C:\HADOOPOUTPUT\spark>spark-submit --verbose wordcountSpark.jar -class JavaWord Count yarn-client

The master URL passed to Spark can be in one of the following formats:

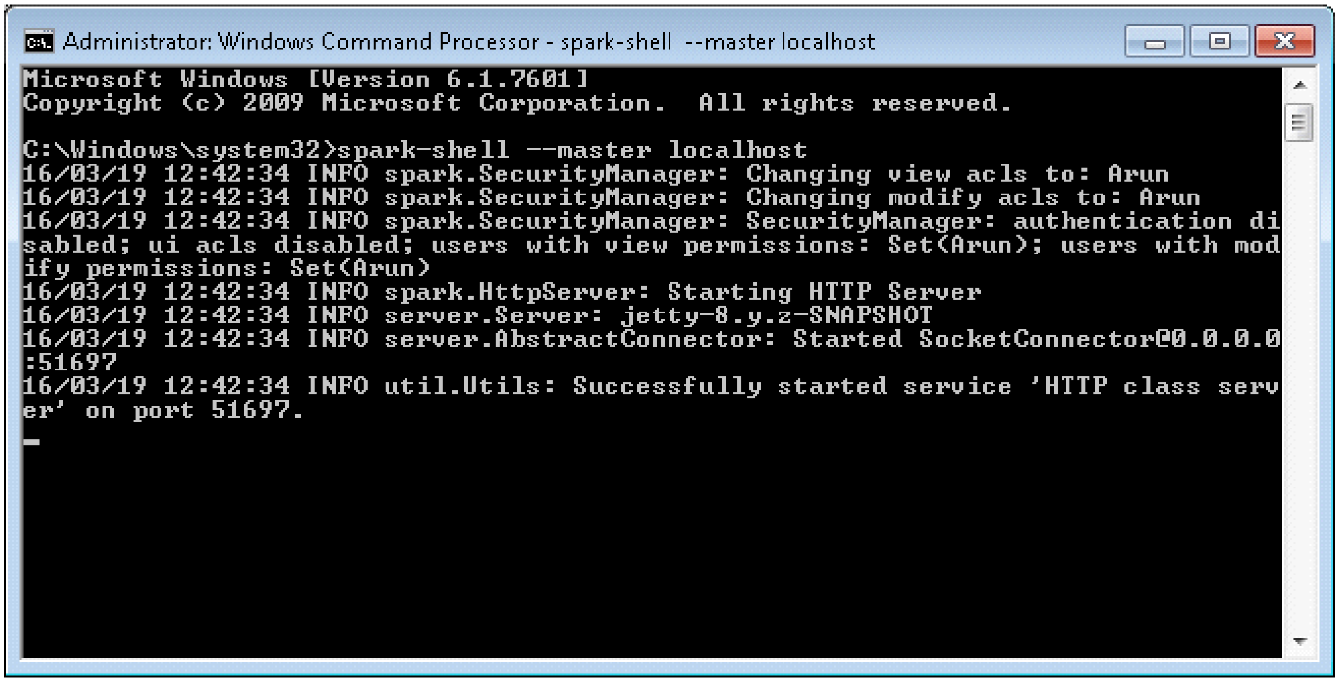
Master URL	**Meaning**
`local`	Run Spark locally with one worker thread (i.e. no parallelism at all).
`local[K]`	Run Spark locally with K worker threads (ideally, set this to the number of cores on your machine).
`local[*]`	Run Spark locally with as many worker threads as logical cores on your machine.
`spark://HOST:PORT`	Connect to the given Spark standalone cluster master. The port must be whichever one your master is configured to use, which is 7077 by default.
`mesos://HOST:PORT`	Connect to the given Mesos cluster. The port must be whichever one your is configured to use, which is 5050 by default. Or, for a Mesos cluster using ZooKeeper, use `mesos://zk://...`. To submit with `--deploy-mode cluster`, the HOST:PORT should be configured to connect to the MesosClusterDispatcher.
`yarn`	Connect to a YARN cluster in `client` or `cluster` mode depending on the value of `--deploy-mode`. The cluster location will be found based on the `HADOOP_CONF_DIR` or `YARN_CONF_DIR` variable.
`yarn-client`	Equivalent to `yarn` with `--deploy-mode client`, which is preferred to `yarn-client`
`yarn-cluster`	Equivalent to `yarn` with `--deploy-mode cluster`, which is preferred to `yarn-cluster

- **file:** - Absolute paths and `file:/` URIs are served by the driver's HTTP file server, and every executor pulls the file from the driver HTTP server.
- **hdfs:**, **http:**, **https:**, **ftp:** - these pull down files and JARs from the URI as expected
- **local:** - a URI starting with local:/ is expected to exist as a local file on each worker node. This means that no network IO will be incurred, and works well for large files/JARs that are pushed to each worker, or shared via NFS, GlusterFS, etc.

NOTE:

To avoid block in use or hdfs running in safe mode block every time starting hadoop, set YARN_CONFIG_DIR

```
YARN_CONFIG_DIR-C:\hadoop-2.2.0\etc\hadoop
```



Administrator: Windows Command Processor - spark-shell --master localhost

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>spark-shell --master localhost
16/03/19 12:42:34 INFO spark.SecurityManager: Changing view acls to: Arun
16/03/19 12:42:34 INFO spark.SecurityManager: Changing modify acls to: Arun
16/03/19 12:42:34 INFO spark.SecurityManager: SecurityManager: authentication di
sabled; ui acls disabled; users with view permissions: Set(Arun); users with mod
ify permissions: Set(Arun)
16/03/19 12:42:34 INFO spark.HttpServer: Starting HTTP Server
16/03/19 12:42:34 INFO server.Server: jetty-8.y.z-SNAPSHOT
16/03/19 12:42:34 INFO server.AbstractConnector: Started SocketConnector@0.0.0.0
:51697
16/03/19 12:42:34 INFO util.Utils: Successfully started service 'HTTP class serv
er' on port 51697.
_
```

above fails finally in localhost
start with spark-shell alone.
>spark-shell

Spark copies the Spark assembly JAR file to HDFS each time you run spark-submit. You can avoid doing this copy each time by manually uploading the Spark assembly JAR file to your HDFS. Then, set the SPARK_JAR environment variable to this HDFS path

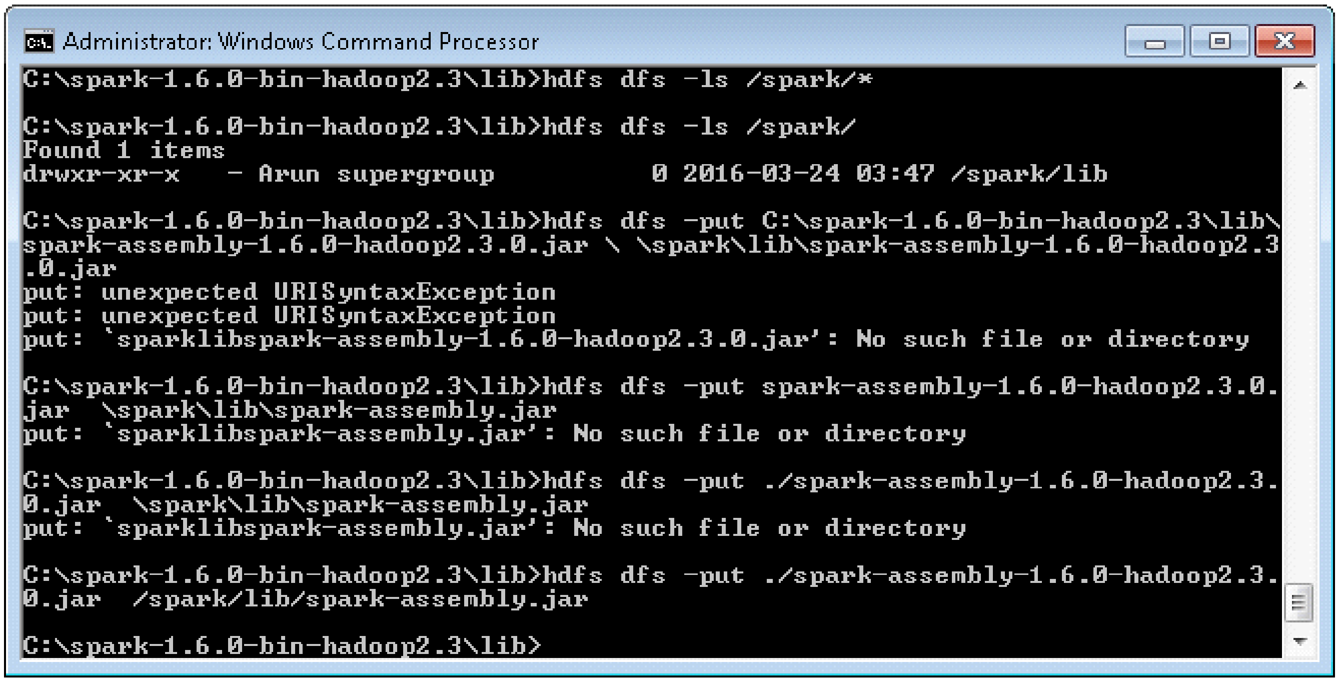
C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -put ./spark-assembly-1.6.0-hadoop2.3.0.jar /spark/lib/spark-assembly.jar

Reading data from hdfs

Files in hdfs are usually stored in the following formats:

- plain txt/csv/json files
- sequence files. You can think of them as serialized java objects. In recent years became less popular. Also they are not portable (need custom readers), so I do not find them interesting for this post.
- avro(row-based)
- paruqet(column-based)
- orc(column-based)

Good news is that Spark (and SparkR!) can read `json`, `parquet`,`orc` with built-in `read.df` function and `csv`,`avro` with`read.df` and spark-avro, spark-csv spark packages.


Administrator: Windows Command Processor

```
C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -ls /spark/*

C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -ls /spark/
Found 1 items
drwxr-xr-x   - Arun supergroup          0 2016-03-24 03:47 /spark/lib

C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -put C:\spark-1.6.0-bin-hadoop2.3\lib\
spark-assembly-1.6.0-hadoop2.3.0.jar \ \spark\lib\spark-assembly-1.6.0-hadoop2.3
.0.jar
put: unexpected URISyntaxException
put: unexpected URISyntaxException
put: `sparklibspark-assembly-1.6.0-hadoop2.3.0.jar': No such file or directory

C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -put spark-assembly-1.6.0-hadoop2.3.0.
jar  \spark\lib\spark-assembly.jar
put: `sparklibspark-assembly.jar': No such file or directory

C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -put ./spark-assembly-1.6.0-hadoop2.3.
0.jar  \spark\lib\spark-assembly.jar
put: `sparklibspark-assembly.jar': No such file or directory

C:\spark-1.6.0-bin-hadoop2.3\lib>hdfs dfs -put ./spark-assembly-1.6.0-hadoop2.3.
0.jar  /spark/lib/spark-assembly.jar

C:\spark-1.6.0-bin-hadoop2.3\lib>
```

NOTE:

for running java use cmd run-example

```
# For Scala and Java, use run-example:
./bin/run-example SparkPi


# For Python examples, use spark-submit directly:
./bin/spark-submit examples/src/main/python/pi.py


# For R examples, use spark-submit directly:
./bin/spark-submit examples/src/main/r/dataframe.R
```

RUNNING JAVA IN SPARK

```
public class SparkExample {
	public static final int NUM_SAMPLES=10;
	SparkExample(){
		SparkConf conf = new SparkConf().setAppName("Spark Pi");
		conf.setMaster("local[2]");
	    SparkContext sc = new SparkContext(conf);

	    RDD<String> textFile =sc.textFile("hdfs://input/wordcount.txt",1);
	    System.out.println("SparkExample.SparkExample()"+textFile);
```

```
          }

          public static void main(String args[]){
                    new SparkExample();
          }
}
```

OUTPUT:

```
16/03/24 05:39:55 INFO BlockManagerMasterEndpoint: Registering block manager
localhost:56793 with 795.0 MB RAM, BlockManagerId(driver, localhost, 56793)
16/03/24 05:39:55 INFO BlockManagerMaster: Registered BlockManager
16/03/24 05:39:56 INFO MemoryStore: Block broadcast_0 stored as values in memory
(estimated size 104.0 KB, free 104.0 KB)
16/03/24 05:39:57 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory
(estimated size 9.8 KB, free 113.8 KB)
16/03/24 05:39:57 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on
localhost:56793 (size: 9.8 KB, free: 795.0 MB)
16/03/24 05:39:57 INFO SparkContext: Created broadcast 0 from textFile at
SparkExample.java:15
SparkExample.SparkExample()hdfs://input/wordcount.txt MapPartitionsRDD[1] at textFile at
SparkExample.java:15
16/03/24 05:39:57 INFO SparkContext: Invoking stop() from shutdown hook
16/03/24 05:39:57 INFO SparkUI: Stopped Spark web UI at http://localhost:4041
16/03/24 05:39:57 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint st
```

MAP REDUCE IN SPARK WITHOUT USING HADOOP

```
package spark;

import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;

import scala.Tuple2;

public final class JavaWordCount {
  private static final Pattern SPACE = Pattern.compile(" ");

  public static void main(String[] args) throws Exception {



    SparkConf sparkConf = new SparkConf().setAppName("JavaWordCount");
    sparkConf.setMaster("local[1]");
    JavaSparkContext ctx = new JavaSparkContext(sparkConf);
    JavaRDD<String> lines = ctx.textFile("c:/HADOOPOUTPUT/wordcount.txt", 1);
System.out.println("JavaWordCount.main()"+lines);
    JavaRDD<String> words = lines.flatMap(new FlatMapFunction<String, String>() {
      @Override
      public Iterable<String> call(String s) {
        return Arrays.asList(SPACE.split(s));
      }
    });

    JavaPairRDD<String, Integer> ones = words.mapToPair(new PairFunction<String, String,
Integer>() {
      @Override
      public Tuple2<String, Integer> call(String s) {
        return new Tuple2<String, Integer>(s, 1);
      }
    });

    JavaPairRDD<String, Integer> counts = ones.reduceByKey(new Function2<Integer, Integer,
Integer>() {
      @Override
      public Integer call(Integer i1, Integer i2) {
        return i1 + i2;
      }
    });

    List<Tuple2<String, Integer>> output = counts.collect();
    for (Tuple2<?,?> tuple : output) {
      System.out.println(tuple._1() + ": " + tuple._2());
    }
    ctx.stop();
  }
}
```

```
16/03/24 06:04:11 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all
completed, from pool
16/03/24 06:04:11 INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:67,
took 0.650911 s
example: 1
are: 1
is: 1
you: 1
wordcount: 1
hadoop: 3
hi: 3
how: 2
16/03/24 06:04:11 INFO SparkUI: Stopped Spark web UI at http://localhost:4041
16/03/24 06:04:11 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint
stopped!
```

See below same program run in hadoop takes 23 sec

```
          if(args!=null && args.length<1){
                    args= new String[2];
                    args[0]="c:/HADOOPOUTPUT/wordcount.txt";
                    args[1]="c:/HADOOPOUTPUT/output";
          }
    Path inputPath = new Path(args[0]);
    Path outputPath = new Path(args[1]);
    Configuration conf = getConf();
    Job job = new ~~Job~~(conf, this.getClass().toString());
    FileInputFormat.setInputPaths(job, inputPath);
    FileOutputFormat.setOutputPath(job, outputPath);
    job.setJobName("WordCount");
    job.setJarByClass(WordCount.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setMapperClass(Map.class);
    job.setCombinerClass(Reduce.class);
    job.setReducerClass(Reduce.class);
    return job.waitForCompletion(true) ? 0 : 1;
  }
```

For Record size 1000:

```
16/03/24 06:27:40 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)
16/03/24 06:27:40 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1
blocks
16/03/24 06:27:40 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 7 ms
16/03/24 06:27:41 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 243558 bytes
result sent to driver
16/03/24 06:27:41 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 776 ms on
localhost (1/1)
16/03/24 06:27:41 INFO DAGScheduler: ResultStage 1 (collect at JavaWordCount.java:67)
finished in 0.777 s
16/03/24 06:27:41 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all
completed, from pool
16/03/24 06:27:41 INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:67,
took 2.073223 s
10:47,Product1,1200,Mastercard,David,Stavenger,Rogaland,country898,1/13/2009: 1
(Bruxelles),country574,1/31/2008: 1
,,,,,,,country1956: 1
,,,,,,,country2735: 1
8:23,Product1,1200,Mastercard,Sarah,Floyds: 1
House: 1
,CA,country82,1/3/2009: 1
22:00,Product2,3600,Amex,Lucien,Wiesbaden,Hessen,country233,1/4/2009: 1
,,,,,,,country2781: 1
```

,,,,,,,c

PERFORMANCE COMPARISION

Comparision of time taken in diff environment.This proves hadoop is good for running small data also

Data limit	Standaone Java	Hadoop	Spark
5 lines of word count	2sec	23sec	0.6 sec(0.79 with hdfs)
1000 lines of csv	2 sec	22 sec	1.52 sec
3500 lines of csv	3sec	28 sec	2.313126 sec
10,000 lines of csv	4sec	26 sec	2.055446sec
65,536 lines of csv(max csv limit)	4sec	26-27	2.998 sec

ACCESS HDFS USING SPARK -JAVA

Spark can accesss hdfs file using hadoop-common-2.2.0.jar(version of inslalled hadoop) in classpath.you need to access using hdfs://localhost:9000/input/wordcount.txt. /input/wordcount.txt is the hdfs file path.

```
package spark;

import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;

import org.apache.hadoop.fs.Hdfs;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;

import scala.Tuple2;

public final class JavaWordCount {
  private static final Pattern SPACE = Pattern.compile(" ");

  public static void main(String[] args) throws Exception {
    SparkConf sparkConf = new SparkConf().setAppName("JavaWordCount");
    sparkConf.setMaster("local[*]");
    JavaSparkContext ctx = new JavaSparkContext(sparkConf);
   // JavaRDD<String> lines = ctx.textFile("C:/HADOOPOUTPUT/SalesJan10000.csv", 1);

    JavaRDD<String> lines = ctx.textFile("hdfs://localhost:9000/input/wordcount.txt", 1);
    System.out.println("JavaWordCount.main()"+lines);
    JavaRDD<String> words = lines.flatMap(new FlatMapFunction<String, String>() {
      @Override
      public Iterable<String> call(String s) {
        return Arrays.asList(SPACE.split(s));
      }
    });

    JavaPairRDD<String, Integer> ones = words.mapToPair(new PairFunction<String, String,
Integer>() {
      @Override
      public Tuple2<String, Integer> call(String s) {
        return new Tuple2<String, Integer>(s, 1);
      }
    });

    JavaPairRDD<String, Integer> counts = ones.reduceByKey(new Function2<Integer, Integer,
Integer>() {
      @Override
      public Integer call(Integer i1, Integer i2) {
        return i1 + i2;
      }
    });

    List<Tuple2<String, Integer>> output = counts.collect();
    for (Tuple2<?,?> tuple : output) {
      System.out.println(tuple._1() + ": " + tuple._2());
    }
    ctx.stop();
  }
}
```

O/P;

```
example: 1
are: 1
is: 1
you: 1
wordcount: 1
hadoop: 3
hi: 3
how: 2

16/03/30 01:24:43 INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:76,
took 0.790907 s
```

NOTE:

1)Access hdfs using hdfs://localhost:<<port>>/<<hdfs file input>>

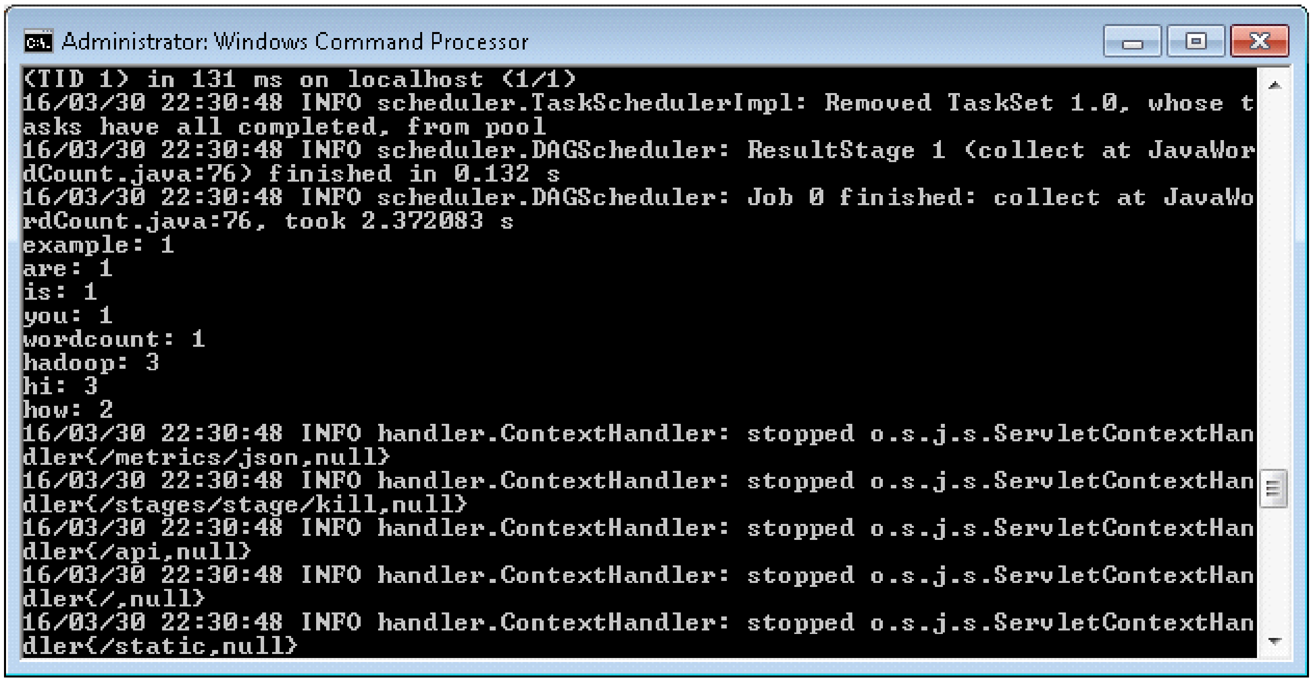
2)for spark with 5 lines of word count program without hdfs it uses 0.6sec when access from local file system but when using hdfs it takes 0.79.
As you can see it takes little longer when using hdfs for small size files.

3) hdfs files shown below

```
C:\Windows\system32>hdfs dfs -ls /input/*
Found 1 items
-rw-r--r--   1 Arun supergroup     123637 2016-02-24 02:11 /input/sales.csv
Found 1 items
-rw-r--r--   1 Arun supergroup    1398907 2016-02-25 00:09 /input/sales10000.csv

Found 1 items
-rw-r--r--   1 Arun supergroup     466379 2016-02-24 22:53 /input/sales3500.csv
Found 1 items
-rw-r--r--   1 Arun supergroup    8594762 2016-02-25 00:22 /input/sales65536.csv

Found 1 items
-rw-r--r--   1 Arun supergroup     129745 2016-03-03 01:29 /input/salesunique.cs
v
Found 1 items
-rw-r--r--   1 Arun supergroup     179820 2016-03-03 01:57 /input/salesunique350
0.csv
Found 1 items
-rw-r--r--   1 Arun supergroup    1476056 2016-03-03 01:47 /input/salesunique655
36.csv
Found 1 items
-rw-r--r--   1 Arun supergroup         70 2016-02-24 02:11 /input/wordcount.txt
```

RUN SPARK FROM CMD PROMPT -JAVA

```
C:\HADOOPOUTPUT\spark>spark-submit --verbose wordcountSpark1.jar  -class spark.
avaWordCount
```



Administrator: Windows Command Processor

```
(TID 1) in 131 ms on localhost (1/1)
16/03/30 22:30:48 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose t
asks have all completed, from pool
16/03/30 22:30:48 INFO scheduler.DAGScheduler: ResultStage 1 (collect at JavaWor
dCount.java:76) finished in 0.132 s
16/03/30 22:30:48 INFO scheduler.DAGScheduler: Job 0 finished: collect at JavaWo
rdCount.java:76, took 2.372083 s
example: 1
are: 1
is: 1
you: 1
wordcount: 1
hadoop: 3
hi: 3
how: 2
16/03/30 22:30:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHan
dler{/metrics/json,null}
16/03/30 22:30:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHan
dler{/stages/stage/kill,null}
16/03/30 22:30:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHan
dler{/api,null}
16/03/30 22:30:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHan
dler{/,null}
16/03/30 22:30:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHan
dler{/static,null}
```

ACCESS HDFS USING SPARK -R Programming

Reading data from hdfs

Files in hdfs are usually stored in the following formats:

- plain txt/csv/json files
- sequence files. You can think of them as serialized java objects. In recent years became less popular. Also they are not portable (need custom readers), so I do not find them interesting for this post.
- avro(row-based)
- paruqet(column-based)
- orc(column-based)

Good news is that Spark (and SparkR!) can read `json`, `parquet`,`orc` with built-in `read.df` function and `csv`,`avro` with`read.df` and spark-avro, spark-csv spark packages.

STEP 1:

```
install devtools
```

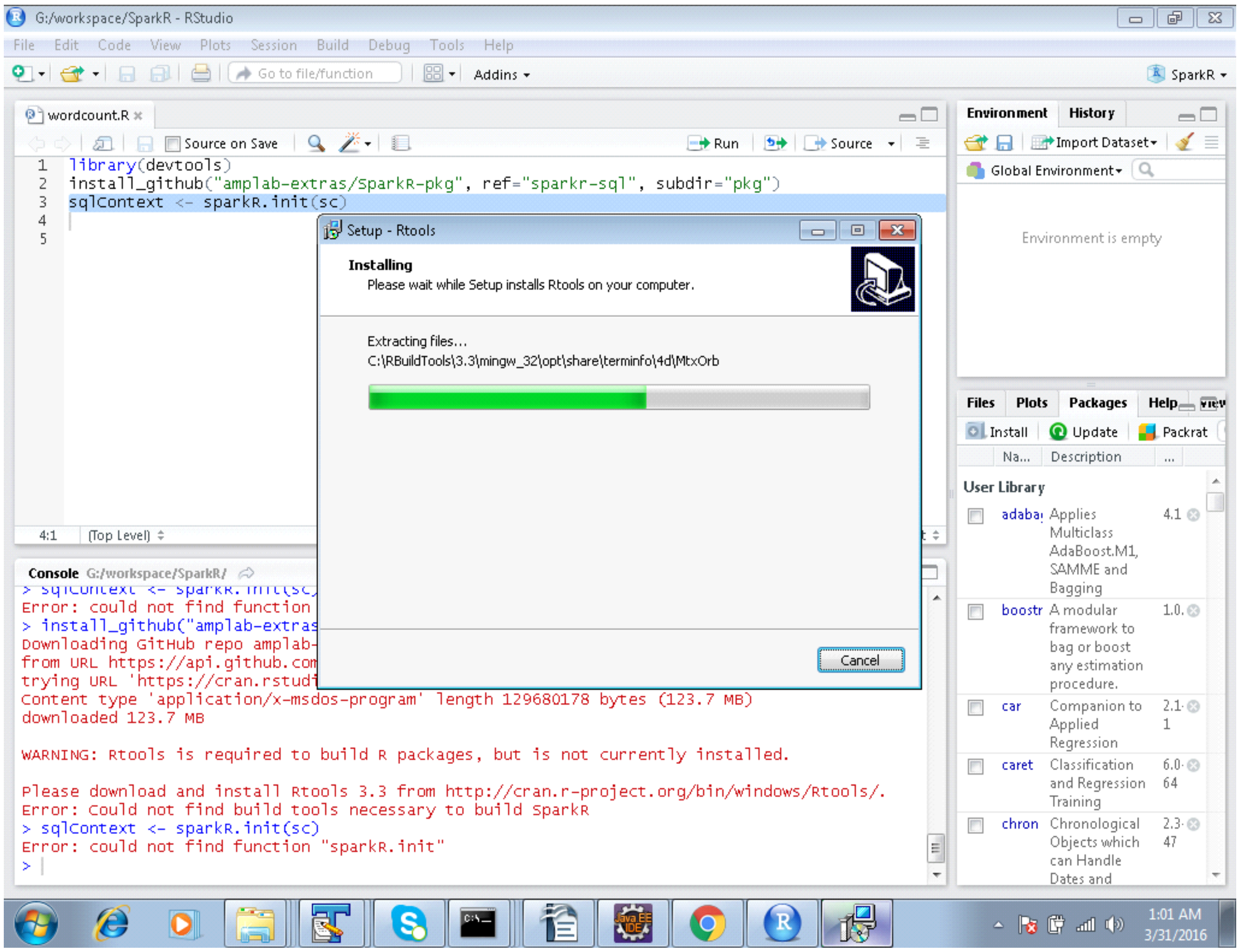
STEP 2:

```
install sparkR
```

STEP 3:

Two ways to install Spark R:
Procedure 1:
when command below is given it automatically download Rbuild Tools
install_github("amplab-extras/SparkR-pkg", ref="sparkr-sql", subdir="pkg")
below are packagesa gets installed:

C:\RBuildTools\3.3
C:\RBuildTools\3.3\gcc-4.6.3
C:\RBuildTools\3.3\mingw_64



```
library(devtools)
install_github("amplab-extras/SparkR-pkg", ref="sparkr-sql", subdir="pkg")
sqlContext <- sparkR.init(sc)
```

Setup - Rtools

Installing
Please wait while Setup installs Rtools on your computer.

Extracting files...
C:\RBuildTools\3.3\mingw_32\opt\share\terminfo\4d\MtxOrb

Cancel

```
> sqlContext <- sparkR.init(sc)
Error: could not find function
> install_github("amplab-extras
Downloading GitHub repo amplab-
from URL https://api.github.com
trying URL 'https://cran.rstudi
Content type 'application/x-msdos-program' length 129680178 bytes (123.7 MB)
downloaded 123.7 MB

WARNING: Rtools is required to build R packages, but is not currently installed.

Please download and install Rtools 3.3 from http://cran.r-project.org/bin/windows/Rtools/.
Error: Could not find build tools necessary to build SparkR
> sqlContext <- sparkR.init(sc)
Error: could not find function "sparkR.init"
>
```

User Library

Na...	Description	...
adaba	Applies Multiclass AdaBoost.M1, SAMME and Bagging	4.1
boostr	A modular framework to bag or boost any estimation procedure.	1.0.
car	Companion to Applied Regression	2.1-1
caret	Classification and Regression Training	6.0-64
chron	Chronological Objects which can Handle Dates and	2.3-47

1:01 AM
3/31/2016

procedure 2:

```
library(devtools)
load_all("C:/spark-1.6.0-bin-hadoop2.3/R/lib/SparkR/R")
```

```
install("C:/spark-1.6.0-bin-hadoop2.3/R/lib/SparkR/R")

library(SparkR)
sqlContext <- SparkR.init(sc)
```

```
install roxygen2 -since load_all requires roxygen2
```

```
Skipping missing files: schema.R, generics.R, jobj.R, RDD.R, pairRDD.R, column.R,
group.R, DataFrame.R, SQLContext.R, backend.R, broadcast.R, client.R, context.R,
deserialize.R, functions.R, mllib.R, serialize.R, sparkR.R, stats.R, types.R,
utils.R
```

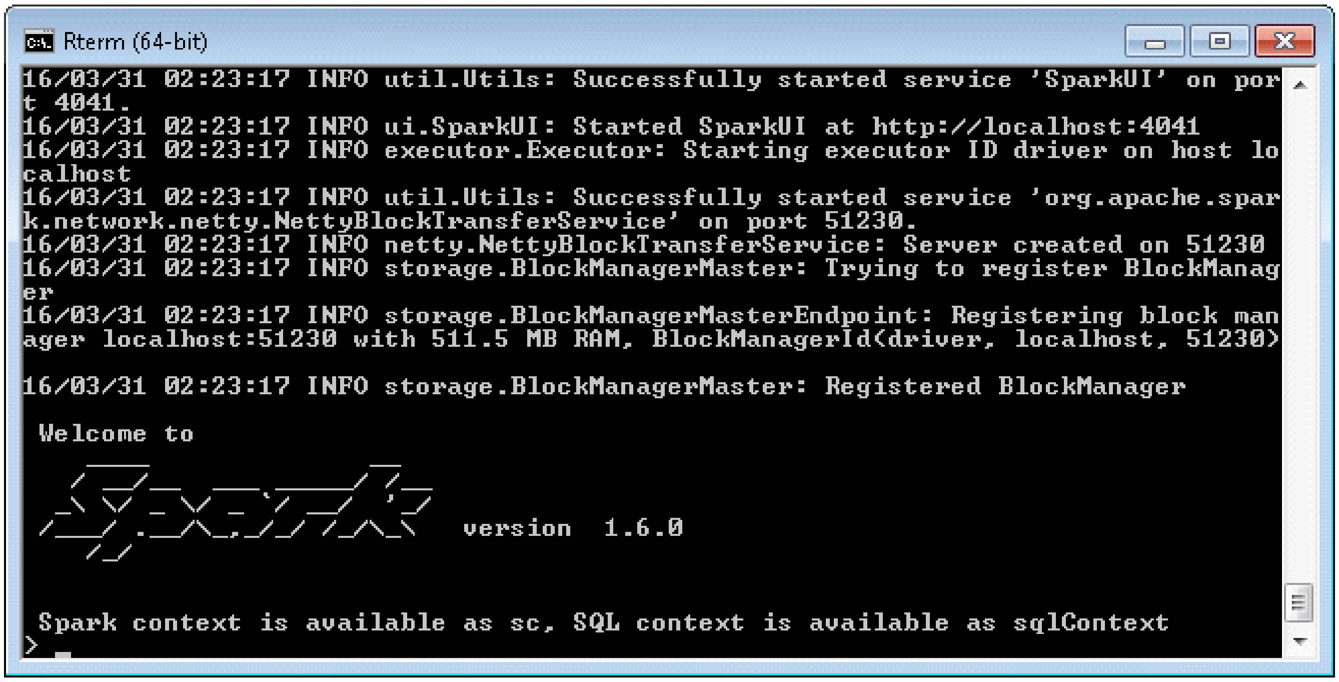
```
> install("C:/spark-1.6.0-bin-hadoop2.3/R/lib/SparkR/R")
Installing SparkR
"C:/PROGRA~1/R/R-32~1.3/bin/x64/R" --no-site-file --no-environ --no-save --no-
restore  \
  CMD INSTALL "C:/spark-1.6.0-bin-hadoop2.3/R/lib/SparkR"  \
  --library="C:/Users/Arun/Documents/R/win-library/3.2" --install-tests
```

procedure 3:

```
Sys.setenv(SPARK_HOME = "C:/spark-1.6.0-bin-hadoop2.3")
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))


library(SparkR)
sc <- sparkR.init(master = "local")
```

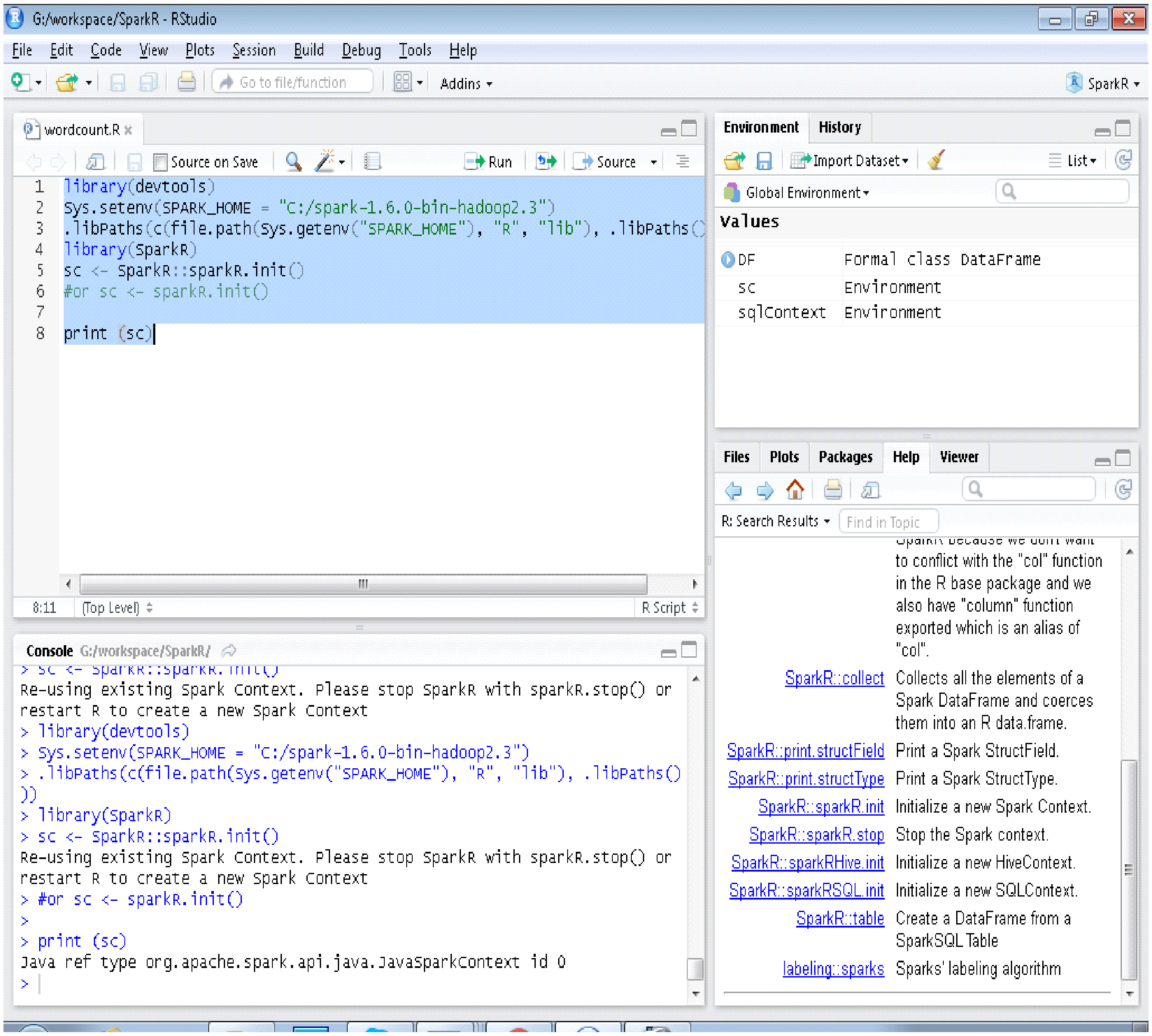
Cmd to start sparkR:

```
C:\Windows\system32>sparkR
```



Rterm (64-bit)

```
16/03/31 02:23:17 INFO util.Utils: Successfully started service 'SparkUI' on por
t 4041.
16/03/31 02:23:17 INFO ui.SparkUI: Started SparkUI at http://localhost:4041
16/03/31 02:23:17 INFO executor.Executor: Starting executor ID driver on host lo
calhost
16/03/31 02:23:17 INFO util.Utils: Successfully started service 'org.apache.spar
k.network.netty.NettyBlockTransferService' on port 51230.
16/03/31 02:23:17 INFO netty.NettyBlockTransferService: Server created on 51230
16/03/31 02:23:17 INFO storage.BlockManagerMaster: Trying to register BlockManag
er
16/03/31 02:23:17 INFO storage.BlockManagerMasterEndpoint: Registering block man
ager localhost:51230 with 511.5 MB RAM, BlockManagerId(driver, localhost, 51230)

16/03/31 02:23:17 INFO storage.BlockManagerMaster: Registered BlockManager

 Welcome to
    ____              __
   / __/__  ___ _____/ /__
  _\ \/ _ \/ _ `/ __/  '_/
 /___/ .__/\_,_/_/ /_/\_\   version  1.6.0
    /_/


 Spark context is available as sc, SQL context is available as sqlContext
>
```



```
library(devtools)
Sys.setenv(SPARK_HOME = "C:/spark-1.6.0-bin-hadoop2.3")
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths(
library(SparkR)
sc <- SparkR::sparkR.init()
#or sc <- sparkR.init()

print (sc)
```

```
> sc <- SparkR::sparkR.init()
Re-using existing Spark Context. Please stop SparkR with sparkR.stop() or
restart R to create a new Spark Context
> library(devtools)
> Sys.setenv(SPARK_HOME = "C:/spark-1.6.0-bin-hadoop2.3")
> .libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()
))
> library(SparkR)
> sc <- SparkR::sparkR.init()
Re-using existing Spark Context. Please stop SparkR with sparkR.stop() or
restart R to create a new Spark Context
> #or sc <- sparkR.init()
>
> print (sc)
Java ref type org.apache.spark.api.java.JavaSparkContext id 0
>
```

Values

DF	Formal class DataFrame
sc	Environment
sqlContext	Environment

	SparkR because we don't want to conflict with the "col" function in the R base package and we also have "column" function exported which is an alias of "col".
SparkR::collect	Collects all the elements of a Spark DataFrame and coerces them into an R data.frame.
SparkR::print.structField	Print a Spark StructField.
SparkR::print.structType	Print a Spark StructType.
SparkR::sparkR.init	Initialize a new Spark Context.
SparkR::sparkR.stop	Stop the Spark context.
SparkR::sparkRHive.init	Initialize a new HiveContext.
SparkR::sparkRSQL.init	Initialize a new SQLContext.
SparkR::table	Create a DataFrame from a SparkSQL Table
labeling::sparks	Sparks' labeling algorithm

Re-using existing Spark Context. Please stop SparkR with sparkR.stop() or restart R to create a new Spark Context

> read.df(sqlContext = sc, path = "hdfs://localhost:9000/input/wordcount.txt")

Error in invokeJava(isStatic = TRUE, className, methodName, ...) :

org.apache.spark.SparkException: Job aborted due to stage failure: Task 3 in stage 9.0 failed 1 times, most recent failure: Lost task 3.0 in stage 9.0 (TID 39, localhost): java.io.IOException: Could not read footer: java.lang.RuntimeException: **hdfs://localhost:9000/input/wordcount.txt is not a Parquet file**. expected magic number at tail [80, 65, 82, 49] but found [109, 112, 108, 101]

Reading and writing

Advantages of R:

- R is highly dyamic
- easy data manipulation using data frames.
- powerful visualization

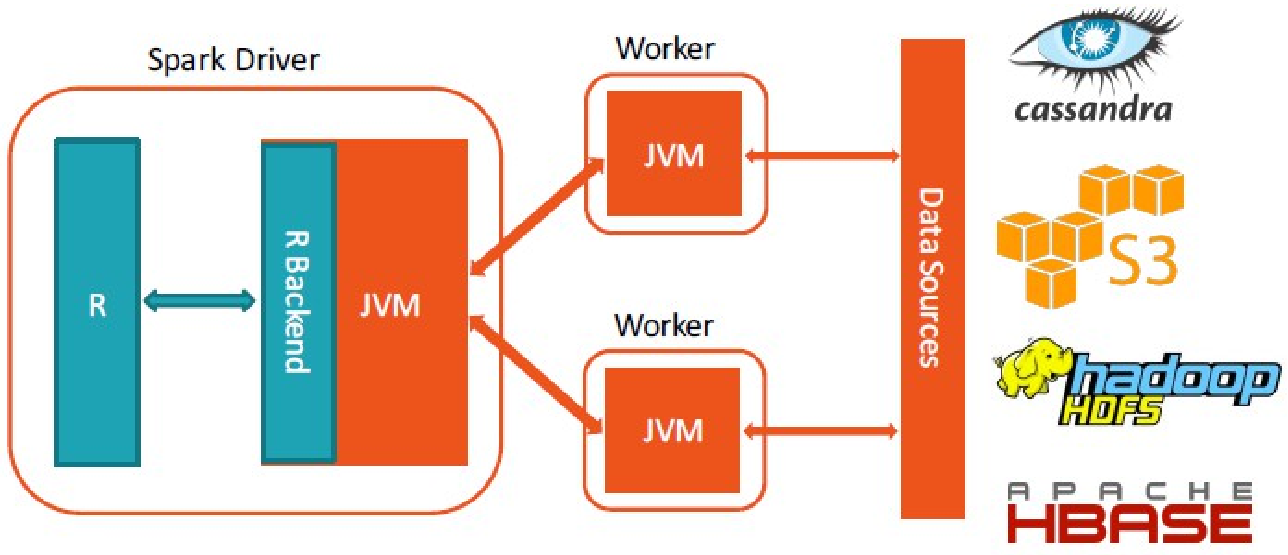
disadvantages of R:

- disadvantage is same -dynamic language features. I.e at runtime the loop variable cannot assume same datatype , every time it need to check datatype.
- Single threaded.
- Everything has to fit in to memory.

SPARKR:

- provide R front end to spark

OVERVIEW OF SPARK



Reading and writing to storage(JVM <->Storage)

1.SPARKDF ← read.df(SqlContext,path =”_” , source =”csv”)

- jsonFile:read.df(sqlContext,path=”_” ,source=”json”)
- parquetFile: read.df(sqlcontext , =” _” ,source =”parquet”)

NOTE:Above read from many source like database(e.g casandra),csv,json,

2.write.df (SparkDF,source=”json”)

saveAsparquetFile: write.df(_ , source=”parquet”)

NOTE:Above write from work to distributed storage and distributed storage to worker. Write uses df not sql context

NOTE:

As stated above **read data** from **distributed storage** to **worker** and **write** from **worker** to **distributed storage**.

To communicate with R process you need to follow below:

Moving data between R and JVM

only below commands talks with R process and jvm

- sparkDF <-createDataFrame(sqlcontext,df)
- df<- collect(sparkDF)

NOTE:Above read from many source

Caching

controls caching of distributed data

- persist(sparkDF,storagelevel)
- cache(sparkDF)
- cacheTable(sqlcontext,”tableName”)

```
library(devtools)
library(SparkR)
Sys.setenv(SPARK_HOME = "C:/spark-1.6.0-bin-hadoop2.3")
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))
Sys.getenv("SPARK_HOME")
#load the Sparkr library
library(SparkR)
# Create a spark context and a SQL context
sc <- sparkR.init(master = "local")
sqlContext <- sparkRSQL.init(sc)
sc <- sparkR.init(sparkPackages="com.databricks:spark-csv_2.11:1.0.3")
 result <- read.df(sqlContext, "/input/sales.csv", "csv")
```

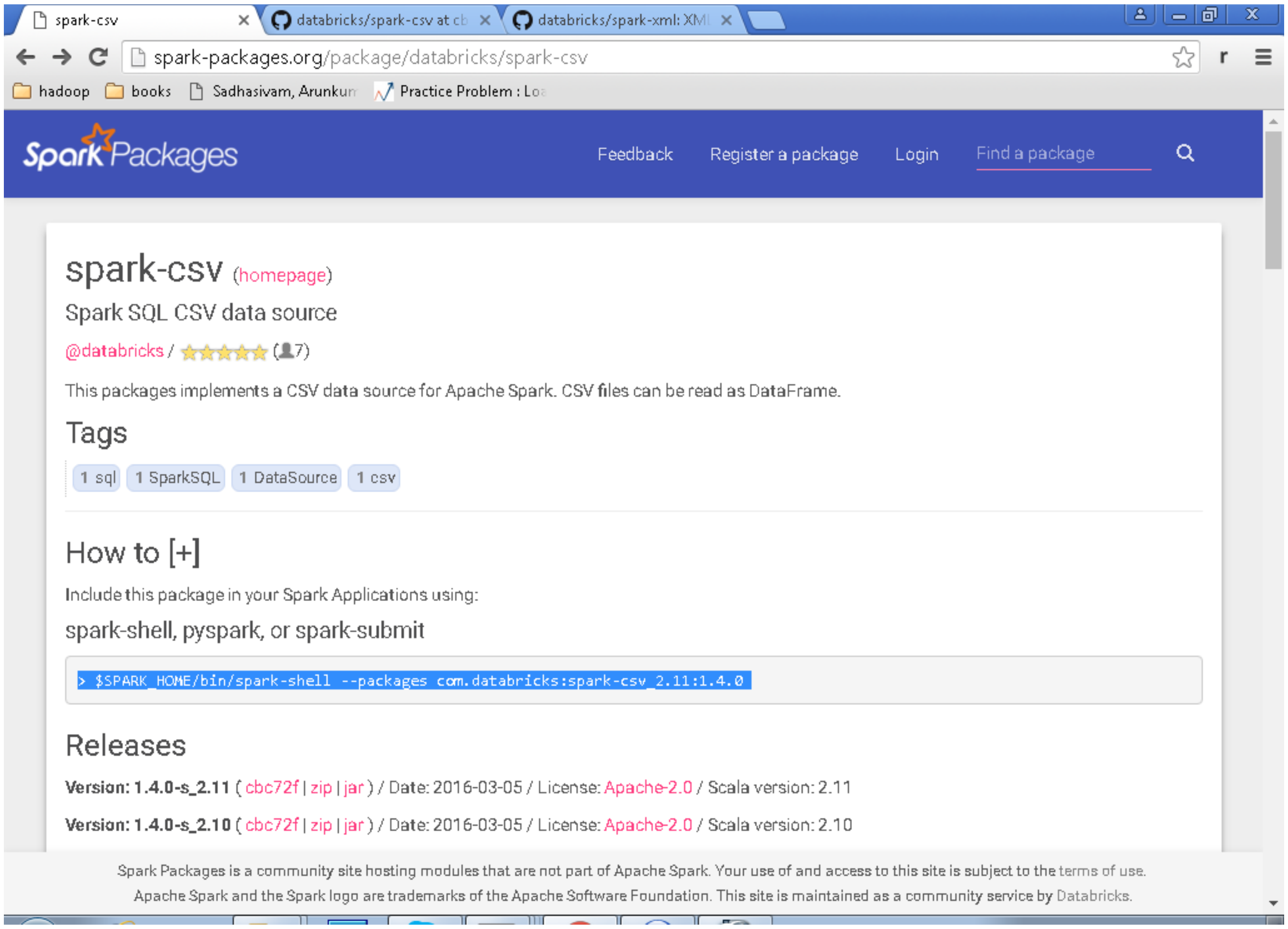
```
>  result <- read.df(sqlContext, "/input/sales.csv", "csv")
Error in invokeJava(isStatic = TRUE, className, methodName, ...) : 
  java.lang.ClassNotFoundException: Failed to find data source: csv. Please find
packages at http://spark-packages.org
```

install com.databricks:spark-csv <<version>> as needed in below command

```
> $SPARK_HOME/bin/spark-shell --packages com.databricks:spark-csv_2.11:1.0.3
```

see above command sc <- sparkR.init(sparkPackages="com.databricks:spark-csv_2.11:1.0.3")

is same as command prompt installation of spark-shell with --packages option.



spark-csv (homepage)

Spark SQL CSV data source

@databricks / ★★★★★ (7)

This packages implements a CSV data source for Apache Spark. CSV files can be read as DataFrame.

Tags

1 sql 1 SparkSQL 1 DataSource 1 csv

How to [+]

Include this package in your Spark Applications using:

spark-shell, pyspark, or spark-submit

```
> $SPARK_HOME/bin/spark-shell --packages com.databricks:spark-csv_2.11:1.4.0
```

Releases

Version: 1.4.0-s_2.11 (cbc72f | zip | jar) / Date: 2016-03-05 / License: Apache-2.0 / Scala version: 2.11

Version: 1.4.0-s_2.10 (cbc72f | zip | jar) / Date: 2016-03-05 / License: Apache-2.0 / Scala version: 2.10

Spark Packages is a community site hosting modules that are not part of Apache Spark. Your use of and access to this site is subject to the terms of use. Apache Spark and the Spark logo are trademarks of the Apache Software Foundation. This site is maintained as a community service by Databricks.

NOTE:

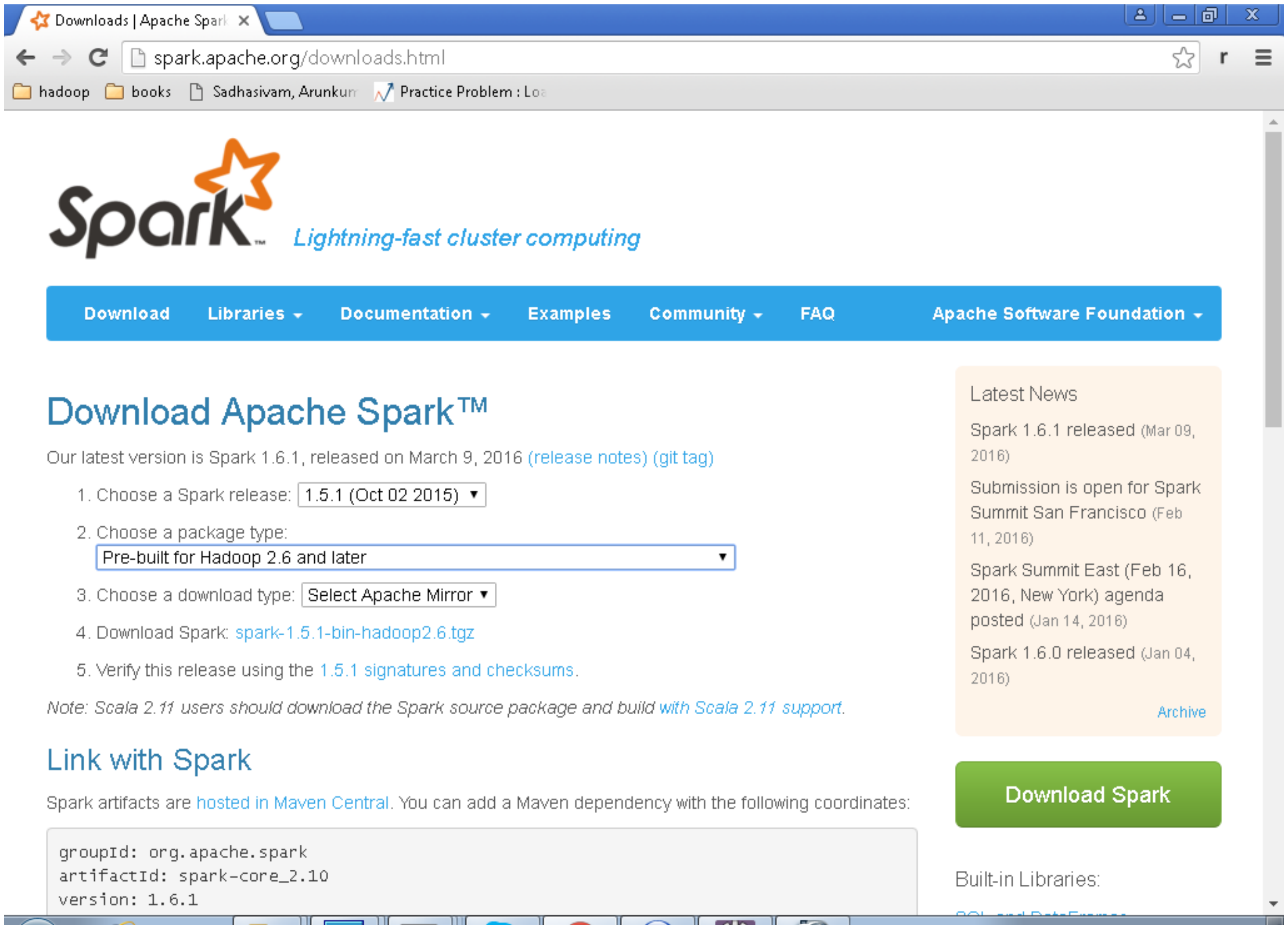
It might be worth mentioning that running `spark-1.5.1-bin-hadoop2.6/bin/spark-shell --packages com.databricks:spark-csv_2.11:1.2.0` works just fine.

Re-install with hadoop 2.6 and spark 1.5.1

STEP 1:

change the environment variable to point to 1.5.1 and close all cmd prompt

%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:\Program Files (x86)\Skype\Phone\;C:\apache-maven-3.3.9\bin;C:\protoc;C:\Program Files\Microsoft SDKs\Windows\v7.1\bin;C:\Program Files\Git\bin;C:\Java\jdk1.7.0_79\bin;C:\Anaconda2;C:\Anaconda2\Library\bin;C:\Anaconda2\Scripts;C:\Program Files\R\R-3.2.3\bin;**C:\spark-1.5.1-bin-hadoop2.6\bin;**C:\scala-2.11.7\bin;C:\hadoop-2.2.0\bin;C:\hadoop-2.2.0\sbin;C:\apache-mahout-distribution-0.10.2\bin;C:\pig-0.15.0\bin;C:\apache-hive-2.0.0-bin\bin;C:\zeppelin-0.5.5\bin



Download Apache Spark™

Our latest version is Spark 1.6.1, released on March 9, 2016 (release notes) (git tag)

1. Choose a Spark release: 1.5.1 (Oct 02 2015)
2. Choose a package type: Pre-built for Hadoop 2.6 and later
3. Choose a download type: Select Apache Mirror
4. Download Spark: spark-1.5.1-bin-hadoop2.6.tgz
5. Verify this release using the 1.5.1 signatures and checksums.

Note: Scala 2.11 users should download the Spark source package and build with Scala 2.11 support.

Link with Spark

Spark artifacts are hosted in Maven Central. You can add a Maven dependency with the following coordinates:

```
groupId: org.apache.spark
artifactId: spark-core_2.10
version: 1.6.1
```

Latest News

Spark 1.6.1 released (Mar 09, 2016)

Submission is open for Spark Summit San Francisco (Feb 11, 2016)

Spark Summit East (Feb 16, 2016, New York) agenda posted (Jan 14, 2016)

Spark 1.6.0 released (Jan 04, 2016)

Archive

STEP 2:

Try running java , see below it stills uses 1.6.1.

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/04/12 22:36:28 INFO SparkContext: Running Spark version 1.6.1
16/04/12 22:36:29 INFO SecurityManager: Changing view acls to: Arun
16/04/12 22:36:29 INFO SecurityManager: Changing modify acls to: Arun
16/04/12 22:36:29 INFO SecurityManager: SecurityManager: authentication disabled; ui acls
disabled; users with view permissions: Set(Arun); users with modify permissions: Set(Arun)
16/04/12 22:36:30 INFO Utils: Successfully started service 'sparkDriver' on port 58098.
16/04/12 22:36:31 INFO Slf4jLogger: Slf4jLogger started
16/04/12 22:36:31 INFO Remoting: Starting remoting
```

change the below env entry-SPARK_JAR

C:\spark-1.5.1-bin-hadoop2.6\lib\spark-assembly-1.5.1-hadoop2.6.0.jar

change pom.xml

```
<dependencies>

        <dependency>

                <groupId>org.apache.spark</groupId>

                <artifactId>spark-streaming_2.10</artifactId>

                <version>1.5.1</version>

        </dependency>

</dependencies>
```

check java output and version

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/04/12 22:45:34 INFO SparkContext: Running Spark version 1.5.1
16/04/12 22:45:35 INFO SecurityManager: Changing view acls to: Arun
16/04/12 22:45:35 INFO SecurityManager: Changing modify acls to: Arun
16/04/12 22:45:35 INFO SecurityManager: SecurityManager: authentication disabled; ui acls
disabled; users with view permissions: Set(Arun); users with modify permissions: Set(Arun)
16/04/12 22:45:36 INFO Slf4jLogger: Slf4jLogger started
16/04/12 22:45:36 INFO Remoting: Starting remoting
16/04/12 22:45:36 INFO Remoting: Remoting started; listening on addresses :
[akka.tcp://sparkDriver@localhost:58361]
16/04/12 22:45:36 INFO Utils: Successfully started service 'sparkDriver' on port 58361.
16/04/12 22:45:41 INFO DAGScheduler: Job 0 finished: collect at JavaWordCountHdfs.java:76,
took 0.585637 s
example: 1
are: 1
```

```
is: 1
you: 1
wordcount: 1
hadoop: 3
hi: 3
how: 2
16/04/12 22:45:41 INFO SparkUI: Stopped Spark web UI at http://localhost:4040
16/04/12 22:45:41 INFO DAGScheduler: Stopping DAGScheduler
16/04/12 22:45:41 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint
stopped!
```

Now try running sparkR R program to Access HDFS:

```
c:\spark-1.5.1-bin-hadoop2.6/bin>  spark-shell --packages com.databricks:spark-csv_2.11:1.2.0
```

Note: **on running the above command no error is showing.**

16/04/12 22:54:11 INFO ui.SparkUI: Started SparkUI at http://localhost:4040

16/04/12 22:54:11 INFO spark.SparkContext: **Added JAR file:/C:/Users/Arun/.ivy2/j**

ars/com.databricks_spark-csv_2.11-1.2.0.jar at http://localhost:54649/jars/com.d

atabricks_spark-csv_2.11-1.2.0.jar with timestamp 1460481851695

16/04/12 22:54:11 INFO spark.SparkContext: **Added JAR file:/C:/Users/Arun/.ivy2/j**

ars/org.apache.commons_commons-csv-1.1.jar at http://localhost:54649/jars/org.ap

ache.commons_commons-csv-1.1.jar with timestamp 1460481851701

16/04/12 22:54:11 INFO spark.SparkContext: **Added JAR file:/C:/Users/Arun/.ivy2/j**

ars/com.univocity_univocity-parsers-1.5.1.jar at http://localhost:54649/jars/com

.univocity_univocity-parsers-1.5.1.jar with timestamp 1460481851706

16/04/12 22:54:11 WARN metrics.MetricsSystem: Using default name DAGScheduler fo

r source because spark.app.id is not set.

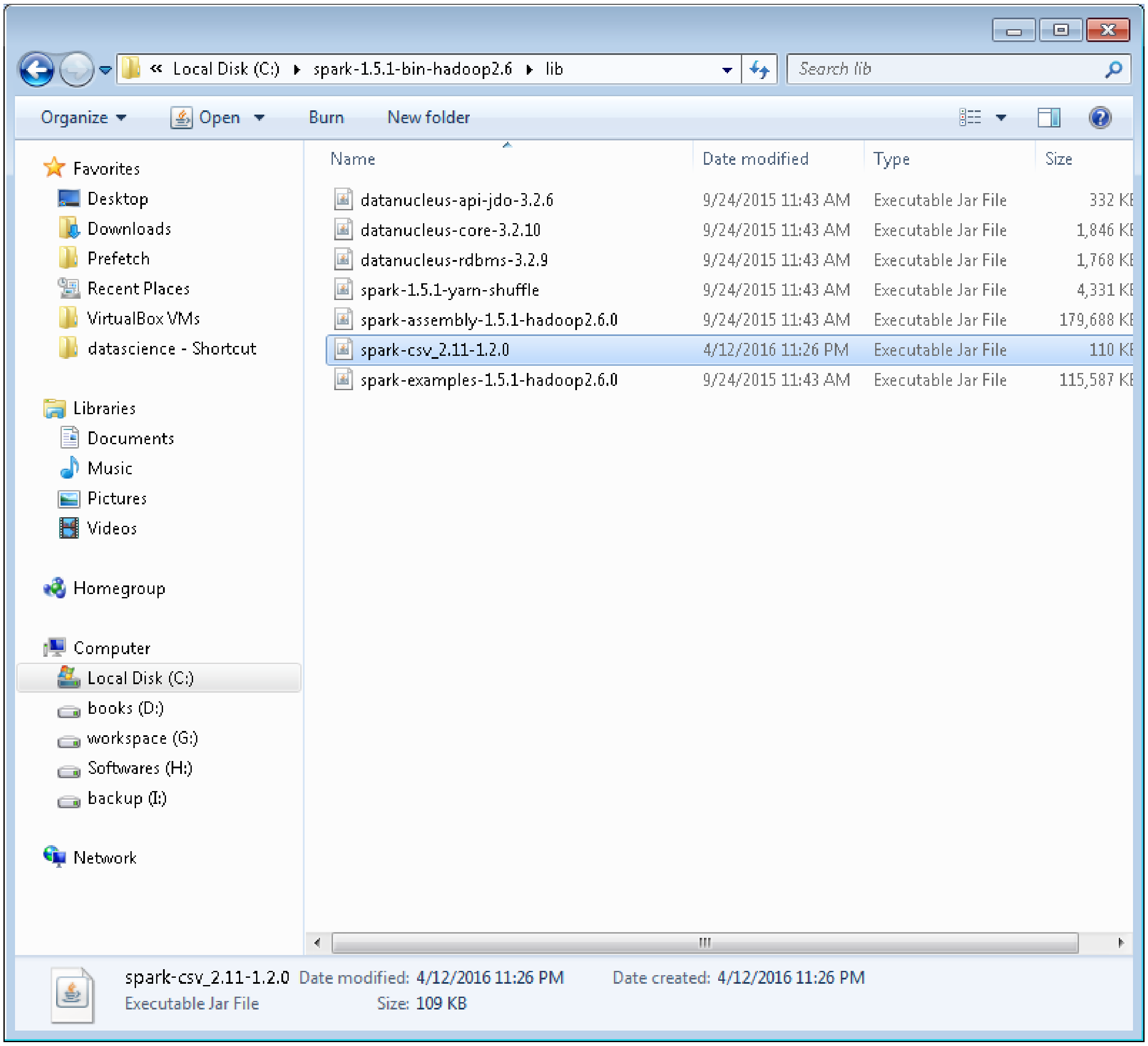
NOTE:

See all jar are automatically deployed to sucessfully when running below command

```
spark-shell --packages com.databricks:spark-csv_2.11:1.2.0
```

both jar spark-csv_2.11-1.2.0 downloaded should match the jar add to class path in SparkR programming.

Add spark csv (databricks) to class path of R programming.

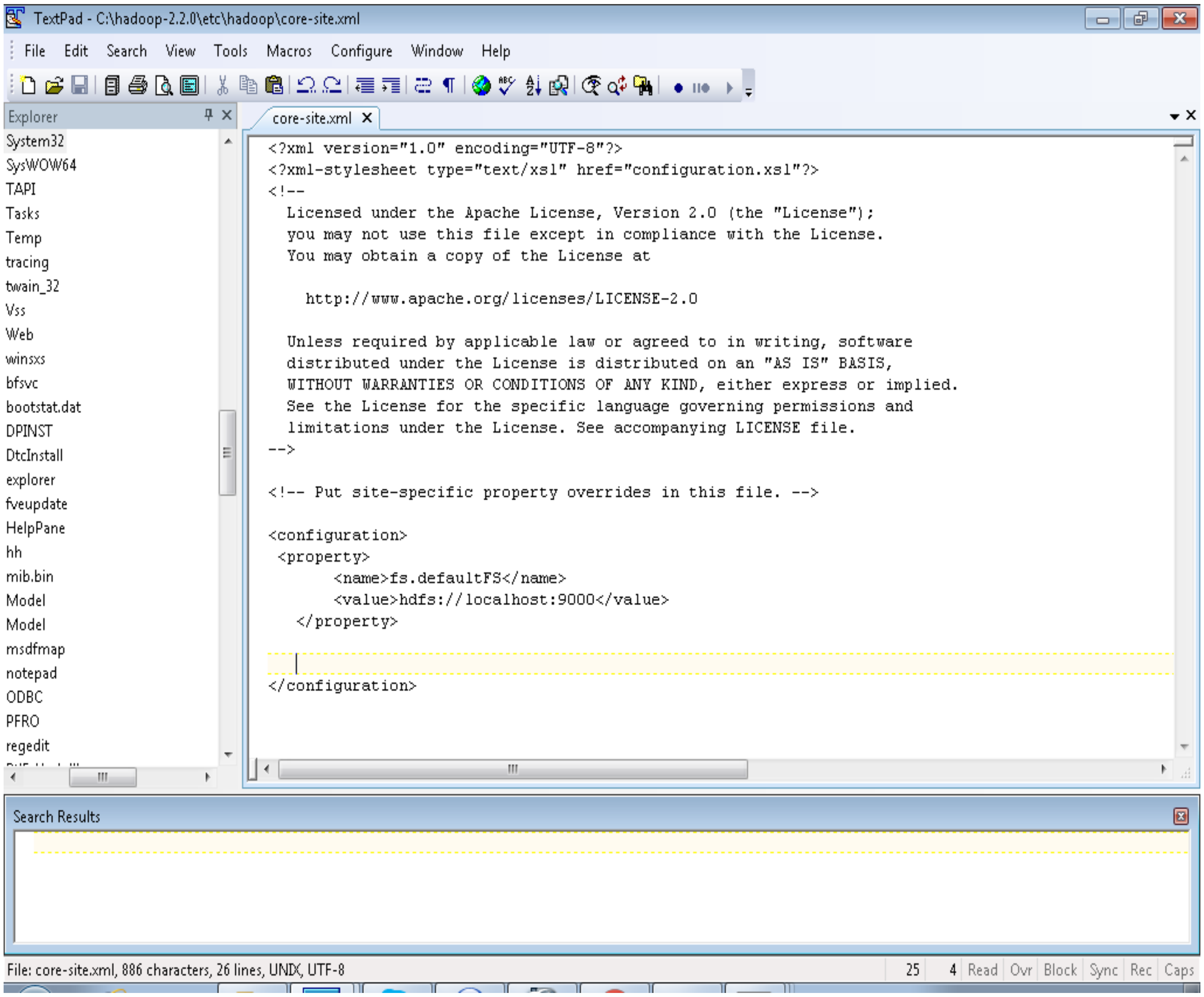


Name	Date modified	Type	Size
datanucleus-api-jdo-3.2.6	9/24/2015 11:43 AM	Executable Jar File	332 KB
datanucleus-core-3.2.10	9/24/2015 11:43 AM	Executable Jar File	1,846 KB
datanucleus-rdbms-3.2.9	9/24/2015 11:43 AM	Executable Jar File	1,768 KB
spark-1.5.1-yarn-shuffle	9/24/2015 11:43 AM	Executable Jar File	4,331 KB
spark-assembly-1.5.1-hadoop2.6.0	9/24/2015 11:43 AM	Executable Jar File	179,688 KB
spark-csv_2.11-1.2.0	4/12/2016 11:26 PM	Executable Jar File	110 KB
spark-examples-1.5.1-hadoop2.6.0	9/24/2015 11:43 AM	Executable Jar File	115,587 KB

spark-csv_2.11-1.2.0 Date modified: 4/12/2016 11:26 PM Date created: 4/12/2016 11:26 PM
Executable Jar File Size: 109 KB

```
library(devtools)

library(SparkR)

Sys.setenv(SPARK_HOME = "C:/spark-1.5.1-bin-hadoop2.6")

.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))

Sys.getenv("SPARK_HOME")

#load the Sparkr library

library(SparkR)

# Create a spark context and a SQL context

sc <- sparkR.init(master = "local")

sqlContext <- sparkRSQL.init(sc)

sc <- sparkR.init(sparkPackages="com.databricks:spark-csv_2.11:1.2.0")

result <- read.df(sqlContext, "hdfs://localhost:9000/input/sales.csv")
```

```
Error in invokeJava(isStatic = TRUE, className, methodName, ...) :
  org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in
stage 0.0 failed 1 times, most recent failure: Lost task 0.0 in stage 0.0 (TID 0,
localhost): java.io.IOException: Could not read footer:
java.lang.UnsatisfiedLinkError:
org.apache.hadoop.util.NativeCrc32.nativeComputeChunkedSums(IILjava/nio/ByteBuffer
;ILjava/nio/ByteBuffer;IILjava/lang/String;JZ)V
        at
org.apache.parquet.hadoop.ParquetFileReader.readAllFootersInParallel(ParquetFileRe
ader.java:247)
        at org.apache.spark.sql.execution.datasources.parquet.ParquetRelation$
$anonfun$28.apply(ParquetRelation.scala:750)
        at org.apache.spark.sql.execution.datasources.parquet.ParquetRelation$
$anonfun$28.apply(ParquetRelation.scala:739)
        at org.apache.spark.rdd.RDD$$anonfun$mapPartitions$1$
$anonfun$apply$17.apply(RDD.scala:706)
        at org.apache.spark.rdd.RDD$$anonfun$mapPartitions$1$
$anonfun$apply$17.apply(RDD.scala:706)
        at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
```


TextPad - C:\hadoop-2.2.0\etc\hadoop\core-site.xml
File Edit Search View Tools Macros Configure Window Help
Explorer
System32
SysWOW64
TAPI
Tasks
Temp
tracing
twain_32
Vss
Web
winsxs
bfsvc
bootstat.dat
DPINST
DtcInstall
explorer
fveupdate
HelpPane
hh
mib.bin
Model
Model
msdfmap
notepad
ODBC
PFRO
regedit
core-site.xml
Search Results
File: core-site.xml, 886 characters, 26 lines, UNIX, UTF-8
25 4 Read Ovr Block Sync Rec Caps

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
 <property>
       <name>fs.defaultFS</name>
       <value>hdfs://localhost:9000</value>
   </property>


</configuration>
```

NOTE:

hdfs root path in R should be same as given in core-site.xml

e.g result <- read.df(sqlContext, "hdfs://localhost:9000/input/sales.csv")