Data Engineering in the Cloud

Apache Spark - Part I

Xuemao Zhang
East Stroudsburg University

January 18, 2025

Outline

- Introduction to Python
- Introduction to Scala
- Apache Spark
- Data Frame

Python

Python is a high-level, general-purpose programming language.

- Widely used in various domains such as web development, automation, scientific computing, data analysis, and machine learning.
- Rich Ecosystem of Libraries:
  - Data Manipulation: Libraries like Pandas and NumPy for efficient data handling.
  - Data Visualization: Tools like Matplotlib, Seaborn, and Plotly for creating insightful visualizations.
  - Machine Learning: Comprehensive libraries such as Scikit-Learn, TensorFlow, and PyTorch for building and deploying machine learning models.

Python

- You **do not** need to **download** and install python on your computer - We'll use cloud computing Google Colaboratory https://colab.research.google.com/notebooks/
- Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.
- Variable names are case-sensitive.

```
a = 4
A = 40
print(a)
```

```
## 4
```

```
print(A)
```

```
## 40
```

Python

- The Python `print()` function is often used to output variables.
- In the `print()` function, you output multiple variables, separated by a comma:

```
x= y= z = 4
```

```
print(x, y, z)
```

```
## 4 4 4
```

```
# f-string
print(f"The value of variable x is {x}.")
```

```
## The value of variable x is 4.
```

- We use function `type()` to check data type

```
type(x)
```

```
## <class 'int'>
```

Python

- A Python **list** is an *ordered mutable/changeable* array. Lists *allow duplicate elements* regardless of their type.
- Create a list in Python by using square brackets, separating individual elements with a comma.
    - Each element can be of any data type.
- All lists in Python are *zero-based indexed*. You can access individual list elements.

```
A = [1, 2, 3, 3.4]
print(A[0],A[1],A[2],A[3])
```

```
## 1 2 3 3.4
```

```
print(A[0:2]) #stop=2; the end element is not included
```

```
## [1, 2]
```

Python functions

- A function takes a list of argument values, performs a computation with those values, and returns a single result. Python gives you many built-in functions.
    - see Python Built-in Functions https://docs.python.org/3/library/functions.html
- We can also create our own functions. These functions are called user-defined functions.
- Functions are declared using the **def** keyword, and the value produced is returned using the **return** keyword. Consider a simple function which returns the square of the input, $y = x^2$.
    - colon : is used to represent an indented block. It is not for slicing.
    - Python uses **indentation** to indicate a block of code.
    - The number of spaces is up to you as a programmer, but it has to be at least one.

Python functions

```python
def square(x):
  return x**2
```

```python
x = 2
y = square(x) # Call the function
print(x,y)
```

```
## 2 4
```

Scala

- Scala is a strong statically typed high-level general-purpose programming language.
  - Statically typed, meaning types are checked at compile-time.
  - It is designed to be a scalable language, combining functional and object-oriented programming in a concise, high-performance language.
- Scala is object-oriented, and uses a syntax termed curly-brace which is similar to the language C.
- Typically faster than Python because it compiles to JVM bytecode and runs on the Java Virtual Machine (JVM).
  - It is often used for scalable systems, big data processing (e.g., with Apache Spark), and back-end services.

Scala

- To avoid installation, we use replit (https://replit.com/) to run Scala code.
- Sign up an account and then **Create a new repl** with selecting Scala as the language.
- In the shell, type `scala`
- At the prompt `scala>`, type `println("Hello, Scala!")`

Scala basic syntax

- Case Sensitivity - Scala is case-sensitive
- Class Names - For all class names, the first letter should be in Upper Case.
- Method Names - All method names should start with a Lower Case letter.
- Program File Name - Name of the program file should exactly match the object name. When saving the file you should save it using the object name and append `.scala` to the end of the name.
- `def main(args: Array[String])` - Scala program processing starts from the main() method which is a mandatory part of every Scala Program.

Scala variables

- Data types see the documentation
- Variable declaration: declared using the keyword `val`
- In the following, `String` is the data type

```
val var1: String = "Foo"
```

```
println(var1)
```

- Variable type inference when an initial value is assigned

```
val var2 = 10
val var3 = "Hello, Scala!"
```

```
println(var3.getClass)
```

Scala variables

- Use print() when you want to output multiple pieces of data on the same line
- Use println() when you want to display each piece of output on a new line.

```
print(var1,var2,var3)
```

Scala functions

- A Scala function definition has the form

```
def functionName ([list of parameters with data types]) : [return ty
   function body
   return [expr]
}
```

```
def square(x: Double): Double= {
val result= x * x
return result
}
```

- or use a shorter format

```
def square(x: Double): Double = x * x
```

Scala functions

- Usage

```
println(square(2))

val number: Double = 5.1
val result: Double = square(number)
println(s"The square of $number is $result")
```

- `s` string interpolator above allows embedding variables and expressions inside strings.
- `f` Interpolator: Allows formatted strings using format specifiers, similar to `printf` in other languages.

```
println(f"The square of $number is $result%.3f")
```

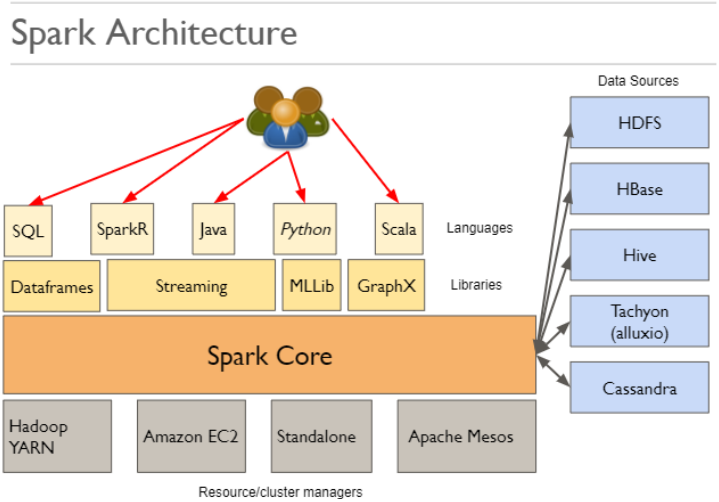
Apache Spark

- Apache Spark is an open-source unified analytics engine for large-scale data processing.
    - Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance.
    - Apache Spark is an **in-memory cluster computing framework** designed to handle a wide range of big data workloads.
    - Spark was developed to overcome some limitations of *Hadoop*'s *MapReduce*, such as slow processing speed and the complexity of writing MapReduce jobs.
    - Spark can run on Hadoop YARN and use HDFS for storage.

Apache Spark key features

- In-Memory Computing: Spark performs computations in memory, improving the speed of data processing.
- Cluster Computing Framework: Designed to handle big data workloads across distributed clusters.
- Wide Range of Workloads: Supports various data processing tasks, including:
    - Batch Processing
    - Interactive Queries
    - Streaming Data
    - Machine Learning
    - Graph Processing

Components of Apache Spark

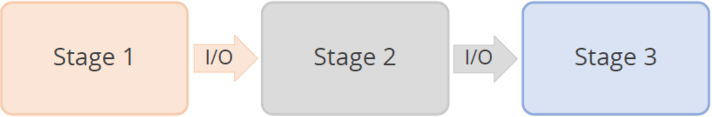


Core Components of Apache Spark

- Spark Core: The foundation of the Spark framework, providing basic functionality such as task scheduling, memory management, and fault recovery.
- Spark SQL: Allows querying of structured data via SQL, integrating seamlessly with Apache Hive.
- Spark Streaming: Enables real-time data processing and streaming analytics.
- MLlib: A scalable machine learning library offering a variety of algorithms and utilities.
- GraphX: A library for graph processing and analysis.

Spark RDDs

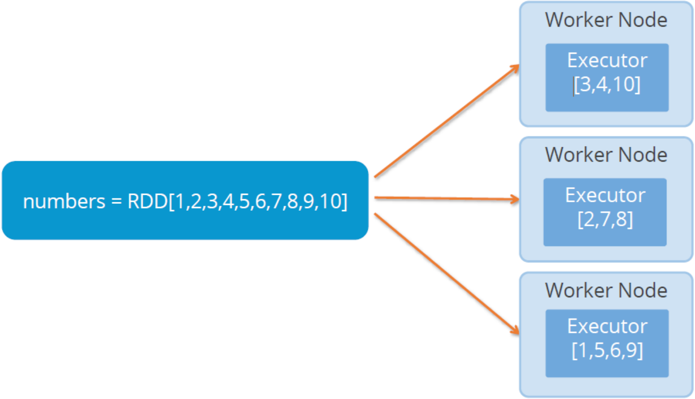
- The processing in distributed computing occurs in numerous phases, with the output of the first stage becoming input to the second stage and the data being shared between the two.



- This kind of processing involves a lot of input output overhead and makes the overall computation slower.
  - Reduction of the input output operations that slow the computation can be accomplished through in memory data sharing.
  - The data transferred between in memory is 10-100 times faster than data sharing through a network or a disc.

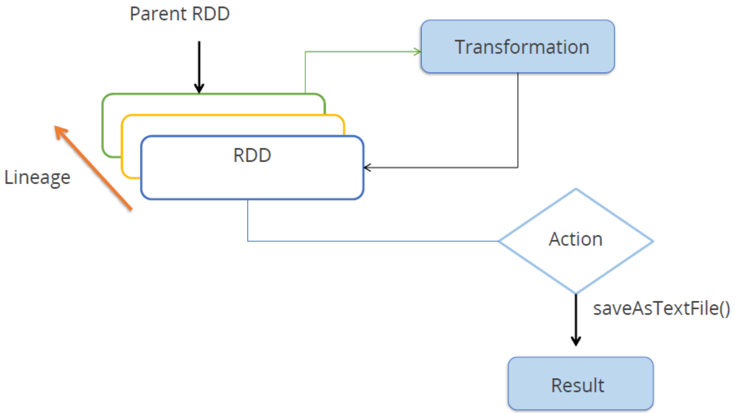
Spark RDDs

- The ideal soluation is RDDs which are core concept in Spark.
  - RDD stands for Resilient Distributed Dataset.
  - RDD enables fault tolerant, distributed in memory computations.
- An RDD is a collection of objects that are distributed across nodes in a cluster. RDD supports lazy evaluation.



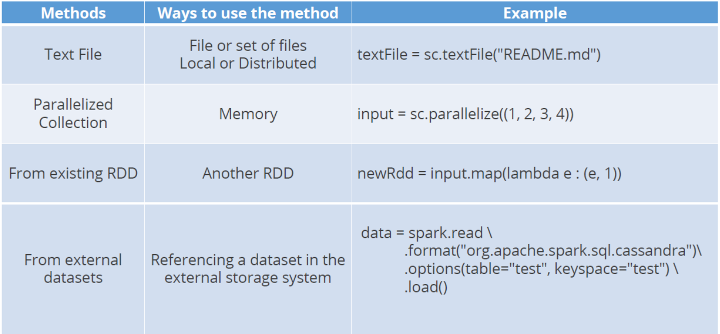
Spark RDDs

- RDD supports lazy evaluation. Lazy evaluation in Spark is a mechanism where the execution of an action will not begin until the action is initiated. Lazy evaluation occurs during Spark transformation.
  - Lazy evaluation occurs during Spark transformation.



Spark RDDs

- Different Ways to Create Spark RDD

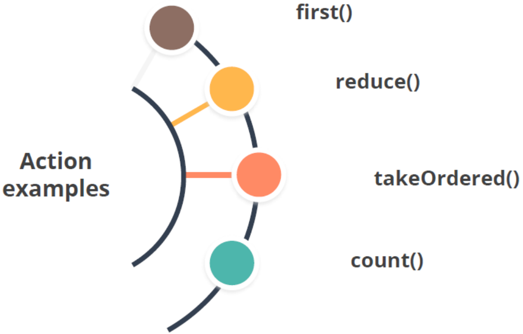


Methods	Ways to use the method	Example
Text File	File or set of files Local or Distributed	textFile = sc.textFile("README.md")
Parallelized Collection	Memory	input = sc.parallelize((1, 2, 3, 4))
From existing RDD	Another RDD	newRdd = input.map(lambda e : (e, 1))

From external datasets	Referencing a dataset in the external storage system	data = spark.read \\
		.format("org.apache.spark.sql.cassandra")\\
		.options(table="test", keyspace="test") \\
		.load()

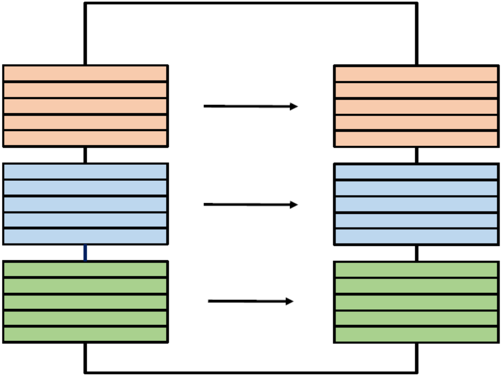
RDD operations

- Once an RDD is created, two types of operations can be performed
  - Actions: Act on the entire RDD
  - Transformations: Create a new RDD from an existing RDD by applying a certain logic
- Actions is an RDD operation that instructs Spark to perform computations and return the results back to the driver. I t will be executed only when an action is encountered.



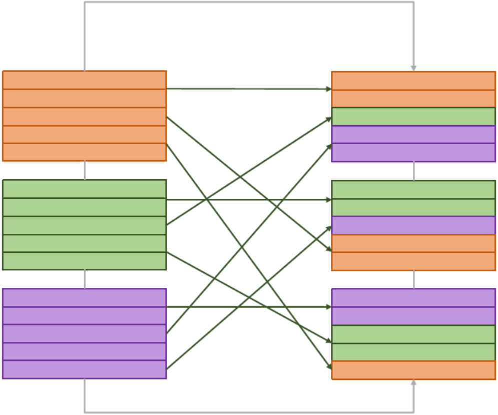
RDD operations

- There are two types of transformation:
  - Narrow Transformation
  - Wide Transformation
- Narrow transformations are for data from a single partition only.
  - Narrow transformations include map(), mapPartition(), flatMap (), filter(), and union() functions.



RDD operations

- Wide transformation are for data that can be from multiple partitions.
    - It is the result of GroupByKey() and ReduceByKey() functions
    - Wide transformations include groupByKey(), aggregateByKey(), aggregate(), join(), and repartition()



Resources for Learning Apache Spark

- Apache Spark Documentation https://spark.apache.org/documentation.html
- Book Learning Spark Lightning-Fast Big Data Analysis https://www.oreilly.com/library/view/learning-spark/9781449359034/
- Coursera Big Data Analysis with Scala and Spark

PySpark

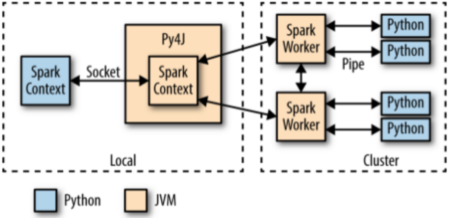
- Spark provides an interactive environment where you can learn Spark quickly.
  - PySpark Shell: Use Python to interact with Spark, making it accessible for Python users.
  - Scala Shell: Use Scala, Spark's native language, for direct interaction with Spark.

PySpark

- PySpark is the **Python API** for Apache Spark, allowing Python developers to harness the power of Spark's distributed computing capabilities.
  - By importing the `pyspark` library. Developers can then use its functions and classes to work with Apache Spark.
- An API, or Application Programming Interface, is a set of rules and protocols for building and interacting with software applications.
  - Web APIs: Accessed via HTTP, commonly used for web services (REST, SOAP).
  - Library APIs: Provided by software libraries, allowing applications to use predefined functions (e.g., Math libraries).
  - Operating System APIs: Allow applications to interact with the OS (e.g., Windows API).
  - Database APIs: Enable interaction with databases (e.g., SQL queries).

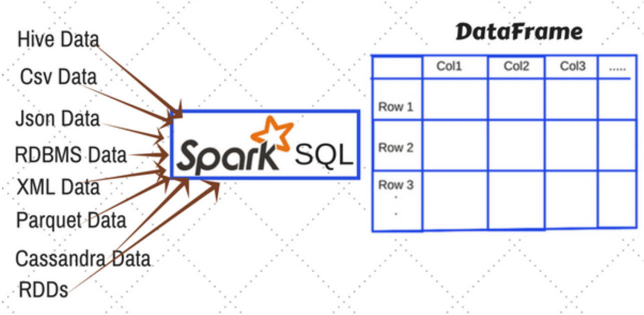
PySpark

- The Pyspark communicates with Spark using Py4J API



Data Frame

- Data Frame is a distributed collection of structured data. It is conceptually similar to a table in a relational database.
  - Data is distributed across multiple nodes in a cluster.
  - Data is organized into named columns.
  - Supports a wide range of data types (numeric, string, date, etc.).
- Spark SQL is one of the modules in the Spark ecosystem for structured data.
  - Allows users to query structured data using SQL as well as the DataFrame API.
  - Spark SQL provides different API's: DataFrame, RDD(Scala/Java), SQL



Hive Data
Csv Data
Json Data
RDBMS Data
XML Data
Parquet Data
Cassandra Data
RDDs

Spark SQL

DataFrame

	Col1	Col2	Col3	.....
Row 1				
Row 2				
Row 3 . .				

License