BigData and Machine Learning with Hadoop and Spark Frameworks

Jean-Marc GRATIEN[1]

[1] Department of Computer Science
IFP New Energy

January 25th 2022 / Master Data-AI

Outline I

Outline II

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?

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Introduction to Hadoop

Why Hadoop?

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
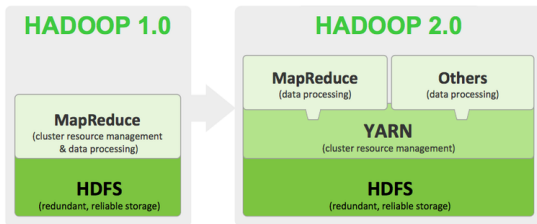MapReduce

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Hadoop distributions

Hadoop Distributions :

Distributions

- Hortonworks
- Cloudera
- MAPR

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Hadoop Cluster

Hadoop Cluster

BRAD HEDLUND .com
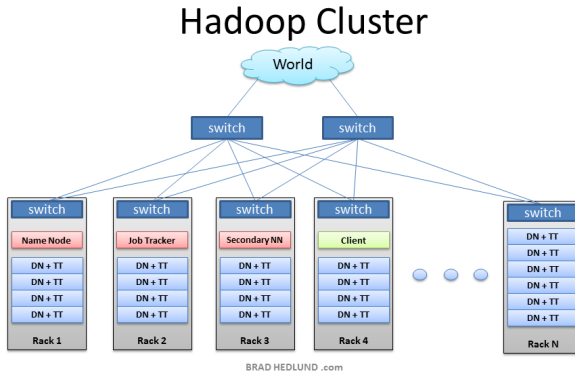
Purpose : Scalability, Fault tolerance and Reliality management

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework
Hadoop Cluster

Concepts

- NameNode
- DataNode
- Replication
- Block, Blocksize

HDFS : Hadoop Distributed Filesystem

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce
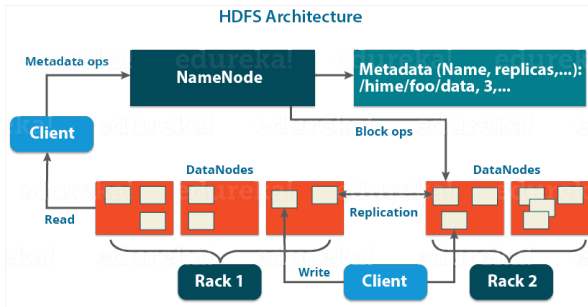
Hadoop Framework: HDFS

HDFS commands

HDFS commands :

Starting HDFS

```
1   # Format nodes
2   > hadoop namenode -format
3
4   # Starting HDFS services
5   > start-dfs.sh
```

Shutting down HDFS

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework: HDFS

HDFS commands

HDFS commands :

Inserting Data into HDFS

```
1  # Step 1 : Create input directory
2  > $HADOOP_HOME/bin/hadoop fs -mkdir /usr/input
3
4  # Step 2 : copy data from local filesystem to hdfs
       filesystem
5  > $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /
       user/input
6
7  # Step 3 : check results with ls cmd
8  > $HADOOP_HOME/bin/hadoop fs -ls /usr/input
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework: HDFS

HDFS commands

HDFS commands :

Retreiving Data from HDFS

```
1  # Step 1 : view data
2  > $HADOOP_HOME/bin/hadoop fs -cat /user/outputfile
3
4  # Step 2 : get data from hdfs filesystem to local
       filesystem
5  > $HADOOP_HOME/bin/hadoop fs -get /user/output/ /
       home/hadoop_out
6
7  # Step 3 : check results with ls cmd
8  > $HADOOP_HOME/bin/hadoop fs -mkdir /usr/input
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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>	

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework
YARN

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce
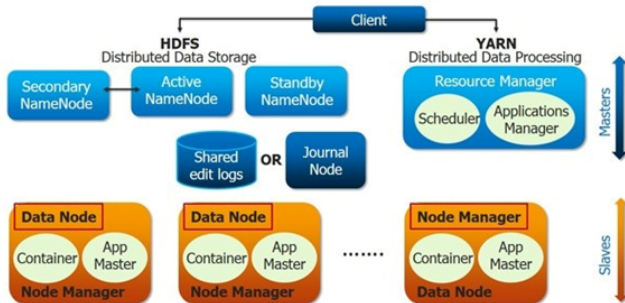
Hadoop Framework

YARN

Starting YARN

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

Shutting down YARN

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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:

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Web tools

Hadoop Web Tools

File browser web page:

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

MapReduce Framework

MapReduce Framework

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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.

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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>)

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

MapReduce Framework

The Overall MapReduce Word Count Process

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce
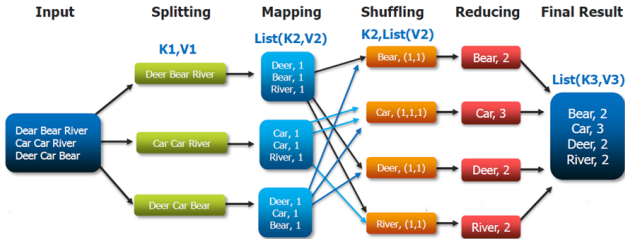
Hadoop Framework

Java exemple

WordCount Java class

```
1   import org.apache.hadoop.*;
2   public class WordCount  {
3     public static class  TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
4       public void map(Object key, Text value, Context context)
5                     throws IOException, InterruptedException {
6           ... ;
7       }
8     }
9     public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable>  {
10      public void reduce(Text key, Iterable<IntWritable> values,Context context)
11                    throws IOException, InterruptedException  {
12          ... ;
13      }
14    }
15    public static void main(String[] args) throws Exception {
16    ...;
17    }
18 }
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Java exemple

Word Count Mapper Java class

```
1   public class WordCount
2   {
3     public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>
4     {
5       public void map(Object key, Text value, Context context)
6                     throws IOException, InterruptedException
7       {
8         StringTokenizer itr = new StringTokenizer(value.toString());
9         while (itr.hasMoreTokens()) {
10          word.set(itr.nextToken());
11          context.write(word, one);
12        }
13      }
14    }
15  }
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Java exemple

WordCount Reducer Java class

```
 1  public class WordCount
 2  {
 3      public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable>
 4      {
 5          private IntWritable result = new IntWritable();
 6          void reduce(Text key, Iterable<IntWritable> values,Context context)
 7                      throws IOException, InterruptedException
 8          {
 9              int sum = 0;
10              for (IntWritable val : values) {
11                  sum += val.get();
12              }
13              result.set(sum);
14              context.write(key, result);
15          }
16      }
17  }
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Java exemple

Main test function

```
 1  public class WordCount
 2  {
 3    public static void main(String[] args) throws Exception
 4    {
 5      Configuration conf = new Configuration();
 6      Job job = Job.getInstance(conf, "word_count");
 7      job.setJarByClass(WordCount.class);
 8      job.setMapperClass(TokenizerMapper.class);
 9      job.setCombinerClass(IntSumReducer.class);
10      job.setReducerClass(IntSumReducer.class);
11      job.setOutputKeyClass(Text.class);
12      job.setOutputValueClass(IntWritable.class);
13      FileInputFormat.addInputPath(job, new Path(args[0]));
14      FileOutputFormat.setOutputPath(job, new Path(args[1]));
15      System.exit(job.waitForCompletion(true) ? 0 : 1);
16    }
17  }
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Java exemple

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

Prepare Test Data

```
1   // Two test file file01 file02 in current directory
2   $ ls
3      file01
4      file02
5   $ hdfs dfs -put file01 /user/gratienj/input
6   $ hdfs dfs -cat /user/gratienj/input/file01
7      Hello World Bye World
8   $ hdfs dfs -put file02 /user/gratienj/input
9   $ hdfs dfs -cat /user/gratienj/input/file02
10    Hello Hadoop Goodbye Hadoop
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Java exemple

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

Run application

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

Check results

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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))
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Python example

Reducer python script Part 1

```python
1  from operator import itemgetter
2  import sys
3  current_word = None
4  current_count = 0
5  word = None
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

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))
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Python example

Python example with Hadoop Streaming : Part 1

Copy test files on HDFS

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

Run application

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Framework

Python example

Python example with Hadoop Streaming : Part 2

Check results

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
HDFS
Yarn
MapReduce

Hadoop Web tools

Ambari Server Tools

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

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Apache Spark Architecture

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX
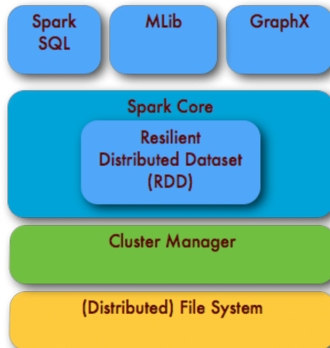
Spark Framework

Apache Spark Ecosystem
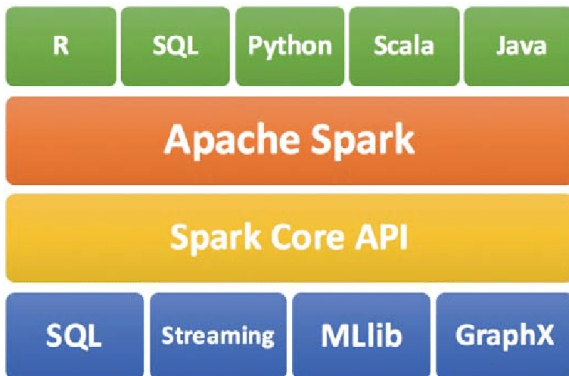
Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : Spark configuration

Configuring a SparkContext

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : Spark context

Create a SparkContext

```
1  import pyspark
2  from pyspark import SparkContext
3  sc =SparkContext()
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : Data concepts

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX
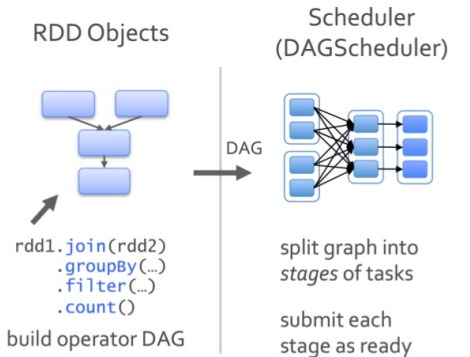
Spark Framework
Spark Core : Data concepts

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : RDD

Create a Spark RDD

```
1  import pyspark
2  from pyspark import SparkContext
3  sc =SparkContext()
4
5  nums= sc.parallelize([1,2,3,4])
6
7  nums.take(1)
```

Output

```
1  [1]
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : RDD Transformationq and Actions

Spark RDD transformations and Actions

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

Output

```
1  1
2  4
3  9
4  16
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : RDD Transformations and Actions

Transformation :

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : RDD Transformations and Actions

Examples :

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

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()	apply lambda
	to all elements
collect()	create a collection
count()	count elements
max(), min()	stats
sum()	

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark SQL : DataFrame

Unified Data Abstraction

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Core : RDD

Create a Spark SQL context

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

Create a DataFrame

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark SQL : DataFrame

Print DataFrame Schema

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark SQL : DataFrame

Print DataFrame Schema

```
1   df = sqlContext.read.csv(SparkFiles.get("adult_data.csv"), header=True, inferSchema= True)
2   df_string.printSchema()
3   root
4    |-- age: string (nullable = true)
5    |-- workclass: string (nullable = true)
6    |-- fnlwgt: string (nullable = true)
7    |-- education: string (nullable = true)
8    |-- education_num: string (nullable = true)
9    |-- marital: string (nullable = true)
10   |-- occupation: string (nullable = true)
11   |-- relationship: string (nullable = true)
12   |-- race: string (nullable = true)
13   |-- sex: string (nullable = true)
14   |-- capital_gain: string (nullable = true)
15   |-- capital_loss: string (nullable = true)
16   |-- hours_week: string (nullable = true)
17   |-- native_country: string (nullable = true)
18   |-- label: string (nullable = true)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark SQL : DataFrame

Select columns

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

Select columns

```
1   +---+------+
2   |age|fnlwgt|
3   +---+------+
4   | 39| 77516|
5   | 50| 83311|
6   | 38|215646|
7   | 53|234721|
8   | 28|338409|
9   +---+------+
10  only showing top 5 rows
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark SQL : DataFrame

Select columns

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

Select columns

```
 1   +------------+-----+
 2   |   education|count|
 3   +------------+-----+
 4   |   Preschool|   51|
 5   |     1st-4th|  168|
 6   |     5th-6th|  333|
 7   |   Doctorate|  413|
 8   |        12th|  433|
 9   |         9th|  514|
10   | Prof-school|  576|
11   |     7th-8th|  646|
12   |        10th|  933|
13   |   Assoc-acdm| 1067|
14   |        11th| 1175|
15   |   Assoc-voc| 1382|
16   |     Masters| 1723|
17   |   Bachelors| 5355|
18   |Some-college| 7291|
19   |     HS-grad|10501|
20   +------------+-----+
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark SQL : DataFrame

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

- count,
- mean,min,max
- standarddeviation

Describe

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

Describe

```
1  +-------+------------------+
2  |summary|       capital_gain|
3  +-------+------------------+
4  |  count|             32561|
5  |   mean|1077.6488437087312|
6  | stddev| 7385.292084840354|
7  |    min|                 0|
8  |    max|             99999|
9  +-------+------------------+
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Mlib

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX
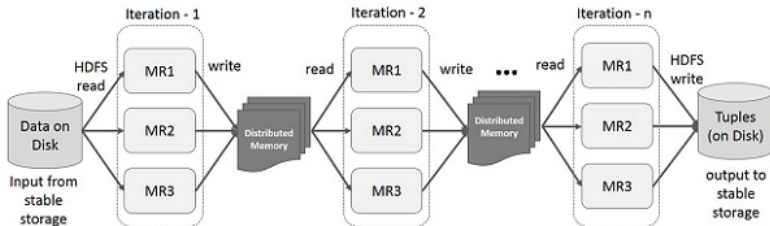
Spark Framework

Spark Mlib

MLlib Pipeline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX
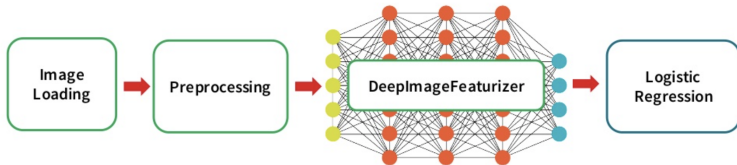
Spark Framework

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

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)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Mlib : Data processing

OneHotEncoder

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

VectorAssembler

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Mlib : Data processing

Pipeline

```
1  from pyspark.ml import Pipeline
2  # DEFINE LIST OF STAGES
3  stages = [[label_indexer],[cat_key_indexer,encoder
      ],[assembler]]
4
5  # DEFINE PIPELINE
6  pipeline = Pipeline(stages=stages)
7
8  # APPLY PIPELINE
9  pipelineModel = pipeline.fit(df_rdd)
10 model_df_rdd = pipelineModel.transform(rdd_df)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark ML : ML pipeline Part 1

create DataFrame

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

Split data

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark ML : ML pipeline Part 2

Train model

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

Make prediction

```
1  lr.transform()
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark Streaming

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture and Ecosystem
Spark Modules : Core, SQL, Mlib, Streaming and GraphX

Spark Framework

Spark GraphX
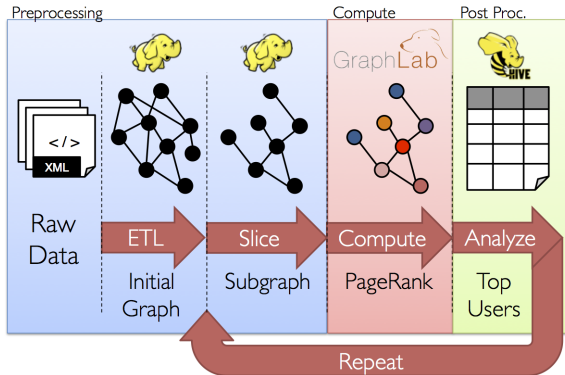
Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Framework
Introduction to Dask

- Dask : Open source python framework for data processing
- developed 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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Framework

Dask Architecture

Dask architecture:

- Dask Cluster
- Dask Scheduler
- Dask collections

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem
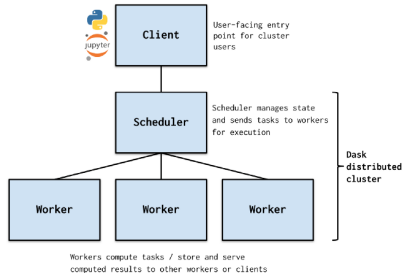
Dask Framework
Dask Cluster

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem
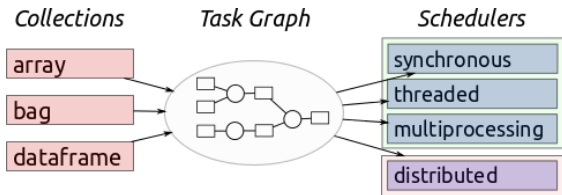
Dask Framework

Dask collections

Dask Array

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Framework
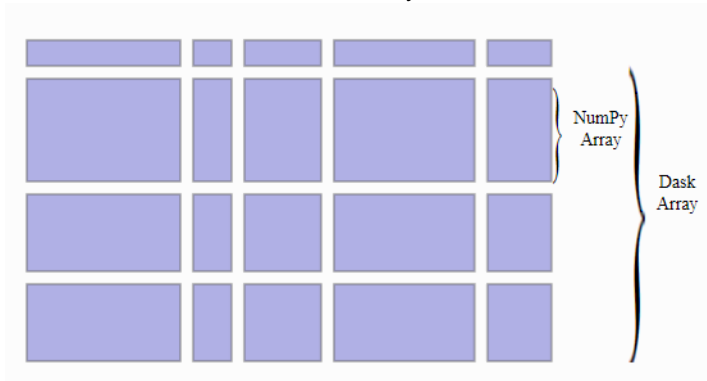
Dask DataFrame

Dask DataFrame

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask installation

Installation

```
1  conda install dask
```

OR

Installation

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask starting cluster

Lauching Dask cluster

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

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

Dashboard

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask Collections

Dask Bag

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

Dask Array

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

Dask DataFrame

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask Distributed

Dask Delayed function

```
1  from dask import delayed
2  @delayed
3  def inc(x):
4      return x + 1
5  @delayed
6  def add(x, y):
7      return x + y
```

Dask Lazy evaluation

```
1  x = inc(15)
2  y = inc(30)
3  total = add(x, y)
4
5  #visualize DAG
6  total.visualize()
7
8  # execute all tasks
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem
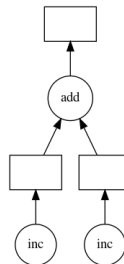
Dask framework

Dask Distributed

Dask Future

```
1  from dask import delayed
2  def inc(x):
3      return x + 1
4  def dec(x):
5      return x - 1
6  def add(x, y):
7      return x + y
```

Dask Lazy evaluation

```
1  from dask.distributed import Client
2  c = Client(n_workers=4)
3
4  x = c.submit(inc, 1)
5  y = c.submit(dec, 2)
6  total = c.submit(add, x, y)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask Distributed

Dask evaluation

```
1   # execute all tasks
2   total.compute()
```

Dask progess

```
1   from dask.distributed import progress
2   # to show progress bar
3   progress(f)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask Distributed

Dask get results

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

Dask persist

```
1    total.persist()
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask framework

Dask Distributed

Dask asynchronism

```
1    from dask.distributed import as_completed
2    def func(x):
3        ...
4        return y
5    futures = [c.submit(func, x) for x in range(n)]
6    iterator = as_completed(futures)
7    for res in iterator:
8        print("RES_Y_:",res.result())
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Outline

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

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, . . .

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Ecosystem

Dask Bag

Dask bag

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

Dask bag

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

Dask bag

```
1  import json
2  js = lines.map(json.loads)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Ecosystem

Dask Array

Dask bag

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

Dask bag : lazy creation

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

Dask bag : Numpy lazy API

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

Dask bag : evaluation

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Ecosystem

Dask DataFrame

Dask DatFrame

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

Dask DataFrame : lazy creation

```
1  import dask.dataframe as dd
2  df = dd.read_csv(filename)
3
4  df = dd.read_csv(os.path.join('data', 'nycflights', '*.csv'),
5                   parse_dates={'Date': [0, 1, 2]})
```

Dask DataFrame : lazy API

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

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Ecosystem

Dask Cluster

Dask Local Cluster

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

Dask YarnCluster

```
1   from dask_yarn import YarnCluster
2   from dask.distributed import Client
3
4   # Create a cluster where each worker has two cores and eight GiB of memory
5   cluster = YarnCluster(environment='environment.tar.gz',
6                         worker_vcores=2,
7                         worker_memory="8GiB")
8   # Scale out to ten such workers
9   cluster.scale(10)
10
11  # Connect to the cluster
12  client = Client(cluster)
```

Introduction
Hadoop
Spark
Dask
TP

Introduction
Architecture
Dask API
Dask Ecosystem

Dask Ecosystem

Dask Cluster Command line SetUp

Dask Scheduler and Workers SetUp

```
1   $ dask-scheduler
2   Scheduler at:   tcp://192.0.0.100:8786
3
4   $ dask-worker tcp://192.0.0.100:8786
5   Start worker at:  tcp://192.0.0.1:12345
6   Registered to:    tcp://192.0.0.100:8786
7
8   $ dask-worker tcp://192.0.0.100:8786
9   Start worker at:  tcp://192.0.0.2:40483
10  Registered to:    tcp://192.0.0.100:8786
```

Dask Client SetUp

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

Introduction
Hadoop
Spark
Dask
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

TPs

TPs require docker usage.

A Docker Quick Sheet can be found in :

```
https:
//github.com/jgratien/BigDataHadoopSparkDaskCourse/
blob/main/TPs/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

Introduction
Hadoop
Spark
Dask
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

Outline

Introduction
Hadoop
Spark
Dask
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

TPs
TP0 : Manual Hadoop Installation

Hadoop Installation: hadoop-2.7.7.tar.gz

Installation

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

Introduction
Hadoop
Spark
Dask
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

TPs

TP0 : Installation Hadoop

Env parameter settings

```
1   # Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
2   export JAVA_HOME=/usr/local/Java/1.8.0-xxx
3
4   # Set Hadoop-related environment variables
5   export HADOOP_HOME=/home/hduser/local/hadoop
6
7   export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
8   export HADOOP_MAPRED_HOME=$HADOOP_HOME
9   export HADOOP_COMMON_HOME=$HADOOP_HOME
10  export HADOOP_HDFS_HOME=$HADOOP_HOME
11  export YARN_HOME=$HADOOP_HOME
12
13  export PATH=${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin:$PATH
```

TPs

TP0 : Installation Hadoop

Configuration files settings

```
1   # CREATE HADOOP TMP DIR
2   > mkdir -p /home/hduser/app/hadoop/tmp
3   > chown hduser:hadoop /home/hduser/app/hadoop/tmp
4   > chmod 750 /home/hduser/app/hadoop/tmp
5
6   # CREATE HDFS WORKINGDIR TO MNG HDFS File System
7   > mkdir -p /home/hduser/var/local/hadoop/hdfs/data
8   > 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

```
1   JAVA_HOME="true_java_JOME_path"
2   export JAVA_HOME=${JAVA_HOME}
```

core-site.xml settings

```
1   <property>
2           <name>hadoop.tmp.dir</name>
3           <value>/home/hduser/app/hadoop/tmp</value>
4           <description>A base for other temporary directories.</description>
5   </property>
6   <property>
7           <name>fs.default.name</name>
8           <value>hdfs://localhost:9000</value>
9           <description>The name of the default file system.</description>
10  </property>
```

TPs

TP0 : Installation Hadoop

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

hdfs-site.xml settings

```
1  <property>
2      <name>dfs.data.dir</name>
3      <value>/home/hduser/var/local/hadoop/hdfs/data</value>
4      <final>true</final>
5  </property>
6
7  <property>
8          <name>dfs.replication</name>
9          <value>1</value>
10 </property>
```

Introduction
Hadoop
Spark
Dask
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

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

```
1  <property>
2          <name>mapred.job.tracker</name>
3          <value>localhost:9001</value>
4  </property>
```

TPs
TP0 : Installation Hadoop

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

yarn-site.xml settings

```
1  <configuration>
2      <!-- Site specific YARN configuration properties -->
3      <property>
4          <name>yarn.nodemanager.aux-services</name>
5          <value>mapreduce_shuffle</value>
6      </property>
7      <property>
8          <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
9          <value>org.apache.hadoop.mapred.ShuffleHandler</value>
10     </property>
11 </configuration>
```

Introduction
Hadoop
Spark
Dask
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

TPs

TP0 : Installation Hadoop

Lauch all services

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

Check lauched services

```
1  > jps
2
3  26867 DataNode
4  28228 Jps
5  27285 ResourceManager
6  26695 NameNode
7  27082 SecondaryNameNode
8  27420 NodeManager
```

Introduction
Hadoop
Spark
Dask
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

TPs

TP0 : Hadoop cluster set up using docker

- Check Docker installation

Docker : Set up

```
1  > docker --version
2  > docker-compose --version
```

- Test of NGINX server

Docker : Check using nginx

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

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

Introduction
Hadoop
Spark
Dask
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

TPs

TP0 : Hadoop cluster set up using docker

- Check Docker installation

Docker : Set up

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

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

Docker : Safely close the hadoop cluster

```
1  > docker-compose down
```

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

Introduction
Hadoop
Spark
Dask
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

Outline

TPs

TP1 : MapReduce with Hadoop

Project MapReduce :

`/home/hduser/BigDataHadoopSpark/TPs/TP1/MapReduce`

Two projects, A java Project and a python project

```
1   MapReduce|
2           |--pom.xml
3           |--bin|
4           |--python|--mapper.py
5           |        |--reduce.py
6           |--src|--hadoop|--WordCount.Java
7           |--target|
8           |--test|wordcount|--file01
9                            |--file02
10               |books|--b0
11                     |--b1
```

Introduction
Hadoop
Spark
Dask
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

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

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

- apply Java WordCount application
- check results

Introduction
Hadoop
Spark
Dask
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

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

Remind : Docker command Quick Sheet :

`BigDataHadoopSparkDaskCourse/TPs/doc/Docker.md`

Docker : exec running conatairner named namenode

```
1  > cd TPs/TP1/MapReduce
2  > docker exec -it namenode bash
```

Docker : copy file in a container

```
1  > cd TPs/TP1/MapReduce
2  > docker cp test/wordcount/file01 namenode:file01
3  > docker cp test/wordcount/file02 namenode:file02
4  > docker cp jars/hadoop-mapreduce-examples-2.7.1-sources.jar namenode:hadoop-mapreduce-examples-2.7.1-
     sources.jar
```

TPs

TP1 : WorldCount and MapReduce using the containerized Hadoop cluster

Docker :Realize WordCount TP within the namenode container

```
1   /$ mkdir input
2   /$ mv file01 file02 input/.
3   /$ hadoop fs -mkdir -p input
4   /$ hdfs dfs -put ./input/* input
5   /$ hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar org.apache.hadoop.examples.WordCount input
        output
```

Docker :Check results

```
1   /$ hdfs dfs -cat output/part-r-00000
```

Introduction
Hadoop
Spark
Dask
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

Outline

Introduction
Hadoop
Spark
Dask
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

TPs

TP2 : Manual Spark Framework Installation

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

Installation

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

TPs

TP2 : PySpark Installation

Installation

```
1   > cd /home/hduser
2   > export SPARK_HOME=/home/hduser/local/spark
3   > export PATH=$SPARK_HOME/bin:$PATH
4   > export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
5   > export PYSPARK_PYTHON="path_to_python"
6   > pip install pyspark
7   > pip install findspark
8   > sbin/start-master.sh
9   > sbin/start-slave.sh spark://localhost:7077
```

TPs

TP2 : PySpark Installation

test Spark shell

```
1   > spark-shell
2
3   Setting default log level to "WARN".
4   To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
5   Spark context Web UI available at http://localhost:4040
6   Spark context available as 'sc' (master = local[*], app id = local-1578948405576).
7   Spark session available as 'spark'.
8   Welcome to
9         ____              __
10       / __/__  ___ _____/ /__
11      _\ \/ _ \/ _ `/ __/  '_/
12    /___/ .__/\_,_/_/ /_/\_\   version 3.0.0-preview2
13       /_/
14
15   Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_92)
16   Type in expressions to have them evaluated.
17   Type :help for more information.
18
19   scala>
```

Introduction
Hadoop
Spark
Dask
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

TPs

TP2 : PySpark Installation

test Spark shell

```
1   > pyspark
2
3   Python 2.7.5 (default, Apr 11 2018, 07:36:10)
4   [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
5   Type "help", "copyright", "credits" or "license" for more information.
6   Setting default log level to "WARN".
7   To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
8   /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.
9     DeprecationWarning)
10  Welcome to
11        ____              __
12       / __/__  ___ _____/ /__
13      _\ \/ _ \/ _ `/ __/  '_/
14     /__ / .__/\_,_/_/ /_/\_\   version 3.0.0-preview2
15        /_/
16
17  Using Python version 2.7.5 (default, Apr 11 2018 07:36:10)
18  SparkSession available as 'spark'.
```

TPs

TP2 : Spark Framework Installation with docker
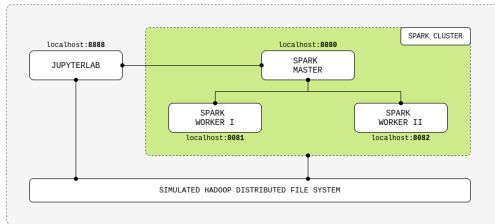
Installation

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

Introduction
Hadoop
Spark
Dask
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

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

Introduction
Hadoop
Spark
Dask
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

TPs
TP3 : Test0 Test1 Test2

Realize Test0, Test 1 and Test2 with the JupiterLab Nodebook

Deploy Spark Cluster

```
1  > cd TP2/Docker
2  > docker-compose up
```

Connect to JupiterLab NoteBook at `http://localhost:8888`

Introduction
Hadoop
Spark
Dask
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

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

Introduction
Hadoop
Spark
Dask
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

Outline

Introduction
Hadoop
Spark
Dask
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

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

Introduction
Hadoop
Spark
Dask
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

Outline

Introduction
Hadoop
Spark
Dask
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

TPs
Project : Spark ML, Image processing

project :

- load Lena.jpg file

- develop a parallel median Filter in python with Spark

Introduction
Hadoop
Spark
Dask
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

Outline

Introduction
Hadoop
Spark
Dask
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

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