album extensively through product
endorsements, television, radio appearances and social media. They pulled
1989 from free streaming services such as Spotify, prompting an industry
discourse on the impact of streaming on music sales.[note 3]
```

part-00001 — Edited

1989 Done

```
('1989', 7)
('the', 23)
('fifth', 2)
('studio', 2)
('by', 6)
('singer-songwriter', 1)
('Taylor', 1)
('Swift.', 1)
('released', 1)
('on', 6)
('October', 1)
('2014,', 1)
('Big', 2)
('Records.', 1)
('Following', 1)
('release', 1)
```

part-00000 — Edited

1989 Done Replace

```
('is', 2)
('album', 7)
('American', 1)
('It', 3)
('was', 4)
('27,', 1)
('Machine', 2)
('of', 14)
('her', 8)
('genre-spanning', 1)
('fourth', 1)
('pop', 2)
('questioned', 1)
('as', 5)
('country', 2)
('artist.', 1)
```

Example - Word Count (Scala)

```
scala> val file = sc.textFile("Wiki1989.txt")

scala> val counts = file.flatMap(line =>
line.split(" ")).map(word => (word,
1)).reduceByKey(_ + _)

scala> counts.saveAsTextFile("Results.txt")
```

Example - Word Count (Python)

```
>>> file = sc.textFile("Wiki1989.txt")

>>> counts = file.flatMap(lambda line:
line.split(" ")).map(lambda word: (word,
1)).reduceByKey(lambda a, b: a+b)

>>> counts.saveAsTextFile("Results.txt")
```

Summary

- Spark is an integrated, high-perform and distributed compute framework
- RDDs provide low-level abstraction for distributed data with compute operations
- RDD operations include:
  - Transformation
  - Action
- DataSet and DataFrame are high level data abstraction with SQL engine optimization
- Spark components include Spark SQL, Spark Streaming, MLlib and GraphX