BigData and Machine Learning with Hadoop and Spark Frameworks

Jean-Marc GRATIEN[1]

[1] Department of Computer Science
IFP New Energy

January 25th 2021 / Master Data-AI

Outline I

Outline II

4 Dask
- Introduction
- Architecture
- Dask API
- Dask Ecosystem

5 TP
- TP0 : Hadoop Installation
- TP1 : Hadoop World count
- TP2 : Start with Spark
- TP4 : Spark ML, Data processing
- TP5 : Spark ML, Machine learning
- TP6 : Spark ML, Image processing
- TP7 : Start with Dask

Objectifs

Objectifs

- General Overview on Hadoop and Spark
- Introduce to Hadoop
- Introduction to Spark Framework

Audience and Prerequisites

- Audience : computer science and data scientist students
- Prerequisites :
  - sequential programming in java and python
  - elementary of machine learning, data analytics
  - image processing
- Material(Slide+TPs) available at :
  git clone `https://github.com/jgratien/BigDataHadoopSparkDaskCourse.git`

Motivation

Introduction to Bigdata

BigData

- What is Bigdata?
- What are the BigData issues?

Outline

Hadoop Framework

Introduction to Hadoop

- Hadoop definition
  - Java opensource software framework
  - Data storage management
  - Parallel data analysis
  - part of Apache project supported by the Apache Software Foundation

Hadoop Framework

Introduction to Hadoop

- Hadoop History
  - 1990 - 2000 : World Wide Web
  - Yahoo, AltaVisa,. . . : first search engines
  - Nutch open source project created by Doug Cutting and Mike Cafarella
  - 2006 : Nutch project is split:
    the distributed storage and computing framework -> Hadoop
  - 2008 : Hadoop 1.0 (Open Source Project proposed by Yahoo)
  - 2012 : Hadoop 2.0 release
  - 2017 : Hadoop 3.0

Hadoop Framework

Introduction to Hadoop

Why Hadoop?

- BigData issues :
  - increasing amount of data amount
  - distributed storage facilities
  - parallel data processing management
  - fault tolerance management

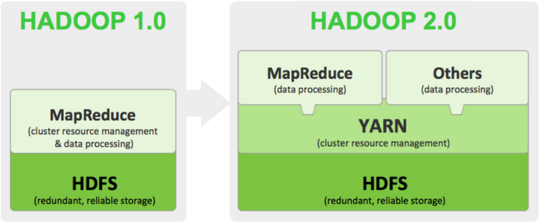
Outline

Hadoop Framework

Hadoop Framework Architecture

the basic Hadoop Framework based of 4 main modules :

- HDFS : Hadoop Distributed File System
- YARN : Yest Another Ressource Negotiator
- Map : parallel data processing
- Reduce : collecting data and producing results



Hadoop Framework

Hadoop Ecosystem

Hadoop ecosystem :

- Ambari : Hadoop component and services web interface management
- Cassandra : Distributed Data Base system
- Flume : Data Stream management layer
- HBase : NoSql distributed Data Base
- HCatalog : data storage management
- Hive : data storage with a SQL API
- Oozie : task framework
- Pig : HDFS data processing framework
- Solr : data indexing framework
- Sqoop : SQL DB and Hadoop data transfer framework
- Zookeeper : distributed data processing management

Hadoop Framework

Hadoop distributions

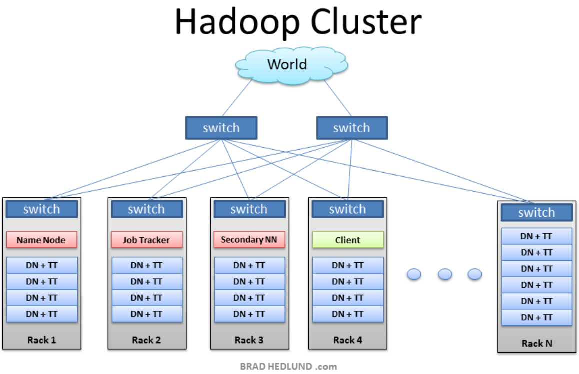
Hadoop Distributions :

- Hortonworks
- Cloudera
- MAPR

Outline

Hadoop Framework

Hadoop Cluster



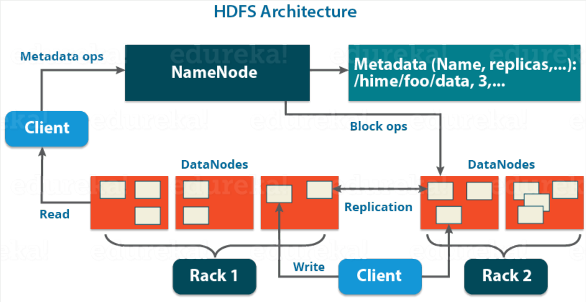
Purpose : Scalability, Fault tolerance and Reliality management

Hadoop Framework

Hadoop Cluster

Concepts

- NameNode
- DataNode
- Replication
- Block, Blocksize



HDFS : Hadoop Distributed Filesystem

Hadoop Framework: HDFS

HDFS commands

HDFS commands :

Starting HDFS

```
# Format nodes
> hadoop namenode -format

# Starting HDFS services
> start-dfs.sh
```

Shutting down HDFS

```
# Stopping HDFS services
> stop-dfs.sh
```

Hadoop Framework: HDFS

HDFS commands

HDFS commands :

Inserting Data into HDFS

```
# Step 1 : Create input directory
> $HADOOP_HOME/bin/hadoop fs -mkdir /usr/input

# Step 2 : copy data from local filesystem to hdfs
    filesystem
> $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /
    user/input

# Step 3 : check results with ls cmd
> $HADOOP_HOME/bin/hadoop fs -ls /usr/input
```

Hadoop Framework: HDFS

HDFS commands

HDFS commands :

Retreiving Data from HDFS

```
# Step 1 : view data
> $HADOOP_HOME/bin/hadoop fs -cat /user/outputfile

# Step 2 : get data from hdfs filesystem to local
    filesystem
> $HADOOP_HOME/bin/hadoop fs -get /user/output/ /
    home/hadoop_out

# Step 3 : check results with ls cmd
> $HADOOP_HOME/bin/hadoop fs -mkdir /usr/input
```

Hadoop Framework: HDFS

HDFS commands list

Commande name	Description
fs -help <cmd-name>	return cmd usage
fs -ls <path>	list <path> directory contents
fs -lsr <path>	ls ,recursively with sub dirs
fs -du <path>	show disk usage in bytes
fs -dus <path>	show disk usage in bytes and summary
fs -test [ezd] <path>	return 1 if path exists; has 0 length; or is a directory, otherwize 0
fs -cat <filename>	
fs -tail [-f] <filename>	

Hadoop Framework: HDFS

HDFS commands list

Commande name	Description
fs -mv <src><dest>	move file or directory within HDFS
fs -cp <src> <dest>	copy file or directory within HDFS
fs -rm <path>	remove file or directory within HDFS
fs -rmr <path>	rm recursively
fs -put <localSrc> <dest>	copy files or dirs from local FS to HDFS

Hadoop Framework: HDFS

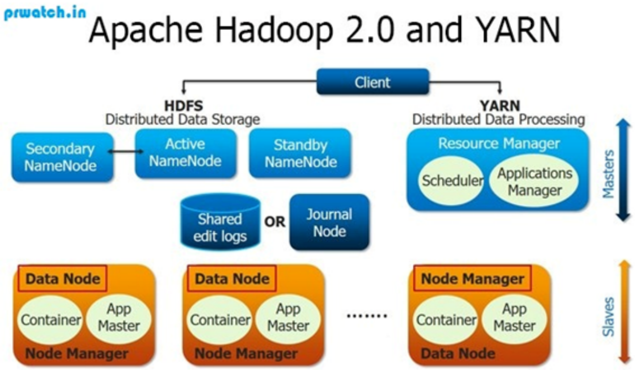
HDFS commands list

Commande name	Description
fs -copyFromLocal <localSrc> <dest>	identical to put
fs -moveFromLocal <localSrc> <dest>	move file or dirs from local FS to HDFS
fs -get [-crc] <src> <localDest>	copy file or dirs from HDFS to local FS
fs -getmerge [-crc] <src> <localDest>	copy all files from HDFS and merge to a single file in FS
fs -copyToLocal <localSrc> <dest>	copy file or dirs from HDFS to local FS
fs -moveToLocal <localSrc> <dest>	move file or dirs from HDFS to local FS
fs -mkdir <path>	create directory in HDFS

Outline

Hadoop Framework

YARN



Hadoop Framework

YARN

Starting YARN

```
# Starting YARN services
> start-yarn.sh
```

Shutting down YARN

```
# Stopping YARN services
> stop-yarn.sh
```

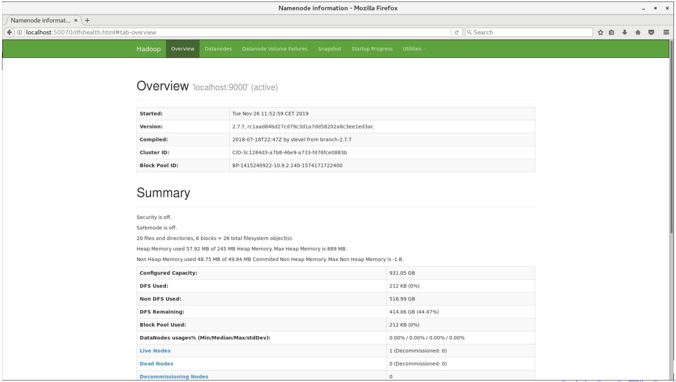
Hadoop Web tools

Hadoop Web Tools

Web tools on : `http://<hostname>:<port>`

<hostname>:<port> (default localhost:50070) are defined in hdfs-site.xml

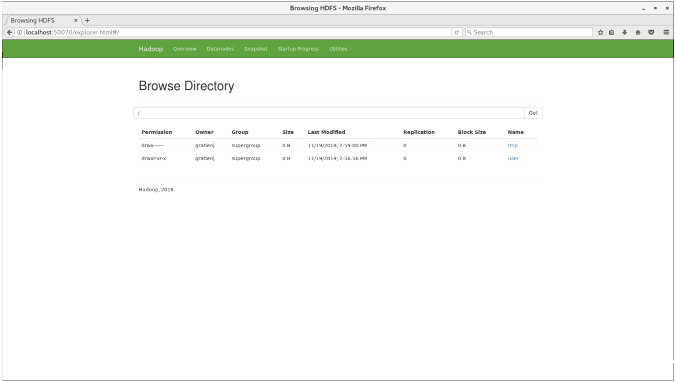
Overview web page:



Hadoop Web tools

Hadoop Web Tools

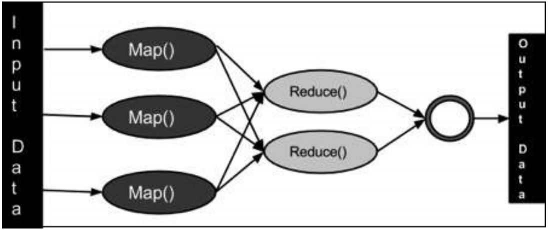
File browser web page:



Outline

Hadoop Framework

MapReduce Framework



MapReduce Framework

Hadoop Framework

MapReduce Framework

MapReduce Algorithm :

- Programming model ;
- Two stages:
  - Map stage :
    - Mapper jobs;
    - data are processed in parallel by mapper jobs;
  - Reduce Stage :
    - Reducer jobs;
    - mapper output data are processed Reducer jobs;
    - Reducer jobs produce new set of output stored in HDFS.

Hadoop Framework

MapReduce Framework

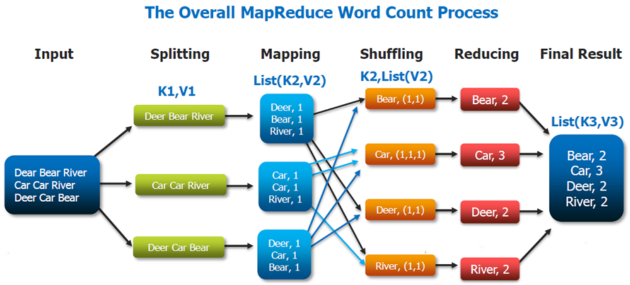
Input and Output :

- Input : <key,value> pairs
  - key and values classes must implement Writable Interface;
  - key class have to implemente WritableComparable Interface;
- Job : (Input)$\rightarrow$ map $\rightarrow$ <k2,v2> $\rightarrow$ reduce $\rightarrow$ <k3,v3>(Output)

	Input	Output
Map	<k1,v1>	list(<k2,v2>)
Reduce	<k2,v2>	list(<k3,v3>)

Hadoop Framework

MapReduce Framework



Hadoop Framework

Java exemple

WordCount Java class

```
import org.apache.hadoop.*;
public class WordCount  {
  public static class  TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
    public void map(Object key, Text value, Context context)
                    throws IOException, InterruptedException {
        ... ;
    }
  }
  public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable>  {
    public void reduce(Text key, Iterable<IntWritable> values,Context context)
                      throws IOException, InterruptedException  {
         ... ;
    }
  }
  public static void main(String[] args) throws Exception {
  ...;
  }
}
```

Hadoop Framework

Java exemple

Word Count Mapper Java class

```java
public class WordCount
{
  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>
  {
    public void map(Object key, Text value, Context context)
                    throws IOException, InterruptedException
    {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
}
```

Hadoop Framework

Java exemple

WordCount Reducer Java class

```
public class WordCount
{
   public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable>
   {
        private IntWritable result = new IntWritable();
        void reduce(Text key, Iterable<IntWritable> values,Context context)
                   throws IOException, InterruptedException
        {
           int sum = 0;
           for (IntWritable val : values) {
              sum += val.get();
           }
           result.set(sum);
           context.write(key, result);
        }
   }
}
```

Hadoop Framework

Java exemple

Main test function

```
public class WordCount
{
  public static void main(String[] args) throws Exception
  {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Hadoop Framework

Java exemple

Suppose we have two test files file01 and file02 in current directory

Prepare Test Data

```
// Two test file file01 file02 in current directory
$ ls
   file01
   file02
$ hdfs dfs -put file01 /user/gratienj/input
$ hdfs dfs -cat /user/gratienj/input/file01
   Hello World Bye World
$ hdfs dfs -put file02 /user/gratienj/input
$ hdfs dfs -cat /user/gratienj/input/file02
  Hello Hadoop Goodbye Hadoop
```

Hadoop Framework

Java exemple

Suppose the Java Project is compiled and generates the jar file BigDataTP1.jar

Run application

```
$ hadoop jar BigDataTP1.jar hadoop.WordCount /user/gratienj/input /user/gratienj/output
```

Check results

```
$ hdfs dfs -cat /user/gratienj/output/part-r-00000
Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2
```

Hadoop Framework

Python example

Python example with Hadoop Streaming

Mapper python script

```python
#!/usr/bin/env python
"""mapper.py"""
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print('%s\t%s' % (word, 1))
```

Hadoop Framework

Python example

Reducer python script Part 1

```
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
```

Hadoop Framework

Python example

Reducer python script Part 2

```python
for line in sys.stdin: # input comes from STDIN
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print('%s\t%s' % (current_word, current_count)) # write result to STDOUT
        current_count = count
        current_word = word
if current_word == word: # do not forget to output the last word if needed!
    print('%s\t%s' % (current_word, current_count))
```

Hadoop Framework

Python example

Python example with Hadoop Streaming : Part 1

Copy test files on HDFS

```
$ hdfs dfs -copyFromLocal /home/gratienj/test/books /user/gratienj/input/books
```

Run application

```
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.7.7.jar \
 -file /home/hduser/mapper.py    -mapper /home/hduser/mapper.py \
 -file /home/hduser/reducer.py   -reducer /home/hduser/reducer.py \
 -input /user/gratienj/input/books/* -output /user/gratienj/books-output
```

Hadoop Framework

Python example

Python example with Hadoop Streaming : Part 2

Check results

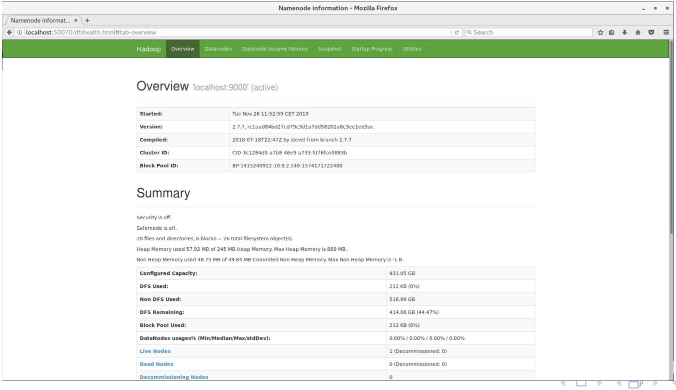
```
$ hdfs dfs -ls /user/gratienj/books-ouput
Found 1 items
/user/gratienj/books-output/part-00000
$ hdfs dfs -cat /user/gratienj/books-output/part-00000
```

Hadoop Web tools

Ambari Server Tools

Ambari Server : tools to manage and monitor applications for Apache Hadoop

Web page : `http://<ambari-server-hostname>:8080`



Outline

Spark Framework

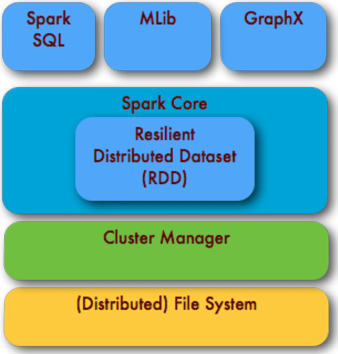
Introduction to Spark

- Spark : Big Data framework for data processing
- History
  - 2009 : AMPLab, UC Berkeley University
  - 2010 : Open source as an Apache project
- Complete and Unified framework
  - Hadoop (MapReduce)
  - Storm (Streaming)
  - Languages : Java, Scala, Python
  - SQL

Outline

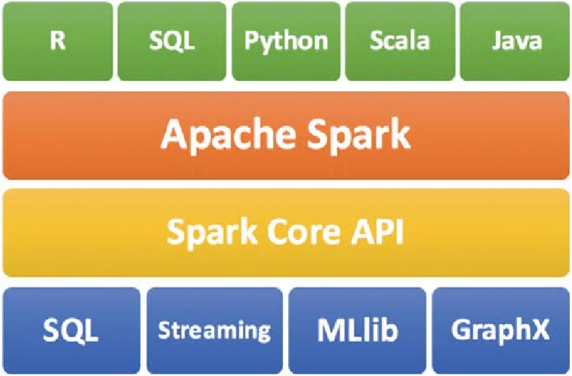
Spark Framework

Apache Spark Architecture



Spark Framework

Apache Spark Ecosystem



Outline

Spark Framework

Spark Core : Spark configuration

Spark Cluster Configurations :

- Local mode
- Cluster mode
- Client mode

Spark parallel concepts:

- multiple executors (private JVM)
- multiple cores per executor

Spark Framework

Spark Core : Spark configuration

Configuring a SparkContext

```python
import pyspark
from pyspark import SparkConf
sc_conf = SparkConf()
sc_conf.setAppName(app_name)
sc_conf.setMaster('local[*]')
sc_conf.set('spark.executor.memory', '4g')
sc_conf.set('spark.executor.cores', nb_cores)
sc_conf.set('spark.driver.memory', '16G')
sc_conf.set('spark.cores.max', '32')
sc_conf.set('spark.driver.maxResultSize', '10G')
sc_conf.set('spark.logConf', True)
```

Spark Framework

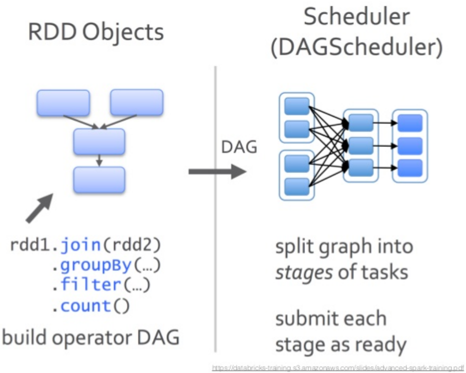
Spark Core : Spark context

Create a SparkContext

```
import pyspark
from pyspark import SparkContext
sc =SparkContext()
```

Spark Framework

Spark Core : Data concepts



Spark Framework

Spark Core : Data concepts

Spark Data concepts :

- **RDD** : Resilient Distributed Data, list of <key,value>
- **Transformations** : apply lambda to creating new RDDs
- **DAG** : pipeline of transformation
- **Actions** : operations on the RDD producing results
- **Scheduler** : perform actions on DAG
- **Stage** : parallel operations
- **Pipeline** : sequence of stages

Spark Framework

Spark Core : RDD

Create a Spark RDD

```python
import pyspark
from pyspark import SparkContext
sc =SparkContext()

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

nums.take(1)
```

Output

```
[1]
```

Spark Framework

Spark Core : RDD Transformationq and Actions

Spark RDD transformations and Actions

```
sc =SparkContext()
nums= sc.parallelize([1,2,3,4])
squared = nums.map(lambda x: x*x).collect()
for num in squared:
    print('%i ' % (num))
```

Output

```
1
4
9
16
```

Spark Framework

Spark Core : RDD Transformations and Actions

Transformation :

- apply lambda function to RDD
- create a new RDD
- lazy evaluation
- create a DAG

Spark Framework

Spark Core : RDD Transformations and Actions

Examples :

Commande name	Description
map() flatMap() mapPartition	apply to each RDD line apply to all RDD elements apply per partition
filter() groupBy() groupByKey() reduceByKey()	apply to a selection of lines create new set of (key,value)
sample() union() join()	selection of lines fusion of two RDDs union without duplicate keys

Spark Framework

Spark Core : RDD Transformations and Actions

Actions :

- get results on a pipeline of transformations
- perform all the transformation
- real evaluation

Examples :

Commande name	Description
getNumPartition() reduce() collect()	 apply lambda to all elements create a collection
count() max(), min() sum()	count elements stats

Spark Framework

Spark SQL : DataFrame

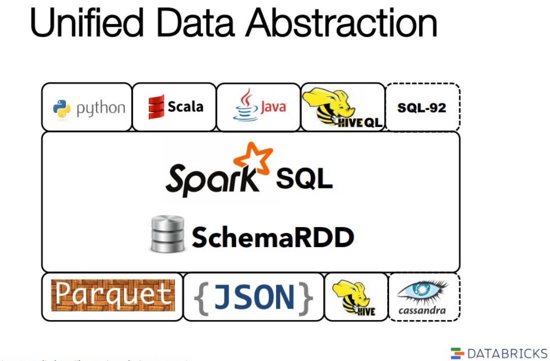


Image credit: http://barrymieny.deviantart.com/

Spark Framework

Spark Core : RDD

Create a Spark SQL context

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

Create a DataFrame

```
list_p=[('John',19),('Smith',29),('Adam',35)]
rdd = sc.parallelize(list_p)
ppl_rdd=rdd.map(lambda x: Row(name=x[0], age=int(x
    [1])))
ppl_df_rdd = sqlContext.createDataFrame(ppl_rdd)
```

Spark Framework

Spark SQL : DataFrame

Print DataFrame Schema

```
DF_ppl.printSchema()
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

Spark Framework

Spark SQL : DataFrame

Print DataFrame Schema

```
df = sqlContext.read.csv(SparkFiles.get("adult_data.csv"), header=True, inferSchema= True)
df_string.printSchema()
root
 |-- age: string (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: string (nullable = true)
 |-- education: string (nullable = true)
 |-- education_num: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital_gain: string (nullable = true)
 |-- capital_loss: string (nullable = true)
 |-- hours_week: string (nullable = true)
 |-- native_country: string (nullable = true)
 |-- label: string (nullable = true)
```

Spark Framework

Spark SQL : DataFrame

Select columns

```
df.select('age','fnlwgt')
    .show(5)
```

Select columns

```
+---+------+
|age|fnlwgt|
+---+------+
| 39| 77516|
| 50| 83311|
| 38|215646|
| 53|234721|
| 28|338409|
+---+------+
only showing top 5 rows
```

Spark Framework

Spark SQL : DataFrame

Select columns

```
df.groupBy("education").
    count().sort("count",
    ascending=True).show()
```

Select columns

```
+------------+-----+
|   education|count|
+------------+-----+
|   Preschool|   51|
|     1st-4th|  168|
|     5th-6th|  333|
|   Doctorate|  413|
|        12th|  433|
|         9th|  514|
| Prof-school|  576|
|     7th-8th|  646|
|        10th|  933|
|  Assoc-acdm| 1067|
|        11th| 1175|
|   Assoc-voc| 1382|
|     Masters| 1723|
|   Bachelors| 5355|
|Some-college| 7291|
|     HS-grad|10501|
+------------+-----+
```

Spark Framework

Spark SQL : DataFrame

Describe data: describe() functions give a summary of statistics :

- count,
- mean,min,max
- standarddeviation

Describe

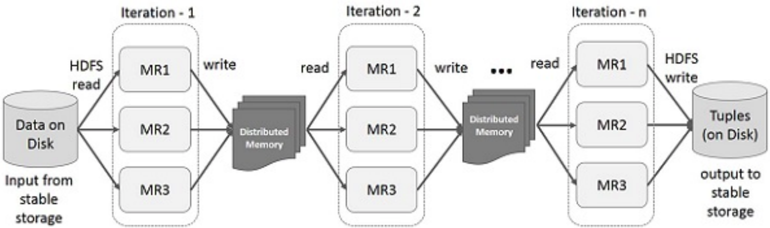
```
df.describe('capital_gain
    ').show()
```

Describe

```
+-------+-----------------+
|summary|     capital_gain|
+-------+-----------------+
|  count|            32561|
|   mean|1077.6488437087312|
| stddev| 7385.292084840354|
|    min|                0|
|    max|            99999|
+-------+-----------------+
```

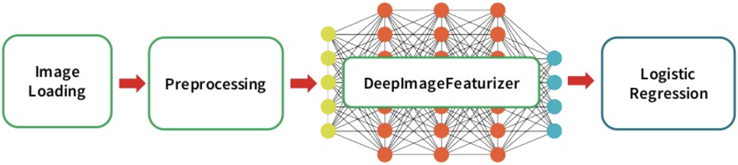
Spark Framework

Spark Mlib



Spark Framework

Spark Mlib



MLlib Pipeline

Spark Framework

Spark Mlib

Mlib provides tools for Machine learning

- set of classifier and regression algorithms
- create models from Spark Dataframe
- set of tools to evaluate the predicting models
- concept of pipeline to process Data

Spark Framework

Spark Mlib Pipeline

Spark Pipeline : a sequence of stages (Transformer, Estimator)

- String Indexer : convert Categorical Data to numerics
- Standard Scaler on Continuous Values
- VectorAssembler : features must be a dense vector

Spark Framework

Spark Mlib : Data processing

StringIndexer

```python
from pyspark.ml.feature import StringIndexer
df_rdd = ...
# Create String Indexer to convert "cat_key" to "
    ind_key"
string_indexer = StringIndexer(inputCol="cat_key",
    outputCol="encoded_key")
# Create model
model = string_indexer.fit(df_rdd)
# Transform RDD
encoded_df_rdd = model.transform(df_rdd)
```

Spark Framework

Spark Mlib : Data processing

OneHotEncoder

```
from pyspark.ml.feature import OneHotEncoder
encoder = OneHotEncoder(dropLast=False, inputCol="
    encoded_key", outputCol="vec_key")
vec_df_rdd = encoder.transform(encoded_df_rdd)
```

VectorAssembler

```
from pyspark.ml.feature import  VectorAssembler
assembler = VectorAssembler(inputCols=[key1,key2,
    key3], outputCol="features")
ass_df_rdd = assembler.transform(df_rdd)
```

Spark Framework

Spark Mlib : Data processing

Pipeline

```
from pyspark.ml import Pipeline
# DEFINE LIST OF STAGES
stages = [[label_indexer],[cat_key_indexer,encoder
    ],[assembler]]

# DEFINE PIPELINE
pipeline = Pipeline(stages=stages)

# APPLY PIPELINE
pipelineModel = pipeline.fit(df_rdd)
model_df_rdd = pipelineModel.transform(rdd_df)
```

Spark Framework

Spark ML : ML pipeline Part 1

create DataFrame

```
rdd.map(lambda x: (x["newlabel"], DenseVector(x["
    features"])))
sqlContext.createDataFrame(input_data, ["label", "
    features"])
```

Split data

```
randomSplit([.8,.2],seed=1234)
```

Spark Framework

Spark ML : ML pipeline Part 2

Train model

```
lr=LogisticRegression(labelCol="label",featuresCol=
    "features",maxIter=10, regParam=0.3)
lr.fit()
```

Make prediction

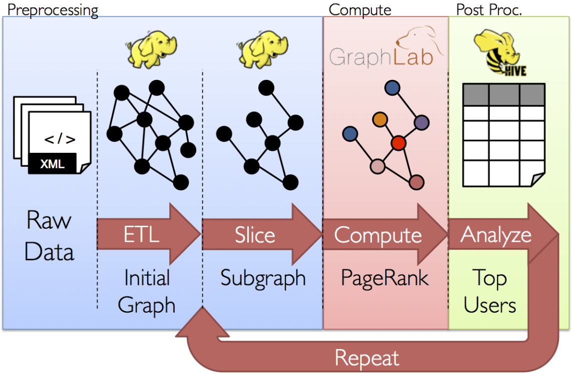
```
lr.transform()
```

Spark Framework

Spark Streaming

Spark Framework

Spark GraphX



Outline

Dask Framework

Introduction to Dask

- Dask : Open source python framework for data processing
- developped with community projects like : Numpy, Pandas, and Scikit-Learn
- supported by: Anaconda, CapitalOne, NSF, Nvidia,. . .
- High-level collections:
  - Array, Bag, and DataFrame collections
  - mimic NumPy, lists, and Pandas
  - operate datasets out of core memory
- Low-Level schedulers :
  - dynamic task schedulers
  - execute task graphs in parallel

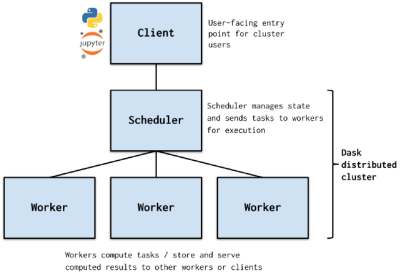
Outline

Dask Framework

Dask Architecture

Dask architecture:

- Dask Cluster
- Dask Scheduler
- Dask collections



Dask Framework

Dask Cluster

Various Dask cluster types:

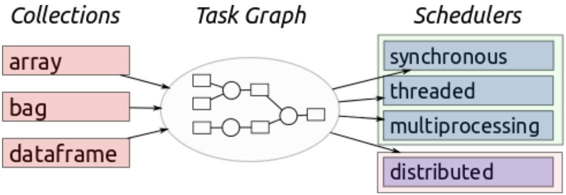
- Hadoop/Spark clusters running YARN
- HPC clusters running job managers like SLURM, SGE, PBS, LSF, or others common in academic and scientific labs
- Kubernetes clusters

Dask Framework

Dask Scheduler

Dask Scheduler:

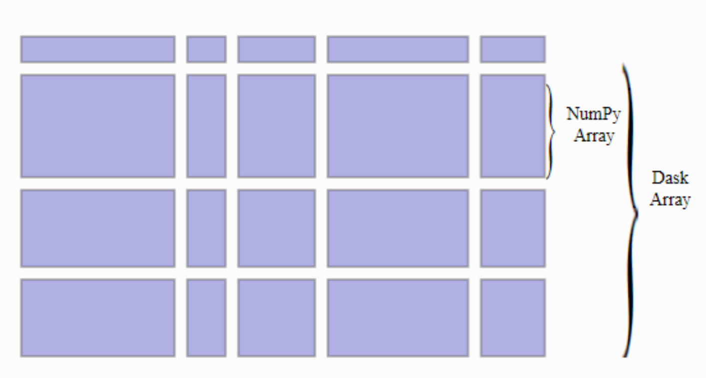
- Single machine scheduler : Optimized for larger-than-memory use. Simple, easy and cheap to use, but does not scale as it only runs on a single machine.
- Distributed scheduler : More sophisticated, fully asynchronous



Dask Framework

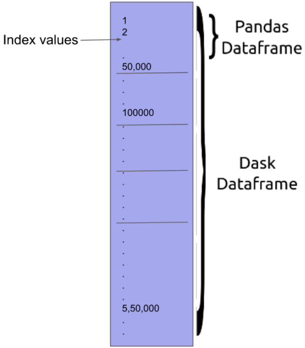
Dask collections

Dask Array



Dask Framework

Dask DataFrame



Dask DataFrame

Outline

Dask framework

Dask installation

Installation

```
conda install dask
```

OR

Installation

```
pip install dask[complete]
```

Dask framework

Dask starting cluster

Lauching Dask cluster

```
from dask.distributed import LocalCluster, Client
cluster = LocalCluster()
client = Client(cluster)
```

Dashboard usually on `http://localhost:8787/status`

Dashboard

```
#To see where the port of the dashboard is, use this command
print(client.scheduler_info()['services'])
# {'dashboard': 8787} --> means you can access it at localhost:8787
```

Dask framework

Dask Collections

Dask Bag

```
import dask.bag as db
b = db.from_sequence([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], npartitions=2)
```

Dask Array

```
import dask.array as da
x = da.random.random((10000, 10000), chunks=(1000, 1000))
```

Dask DataFrame

```
from dask import datasets
import dask.dataframe as dd
df = datasets.timeseries()
```

Dask framework

Dask Distributed

Dask Delayed function

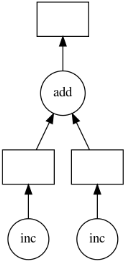
```python
from dask import delayed
@delayed
def inc(x):
    return x + 1
@delayed
def add(x, y):
    return x + y
```

Dask Lazy evaluation

```python
x = inc(15)
y = inc(30)
total = add(x, y)

#visualize DAG
total.visualize()

# execute all tasks
```



Dask framework

Dask Distributed

Dask Future

```python
from dask import delayed
def inc(x):
    return x + 1
def dec(x):
    return x - 1
def add(x, y):
    return x + y
```

Dask Lazy evaluation

```python
from dask.distributed import Client
c = Client(n_workers=4)

x = c.submit(inc, 1)
y = c.submit(dec, 2)
total = c.submit(add, x, y)
```

Dask framework

Dask Distributed

Dask evaluation

```python
# execute all tasks
total.compute()
```

Dask progess

```python
from dask.distributed import progress
# to show progress bar
progress(f)
```

Dask framework

Dask Distributed

Dask get results

```
# get result.
c.gather(f)
```

Dask persist

```
total.persist()
```

Dask framework

Dask Distributed

Dask asynchronism

```
from dask.distributed import as_completed
def func(x):
    ...
    return y
futures = [c.submit(func, x) for x in range(n)]
iterator = as_completed(futures)
for res in iterator:
    print("RES␣Y␣:",res.result())
```

Outline

Dask ecosystem

Ecosystem overview

Dask ecosystem overview :

- Dask tutorial :
  `https://github.com/dask/dask-tutorial.git`
- Collection : Bag, Array, DataFrame
- Data Storage : CSV, HDF5, . . .
- Machine Learning : Scikit-learn, XGBoost,. . .
- Cluster : Local, SSH, YARN, . . .

Dask Ecosystem

Dask Bag

Dask bag

```python
import dask.bag as db
b = db.from_sequence([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], npartitions=2)
```

Dask bag

```python
import os
b = db.read_text(os.path.join('data', 'accounts.*.json.gz'))
```

Dask bag

```python
import json
js = lines.map(json.loads)
```

Dask Ecosystem

Dask Array

Dask bag

```
import h5py
import os
f = h5py.File(os.path.join('data', 'random.hdf5'), mode='r')
dset = f['/x']
```

Dask bag : lazy creation

```
import dask.array as da
x = da.from_array(dset, chunks=(1_000_000,))
```

Dask bag : Numpy lazy API

```
result = x.sum()
print(result)
```

Dask bag : evaluation

Dask Ecosystem

Dask DataFrame

Dask DatFrame

```
import os
import dask
filename = os.path.join('data', 'accounts.*.csv')
```

Dask DataFrame : lazy creation

```
import dask.dataframe as dd
df = dd.read_csv(filename)

df = dd.read_csv(os.path.join('data', 'nycflights', '*.csv'),
                 parse_dates={'Date': [0, 1, 2]})
```

Dask DataFrame : lazy API

```
df.DepDelay.max().visualize()
```

Dask Ecosystem

Dask Cluster

Dask Local Cluster

```python
from dask.distributed import Client, LocalCluster
cluster = LocalCluster()
client = Client(cluster)
```

Dask YarnCluster

```python
from dask_yarn import YarnCluster
from dask.distributed import Client

# Create a cluster where each worker has two cores and eight GiB of memory
cluster = YarnCluster(environment='environment.tar.gz',
                      worker_vcores=2,
                      worker_memory="8GiB")
# Scale out to ten such workers
cluster.scale(10)

# Connect to the cluster
client = Client(cluster)
```

Dask Ecosystem

Dask Cluster Command line SetUp

Dask Scheduler and Workers SetUp

```
$ dask-scheduler
Scheduler at:   tcp://192.0.0.100:8786

$ dask-worker tcp://192.0.0.100:8786
Start worker at:  tcp://192.0.0.1:12345
Registered to:    tcp://192.0.0.100:8786

$ dask-worker tcp://192.0.0.100:8786
Start worker at:  tcp://192.0.0.2:40483
Registered to:    tcp://192.0.0.100:8786
```

Dask Client SetUp

```
from distributed import Client
client = Client('192.0.0.100:8786')
```

TPs

TPs require docker usage.
A Docker Quick Sheet can be found in :
`https://github.com/jgratien/BigDataHadoopSparkDaskCourse/tree/main/doc/Docker.md`

- TP 0 : Hadoop Installation
- TP 1 : WorldCount
- TP 2 : DataBase request
- TP 3 : Spark Installation
- TP 4 : Spark Compute PI
- TP 5 : Spark Image Processing
- TP 6 : Spark ML
- TP 7 : Dask

Outline

TPs

TP0 : Manual Hadoop Installation

Hadoop Installation: hadoop-2.7.7.tar.gz

Installation

```
> cd /home/hduser
> mkdir local ; cd local
> wget https://www.apache.org/dist/hadoop/core/
    hadoop-2.7.7/hadoop-2.7.7.tar.gz
> tar xvfz hadoop-2.7.7.tar.gz
> mv hadoop-2.7.7 hadoop
> chown -R hduser:hadoop hadoop
```

TPs

TP0 : Installation Hadoop

Env parameter settings

```
# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
export JAVA_HOME=/usr/local/Java/1.8.0-xxx

# Set Hadoop-related environment variables
export HADOOP_HOME=/home/hduser/local/hadoop

export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME

export PATH=${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin:$PATH
```

TPs

TP0 : Installation Hadoop

Configuration files settings

```
# CREATE HADOOP TMP DIR
> mkdir -p /home/hduser/app/hadoop/tmp
> chown hduser:hadoop /home/hduser/app/hadoop/tmp
> chmod 750 /home/hduser/app/hadoop/tmp

# CREATE HDFS WORKINGDIR TO MNG HDFS File System
> mkdir -p /home/hduser/var/local/hadoop/hdfs/data
> chmod -R 777 /home/hduser/var/local/hadoop/hdfs
```

TPs

TP0 : Installation Hadoop

Configuration files in /home/hduser/local/hadoop/etc/hadoop

hadoop-env.sh modification

```
JAVA_HOME="true␣java␣JOME␣path"
export JAVA_HOME=${JAVA_HOME}
```

core-site.xml settings

```
<property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hduser/app/hadoop/tmp</value>
        <description>A base for other temporary directories.</description>
</property>
<property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
        <description>The name of the default file system.</description>
</property>
```

TPs

TP0 : Installation Hadoop

Configuration files in `/home/hduser/local/hadoop/etc/hadoop`

hdfs-site.xml settings

```
<property>
    <name>dfs.data.dir</name>
    <value>/home/hduser/var/local/hadoop/hdfs/data</value>
    <final>true</final>
</property>

<property>
        <name>dfs.replication</name>
        <value>1</value>
</property>
```

TPs

TP0 : Installation Hadoop

Configuration files in /home/hduser/local/hadoop/etc/hadoop
Copy mapred-site.xml.template mapred-site.xml

mapred-site.xml settings

```
<property>
        <name>mapred.job.tracker</name>
        <value>localhost:9001</value>
</property>
```

TPs

TP0 : Installation Hadoop

Configuration files in /home/hduser/local/hadoop/etc/hadoop

yarn-site.xml settings

```xml
<configuration>
        <!-- Site specific YARN configuration properties -->
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
                <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
</configuration>
```

TPs

TP0 : Installation Hadoop

Lauch all services

```
> $HADOOP_HOME/sbin/start-hdfs.sh
> $HADOOP_HOME/sbin/start-yarn.sh
```

Check lauched services

```
> jps

26867 DataNode
28228 Jps
27285 ResourceManager
26695 NameNode
27082 SecondaryNameNode
27420 NodeManager
```

TPs

TP0 : Hadoop cluster set up using docker

- Check Docker installation

Docker : Set up

```
> docker --version
> docker-compose --version
```

- Test of NGINX server

Docker : Check using nginx

```
> docker run -d -p 80:80 --name myserver nginx
```

Visit `http://localhost` and check nginx server homepage

TPs

TP0 : Hadoop cluster set up using docker

- Check Docker installation

Docker : Set up

```
> cd TP0/Docker
> git clone https://github.com/big-data-europe/
    docker-hadoop.git
> cd docker-hadoop
> docker-compose up -d
> docker ps
```

- Test the cluster installation on `http://localhost:9870`
- Close the Hadoop cluster

Docker : Safely close the hadoop cluster

```
> docker-compose down
```

Outline

TPs

TP1 : MapReduce with Hadoop

Project MapReduce :
`/home/hduser/BigDataHadoopSpark/TPs/TP1/MapReduce`
Two projects, A java Project and a python project

```
MapReduce|
         |--pom.xml
         |--bin|
         |--python|--mapper.py
         |        |--reduce.py
         |--src|--hadoop|--WordCount.Java
         |--target|
         |--test|wordcount|--file01
                          |--file02
                |books|--b0
                      |--b1
```

TPs

TP1 : MapReduce with Hadoop

Java project:

- create directory in hdfs `/user/hduser/`**`input`**
- copy the files of `MapReduce/test/wordcount` in `/user/hduser/`**`input`**
- generate Java project BigDataTP1

```
cd BigDataHadoopSpark/TPs/TP1/MapReduce
mvn package
```

- apply Java WordCount application
- check results

TPs

TP1 : MapReduce with Hadoop

Python project

- create directory in hdfs `/user/hduser/input/book`
- copy the files of `MapReduce/test/book` in `/user/hduser/input`
- apply Python WordCount application
- check results

TPs

TP1 : WorldCount and MapReduce using the containerized Hadoop cluster

Commande to check available container

Docker : get list of container id

```
> docker ps -a
> docker container ls -a
```

Commande to copy file in a container :

Docker : copy file in a container

```
> docker cp my_file namenode:/tmp/myfile
```

Running a container mounting a volume '$PWD' in a one directory '/data':

Docker : copy file in a container

Outline

TPs

TP2 : Manual Spark Framework Installation

Spark Installation: spark-2.4.0-bin-hadoop-2.7.tgz

Installation

```
> cd /home/hduser
> mkdir local ; cd local
> wget https://www.apache.org/dyn/closer.lua/spark/spark-2.4.4/spark-2.4.4-bin-hadoop2.7.tgz
> tar xvfz spark-2.4.0-bin-hadoop-2.7.7.tar.gz
> mv spark-2.4.0-bin-hadoop-2.7.7 spark
> export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
> export SPARK_HOME=/home/hduser/local/spark
> export PATH=$SPARK_HOME/bin:$PATH
> export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
```

TPs

TP2 : PySpark Installation

Installation

```
> cd /home/hduser
> export SPARK_HOME=/home/hduser/local/spark
> export PATH=$SPARK_HOME/bin:$PATH
> export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
> export PYSPARK_PYTHON="path_to_python"
> pip install pyspark
> pip install findspark
> sbin/start-master.sh
> sbin/start-slave.sh spark://localhost:7077
```

TPs

TP2 : PySpark Installation

test Spark shell

```
> spark-shell

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://localhost:4040
Spark context available as ’sc’ (master = local[*], app id = local-1578948405576).
Spark session available as ’spark’.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
␣␣␣/___/␣.__/\_,_/_/␣/_/\_\␣␣␣version␣3.0.0-preview2
␣␣␣␣␣␣/_/

Using␣Scala␣version␣2.12.10␣(Java␣HotSpot(TM)␣64-Bit␣Server␣VM,␣Java␣1.8.0_92)
Type␣in␣expressions␣to␣have␣them␣evaluated.
Type␣:help␣for␣more␣information.

scala>
```

TPs

TP2 : PySpark Installation

test Spark shell

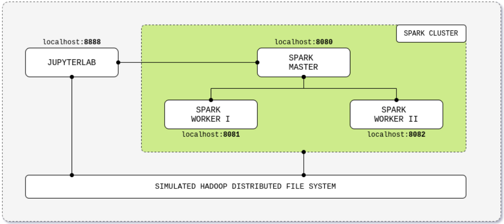
```
> pyspark

Python 2.7.5 (default, Apr 11 2018, 07:36:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
/work/irlin355_1/gratienj/BigData/local/spark/spark-3.0.0-preview2-bin-hadoop2.7/python/pyspark/
      context.py:219: DeprecationWarning: Support for Python 2 and Python 3 prior to version 3.6 is
      deprecated as of Spark 3.0. See also the plan for dropping Python 2 support at https://spark.
      apache.org/news/plan-for-dropping-python-2-support.html.
  DeprecationWarning)
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.0-preview2
      /_/

Using Python version 2.7.5 (default, Apr 11 2018 07:36:10)
SparkSession available as 'spark'.
```

TPs

TP2 : Spark Framework Installation with docker



Installation

```
> cd TP2/Docker
> git clone https://github.com/cluster-apps-on-docker/spark-standalone-cluster-on-docker.git
> curl -LO https://raw.githubusercontent.com/cluster-apps-on-docker/spark-standalone-cluster-on-docker
      /master/docker-compose.yml
> docker-compose up
```

TPs

TP3 : Test0 Test1 Test2

Test0 :

- create spark context
- create liste of integer
- partition list with spark
- print num of partions

Test1 :

- compute square of integer list
- print square list

Test2 :

- compute PI

Outline

TPs

TP4 : Spark ML, Data processing

Test0 :

- load `TPs/data/iris.csv` file in Panda DataFrame
- create Spark DataFrame
- show 5 first lines
- select two columns
- print some statistics on Spark Data frame

Outline

TPs

TP5 : Spark ML, Data processing

Spark ML :

- load `TPs/data/iris.csv` file in Panda DataFrame
- create Spark DataFrame
- create Pipeline to prepare date for machine learning
- compute a predicting model
- evaluate the predicting model

Outline

TPs

Project : Spark ML, Image processing

project :

- load Lena.jpg file
- develop a parallel median Filter in python with Spark

Outline

TPs

Test0 Test1 Test2 with Dask

Test0 :

- create Dask client
- create liste of integer
- partition list with dask

Test1 :

- compute square of integer list
- print square list

Test2 :

- compute PI

Test3 :

- create Dask bags, Array and DataFrame form h5, csv and json files
- directories small_weather account and nycflights