Here's a comparison of SparkContext and SparkSession in tabular format, along with examples for each:

| **Feature** | **SparkContext** | **SparkSession** |
| --- | --- | --- |
| **Definition** | Entry point for Spark applications, manages cluster connection and resource allocation. | Unified entry point for Spark, integrates SparkContext, SQLContext, and HiveContext. |
| **Level of Abstraction** | Lower-level abstraction primarily for RDDs. | Higher-level abstraction for DataFrames and Datasets. |
| **Creation** | Created using SparkConf. | Created using SparkSession.builder. |
| **Access to SQL** | Does not directly support SQL operations. | Directly supports SQL queries and DataFrames. |
| **Data Sources** | Limited support for structured data sources. | Extensive support for various data formats (e.g., JSON, Parquet, etc.). |
| **Use Case** | Ideal for applications focused solely on RDDs. | Recommended for applications that involve structured data or require SQL functionalities. |
| **Example of Creating** | sc = SparkContext(appName="MyApp") | spark = SparkSession.builder.appName("MyApp").getOrCreate() |
| **Example Usage** | rdd = sc.parallelize([1, 2, 3]) | df = spark.read.csv("data.csv") |
| **Stopping the Context** | sc.stop() | spark.stop() |

**Examples**

**SparkContext Example**

from pyspark import SparkConf, SparkContext

# Create SparkConf and SparkContext

conf = SparkConf().setAppName("MyApp")

sc = SparkContext(conf=conf)

# Create an RDD

rdd = sc.parallelize([1, 2, 3, 4, 5])

# Perform a transformation and action

result = rdd.map(lambda x: x \* 2).collect()

print(result) # Output: [2, 4, 6, 8, 10]

# Stop the SparkContext

sc.stop()

**SparkSession Example**

from pyspark.sql import SparkSession

# Create SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()

# Read a CSV file into a DataFrame

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

# Show the DataFrame

df.show()

# Perform a SQL query

df.createOrReplaceTempView("my\_table")

result = spark.sql("SELECT \* FROM my\_table WHERE age > 30")

result.show()

# Stop the SparkSession

spark.stop()

In Apache Spark, SparkContext and SparkSession are two key components, but they serve different purposes and are used in different contexts. Here's a brief overview of each:

**SparkContext**

* **Definition**: SparkContext is the entry point to Spark functionality and is responsible for connecting to a Spark cluster.
* **Responsibilities**:
  + Initializes Spark and sets up the environment for running Spark applications.
  + Manages the distribution of data and resources across the cluster.
  + Provides methods for creating RDDs (Resilient Distributed Datasets) and broadcasting variables.
* **Usage**: Typically used in the early versions of Spark (prior to Spark 2.0) and in scenarios focused on RDDs.

**SparkSession**

* **Definition**: SparkSession is a newer and more unified entry point introduced in Spark 2.0, which encompasses both SparkContext and SQL-related functionalities.
* **Responsibilities**:
  + Combines SparkContext, SQLContext, and HiveContext into a single interface.
  + Facilitates working with DataFrames and Datasets, which are higher-level abstractions built on top of RDDs.
  + Provides a way to work with Spark SQL and DataFrame API in a more convenient manner.
* **Usage**: Recommended for most applications, especially when working with structured data or using DataFrames.

**Key Differences**

1. **Level of Abstraction**:
   * SparkContext is lower-level, focusing on RDDs.
   * SparkSession is higher-level, providing access to DataFrames, Datasets, and SQL functionality.
2. **Creation**:
   * SparkContext is created using SparkConf, while SparkSession can be created using SparkSession.builder.
3. **Flexibility**:
   * SparkSession allows for easier integration with various data sources and built-in functions for structured data.

**Example**

Here’s a simple example to illustrate how to create both:

from pyspark.sql import SparkSession

# Creating a SparkSession

spark = SparkSession.builder \

.appName("Example App") \

.getOrCreate()

# Accessing SparkContext from SparkSession

sc = spark.sparkContext

# Working with DataFrames

df = spark.read.csv("data.csv")

df.show()