Hadoop Training RStudio and Apache Spark

Working with Apache RStudio and Apache Spark

During this exercise, you will become an Apache Spark Developer/Analyst who works with RStudio and sparklyr package.

Getting ready

1. Please use your Linux account at the edge node (cdh00.cl.ii.pw.edu.pl)

2. Launch a web browser and navigate to http://cdh00.cl.ii.pw.edu.pl:8788/ and use your Linux account to login to RStudio Server.

3. Start a new connection to the Apache Spark cluster in yarn-client mode.

You will allocate 3 Spark executors and configure the necessary environment variables, memory settings for both driver and executors. You will use sparklyr package for interacting with the cluster:

```
#load packages
library(dplyr)
library(sparklyr)


#set necessary environment variables


#create a Spark configuration
config <- sparklyr::spark_config()
config$spark.driver.cores <- 1
config$spark.executor.instances <- 3
config$spark.driver.memory <- "1g"
config$spark.executor.memory <- "2g"
config$spark.ui.port <-9502

#create a spark context and connect to YARN
sc <- sparklyr::spark_connect(master="yarn-client", config=config)
```

4. Once you are connected you should be able to see a new “Spark” tab in the panel on the right side of your RStudio window:

5. Read the CSV you used in a Hive/ Spark exercise before and cache it in memory.

```
>md_remote<-spark_read_csv(sc, name="measured_data",
path="hdfs:////data/local/datascience/measured_data/measured_data.csv",
header=FALSE,memory=TRUE,delimiter = "|")

> md_remote
Source:   query [7.2e+06 x 8]
Database: spark connection master=yarn-client app=sparklyr local=FALSE

# A tibble: 7.2e+06 x 8
      V1                  V2    V3         V4    V5    V6    V7
V8
   <int>               <chr> <int>      <dbl> <chr> <chr> <chr>
<chr>
 1     1 2015-05-18 19:24:00.0   236 0.09920930    kW     s     D
Warsaw-Dereniowa
 2     2 2015-05-18 19:24:00.0   237 0.84677513    kW     s     D
Warsaw-Dereniowa
 3     3 2015-05-18 19:24:00.0   238 0.46089532    kW     s     D
Warsaw-Dereniowa
 4     4 2015-05-18 19:24:00.0   239 0.02923115    kW     s     D
Warsaw-Dereniowa
 5     5 2015-05-18 19:24:00.0   240 0.76983756    kW     s     D
Warsaw-Dereniowa
 6     6 2015-05-18 19:24:00.0   241 0.77489693    kW     s     D
Warsaw-Dereniowa
 7     7 2015-05-18 19:24:00.0   242 0.20359460    kW     s     D
Warsaw-Dereniowa
 8     8 2015-05-18 19:24:00.0   243 0.45276429    kW     s     D
Warsaw-Dereniowa
 9     9 2015-05-18 19:24:00.0   244 0.42352181    kW     s     D
Warsaw-Dereniowa
10    10 2015-05-18 19:24:00.0   245 0.97955139    kW     s     D
Warsaw-Dereniowa
# ... with 7.2e+06 more rows



# Source: spark<measured_data> [?? x 8]
      V1 V2                     V3     V4 V5    V6    V7    V8
```

```
   <int> <dttm>              <int>  <dbl> <chr> <chr> <chr> <chr>
 1     1 2015-05-18 17:24:00   236 0.0992 kW    s     D
Warsaw-Dereniowa
 2     2 2015-05-18 17:24:00   237 0.847  kW    s     D
Warsaw-Dereniowa
 3     3 2015-05-18 17:24:00   238 0.461  kW    s     D
Warsaw-Dereniowa
 4     4 2015-05-18 17:24:00   239 0.0292 kW    s     D
Warsaw-Dereniowa
 5     5 2015-05-18 17:24:00   240 0.770  kW    s     D
Warsaw-Dereniowa
 6     6 2015-05-18 17:24:00   241 0.775  kW    s     D
Warsaw-Dereniowa
 7     7 2015-05-18 17:24:00   242 0.204  kW    s     D
Warsaw-Dereniowa
 8     8 2015-05-18 17:24:00   243 0.453  kW    s     D
Warsaw-Dereniowa
 9     9 2015-05-18 17:24:00   244 0.424  kW    s     D
Warsaw-Dereniowa
10    10 2015-05-18 17:24:00   245 0.980  kW    s     D
Warsaw-Dereniowa
# … with more rows
```

6. Perform **some basic operations on the Spark DataFrame**

```
#row count
>md_remote %>% count()
# Source: spark<?> [?? x 1]
        n
    <dbl>
1 7200000


#filter
> md_remote %>% filter(V1==8)
Source:   query [7.2e+04 x 8]
Database: spark connection master=yarn-client app=sparklyr local=FALSE

# A tibble: 7.2e+04 x 8
      V1                    V2    V3         V4    V5    V6    V7
V8
   <int>                 <chr> <int>      <dbl> <chr> <chr> <chr>
<chr>
 1     8 2015-05-18 19:24:00.0   243 0.45276429    kW     s     D
Warsaw-Dereniowa
 2     8 2015-05-18 19:25:00.0   243 0.37307671    kW     s     D
Warsaw-Dereniowa
 3     8 2015-05-18 19:26:00.0   243 0.23607370    kW     s     D
Warsaw-Dereniowa
 4     8 2015-05-18 19:27:00.0   243 0.06600368    kW     s     D
Warsaw-Dereniowa
 5     8 2015-05-18 19:28:00.0   243 0.40257417    kW     s     D
Warsaw-Dereniowa
 6     8 2015-05-18 19:29:00.0   243 0.25740322    kW     s     D
Warsaw-Dereniowa
 7     8 2015-05-18 19:30:00.0   243 0.69475873    kW     s     D
Warsaw-Dereniowa
 8     8 2015-05-18 19:31:00.0   243 0.43615179    kW     s     D
Warsaw-Dereniowa
 9     8 2015-05-18 19:32:00.0   243 0.44010886    kW     s     D
Warsaw-Dereniowa
10     8 2015-05-18 19:33:00.0   243 0.11645484    kW     s     D
Warsaw-Dereniowa
# ... with 7.199e+04 more rows



#get first 10 rows into a local data frame
> md_local <- md_remote %>% head(10) %>% collect
> md_local
# A tibble: 10 x 8
      V1                    V2    V3         V4    V5    V6    V7
V8
   <int>                 <chr> <int>      <dbl> <chr> <chr> <chr>
<chr>
```

```
 1     1 2015-05-18 19:24:00.0   236 0.09920930    kW     s     D
Warsaw-Dereniowa
 2     2 2015-05-18 19:24:00.0   237 0.84677513    kW     s     D
Warsaw-Dereniowa
 3     3 2015-05-18 19:24:00.0   238 0.46089532    kW     s     D
Warsaw-Dereniowa
 4     4 2015-05-18 19:24:00.0   239 0.02923115    kW     s     D
Warsaw-Dereniowa
 5     5 2015-05-18 19:24:00.0   240 0.76983756    kW     s     D
Warsaw-Dereniowa
 6     6 2015-05-18 19:24:00.0   241 0.77489693    kW     s     D
Warsaw-Dereniowa
 7     7 2015-05-18 19:24:00.0   242 0.20359460    kW     s     D
Warsaw-Dereniowa
 8     8 2015-05-18 19:24:00.0   243 0.45276429    kW     s     D
Warsaw-Dereniowa
 9     9 2015-05-18 19:24:00.0   244 0.42352181    kW     s     D
Warsaw-Dereniowa
10    10 2015-05-18 19:24:00.0   245 0.97955139    kW     s     D
Warsaw-Dereniowa


#compare a Spark and R dataframe
> class(md_local)
[1] "tbl_df"     "tbl"        "data.frame"

> class(md_remote)
[1] "tbl_spark" "tbl_sql"   "tbl_lazy"  "tbl"
>

#create a new Spark Dataframe with only 10 rows using md_local
DataFrame

> md_remote10<-copy_to(sc,md_local,"md_remote10")

> md_remote10
Source:   query [10 x 8]
Database: spark connection master=yarn-client app=sparklyr local=FALSE

# A tibble: 10 x 8
      V1                    V2    V3         V4    V5    V6    V7
V8
   <int>                 <chr> <int>      <dbl> <chr> <chr> <chr>
<chr>
 1     1 2015-05-18 19:24:00.0   236 0.09920930    kW     s     D
Warsaw-Dereniowa
 2     2 2015-05-18 19:24:00.0   237 0.84677513    kW     s     D
Warsaw-Dereniowa
 3     3 2015-05-18 19:24:00.0   238 0.46089532    kW     s     D
Warsaw-Dereniowa
```

```
 4     4 2015-05-18 19:24:00.0   239 0.02923115    kW      s      D
Warsaw-Dereniowa
 5     5 2015-05-18 19:24:00.0   240 0.76983756    kW      s      D
Warsaw-Dereniowa
 6     6 2015-05-18 19:24:00.0   241 0.77489693    kW      s      D
Warsaw-Dereniowa
 7     7 2015-05-18 19:24:00.0   242 0.20359460    kW      s      D
Warsaw-Dereniowa
 8     8 2015-05-18 19:24:00.0   243 0.45276429    kW      s      D
Warsaw-Dereniowa
 9     9 2015-05-18 19:24:00.0   244 0.42352181    kW      s      D
Warsaw-Dereniowa
10    10 2015-05-18 19:24:00.0   245 0.97955139    kW      s      D
Warsaw-Dereniowa
```

7. Try to do the exercise available at https://beta.rstudioconnect.com/content/1518/notebook-classification.html

with the following changes in 3 steps (for the rest of the code snippets please refer to the web page):

1. Load the data step – you just need to run the following code (provided you already have your sc – SparkContext available in RStudio – if not then just come back to the 2. step of this exercise):

```
library(tidyr)
library(titanic)
library(ggplot2)
library(purrr)

titanic_tbl <- copy_to(sc, titanic::titanic_train, "titanic", overwrite =
TRUE)

titanic2_tbl <- titanic_tbl %>%
    mutate(Family_Size = SibSp + Parch + 1L) %>%
    mutate(Pclass = as.character(Pclass)) %>%
    filter(!is.na(Embarked)) %>%
    filter(length(Embarked) > 0)%>%
    mutate(Age = if_else(is.na(Age), mean(Age), Age)) %>%
      select(Survived, Pclass, Sex, Age, SibSp, Parch, Fare, Embarked,
Family_Size) %>%
    sdf_register("titanic2")
```

2. Other ML algorithms – here we exclude neural networks

```
## Decision Tree
ml_dt <- ml_decision_tree(train_tbl, ml_formula)

## Random Forest
ml_rf <- ml_random_forest(train_tbl, ml_formula)

## Gradient Boosted Tree
ml_gbt <- ml_gradient_boosted_trees(train_tbl, ml_formula)

## Naive Bayes
ml_nb <- ml_naive_bayes(train_tbl, ml_formula)
```

3. Validation data – again exclude neural networks

```
ml_models <- list(
  "Logistic" = ml_log,
  "Decision Tree" = ml_dt,
  "Random Forest" = ml_rf,
  "Gradient Boosted Trees" = ml_gbt,
  "Naive Bayes" = ml_nb
)

# Create a function for scoring
score_test_data <- function(model, data=test_tbl){
  pred <- sdf_predict(model, data)
  select(pred, Survived, prediction)
}
```

 # Score all the models

```
      ml_score <- lapply(ml_models, score_test_data)
```