SE ZG522: Big Data Systems

14 Spark Introd^n & Programming

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

May 08, 2022 ONLINE (WILP @ BITS-Pilani Jan-May 2022)

http://ktiwari.in/bds

Apache Spark

- Unified engine for large-scale data analytics. Started in 2009
- Fast cluster computing framework for processing, querying and analyzing Big data.
- In-memory computation. Open source. 100 times faster then Hadoop (map/reduce). Spark is designed to run on top of Hadoop.
- **Batch/streaming data:** Unify the data processing in batches and real-time streaming. Use Python, SQL, Scala, Java or R.
- **SQL Analytics:** Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting.
- **Data Science at Scale:** Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling
- **Machine Learning:** Train ML models on laptop and use it to scale to fault-tolerant clusters of thousands of machines.

Spark Features

1. **Fast Processing:** user Resilient Distributed Datasets (RDD) which saves time in reading and writing
2. **In-mempry computing:** data is in RAM, so quicker access.
3. **Flexible:** supports multiple languages like Python, SQL, Scala, Java or R

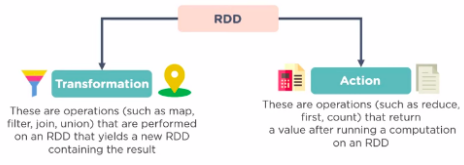
```
$ pip install pyspark
$ pyspark
df = spark.read.csv("accounts.csv", header=True)
filteredDf = df.select("AccountBalance", "CountOfDependents").filter("AccountBalance > 0")
filteredDf.summary().show()
```

4. **Fault Tolerance:** RDDs are designed to handle failure of worker node
5. **Better Analytics:** has rich set of libraries for ML and analytics.

Spark Components

Spark Core

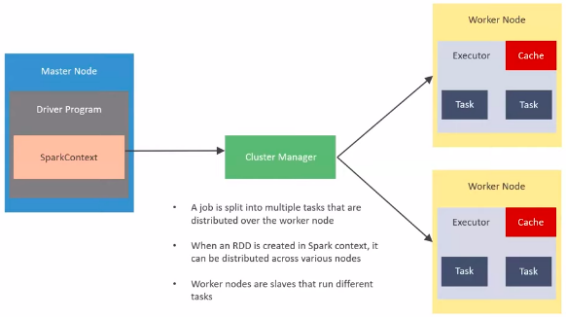
- Responsible for
  - Memory managemnt
  - Fault recovery
  - Scheduling distributing and monitering jobs
  - Interacting with storage systems
- Resilient Distributed Datasets (RDD) it is logical only.




```
val x = SC.textfile(.....) file could be in hodoop
val y = x.map(....)
val z = y.filter(...)
```

```
z.count
z.count
```


Architecture



Master Slave. Driver provides SparkContext, a gateway to functionality

Example on Jupyter notebook [1]

```
pip install pyspark
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('kt1').getOrCreate()
ktDF = spark.read.option('header','true').csv('f1.csv',InferSchema=True)
type(ktDF), ktDF.head(3), ktDF.printSchema(), ktDF.columns, ktDF.dtypes, ktDF.discribe().show()
ktDF.select('name').show(),ktDF.select(['name','sal']).show()
ktDF.select(['name','sal']).show()
ktDF.withColumn('salNew', ktDF['sal']+500).show()
ktDF.drop('salNew').show()
ktDF.withColumnRename('sal','salNew').show()
ktDF.na.drop(how='all/any',thresh=3).show()
ktDF.na.fill('31').show()
ktDF.filter( (ktDF['sal']<=30000 &| ~ ktDF['sal']>=20000)).show()
ktDF.groupBy('name').sum().show(), max, min, mean, avg, median, count
```

[1] https://spark.apache.org/

Thank You!

Thank you very much for your attention!