Spark with R Example-2

siju.swamy@saintgits.org

2023-01-17

Introduction to Spark in R using `sparklyr`

- Apache Spark is a unified analytics engine for large-scale data processing.
- It is an open source cluster computing platform.
- Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Experimenting with `spark` in R

- Spark also supports a pseudo-distributed local mode
- in such a scenario, Spark is run on a single machine with one executor per CPU core.
- `sparklyr` is an R package that lets you write R code to work with data in a Spark cluster.

Installing spark from R

We then need to install Spark, which we can do from R.

```
#spark_install()
```

Connect to `spark`

The typical workflow has three steps:

- Connect to Spark using `spark_connect()`.
- Do some work.
- Close the connection to Spark using `spark_disconnect()`.

`spark_connect()` takes a URL that gives the location to `Spark`.

```
# Load sparklyr
library(sparklyr)
```

```
## Warning: package 'sparklyr' was built under R version 4.2.2
```

```
##
## Attaching package: 'sparklyr'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
# # install a local version of Spark for development purposes (only once!)
# spark_install()

# set Java home to Java 8 (only working with Java 8 at the moment)
java_path <- normalizePath('C:/Progra~1/Java/jre1.8.0_201')
Sys.setenv(JAVA_HOME=java_path)

# Connect to your Spark cluster
sc <- spark_connect("local")

# Print the version of Spark
spark_version(sc)
```

```
## [1] '2.4.3'
```

Copying data to spark

```
# copy data: create a Spark table flights and airlines
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
flights_tbl <- copy_to(sc, nycflights13::flights, "flights")
#airlines_tbl <- copy_to(sc, nycflights13::airlines, "airlines")
# show tables
src_tbls(sc)
```

```
## [1] "flights"
```

```
# Link to the track_metadata table in Spark
flights_tbl <- tbl(sc, "flights")

# which class it belongs to
class(flights_tbl)
```

```
## [1] "tbl_spark" "tbl_sql"   "tbl_lazy"  "tbl"
```

Viewing flights table

```
# look inside flights Spark table
flights_tbl
```

```
## # Source: spark<flights> [?? x 19]
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # ... with more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

Viewing dimension of data in the spark handle

```
# how big the dataset is (we don't know in fact!)
dim(flights_tbl)
```

```
## [1] NA 19
```

Running query on spark dataframe

The easiest way to manipulate data frames stored in Spark is to use dplyr syntax.

```
flight_delay <-
  flights_tbl %>%
  group_by(tailnum) %>%
  summarise(count = n(),
```

```
            dist = mean(distance, na.rm = TRUE),
            delay = mean(arr_delay, na.rm = TRUE)) %>%
  mutate(delay_by_distance = delay / dist) %>%
  filter(count > 20, dist < 2000, !is.na(delay)) %>%
  arrange(desc(delay_by_distance))

flight_delay
```

```
## # Source:     spark<?> [?? x 5]
## # Ordered by: desc(delay_by_distance)
##    tailnum count  dist delay delay_by_distance
##    <chr>   <dbl> <dbl> <dbl>             <dbl>
##  1 N645MQ     25  480.  51              0.106
##  2 N832AS    163  228.  23.4            0.102
##  3 N8475B     35  326.  32.4            0.0993
##  4 N8683B     42  385.  35.8            0.0930
##  5 N835AS    194  228.  20.1            0.0881
##  6 N828AS    208  228.  20.0            0.0875
##  7 N8646A     38  353.  30.1            0.0852
##  8 N942MQ     44  462.  38.3            0.0830
##  9 N834AS    173  229.  18.7            0.0820
## 10 N908MQ     22  472   38.5            0.0816
## # ... with more rows
```

```
# showing selected columns only

select(flights_tbl, year:day, arr_delay, dep_delay)
```

```
## # Source: spark<?> [?? x 5]
##     year month   day arr_delay dep_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1        11         2
##  2  2013     1     1        20         4
##  3  2013     1     1        33         2
##  4  2013     1     1       -18        -1
##  5  2013     1     1       -25        -6
##  6  2013     1     1        12        -4
##  7  2013     1     1        19        -5
##  8  2013     1     1       -14        -3
##  9  2013     1     1        -8        -3
## 10  2013     1     1         8        -2
## # ... with more rows
```

```
filter(flights_tbl, dep_delay > 1000)
```

```
## # Source: spark<?> [?? x 19]
##    year month   day dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013     1     9      641         900    1301    1242    1530    1272 HA
## 2  2013     1    10     1121        1635    1126    1239    1810    1109 MQ
## 3  2013     6    15     1432        1935    1137    1607    2120    1127 MQ
## 4  2013     7    22      845        1600    1005    1044    1815     989 MQ
```

```
## 5  2013     9    20     1139        1845    1014    1457    2210    1007 AA
## # ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

```
# sorting based on departure delay
arrange(flights_tbl, desc(dep_delay))
```

```
## # Source:     spark<?> [?? x 19]
## # Ordered by: desc(dep_delay)
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     9      641        900    1301    1242    1530    1272 HA
##  2  2013     6    15     1432       1935    1137    1607    2120    1127 MQ
##  3  2013     1    10     1121       1635    1126    1239    1810    1109 MQ
##  4  2013     9    20     1139       1845    1014    1457    2210    1007 AA
##  5  2013     7    22      845       1600    1005    1044    1815     989 MQ
##  6  2013     4    10     1100       1900     960    1342    2211     931 DL
##  7  2013     3    17     2321        810     911     135    1020     915 DL
##  8  2013     6    27      959       1900     899    1236    2226     850 DL
##  9  2013     7    22     2257        759     898     121    1026     895 DL
## 10  2013    12     5      756       1700     896    1058    2020     878 AA
## # ... with more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```
# finding average departure delay based on the data
summarise(
  flights_tbl,
  mean_dep_delay = mean(dep_delay, na.rm = TRUE)
  )
```

```
## # Source: spark<?> [?? x 1]
##   mean_dep_delay
##            <dbl>
## 1           12.6
```

```
#creating a new variable "speed" with in the spark data table
mutate(flights_tbl, speed = distance / air_time * 60)
```

```
## # Source: spark<?> [?? x 20]
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
```

```
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # ... with more rows, 10 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, speed <dbl>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

Creating a new data frame within spark

```
c4 <- flights_tbl %>%
  filter(month == 5, day == 17, carrier %in% c('UA', 'WN', 'AA', 'DL')) %>%
  select(carrier, dep_delay, air_time, distance) %>%
  mutate(air_time_hours = air_time / 60) %>%
  arrange(carrier)
```

Summarizing the airline data over carrier

```
flights_tbl %>%
  group_by(carrier) %>%
  summarize(
    count = n(),
    mean_dep_delay = mean(dep_delay, na.rm = FALSE)
    )
```

```
## Warning: Missing values are always removed in SQL aggregation functions.
## Use 'na.rm = TRUE' to silence this warning
## This warning is displayed once every 8 hours.
```

```
## # Source: spark<?> [?? x 3]
##    carrier count mean_dep_delay
##    <chr>   <dbl>          <dbl>
##  1 EV      54173          20.0
##  2 US      20536           3.78
##  3 WN      12275          17.7
##  4 VX       5162          12.9
##  5 YV        601          19.0
##  6 UA      58665          12.1
##  7 DL      48110           9.26
##  8 MQ      26397          10.6
##  9 OO         32          12.6
## 10 B6      54635          13.0
## # ... with more rows
```

Accessing Data stored in Spark

- `copy_to()` moves your data from R to Spark

- `collect()` moves your data from Spark to R

  collecting data from `spark` for analysis

```
collected_flight_delay <- flight_delay %>%collect()
class(flight_delay) # return the class of the spark object
```

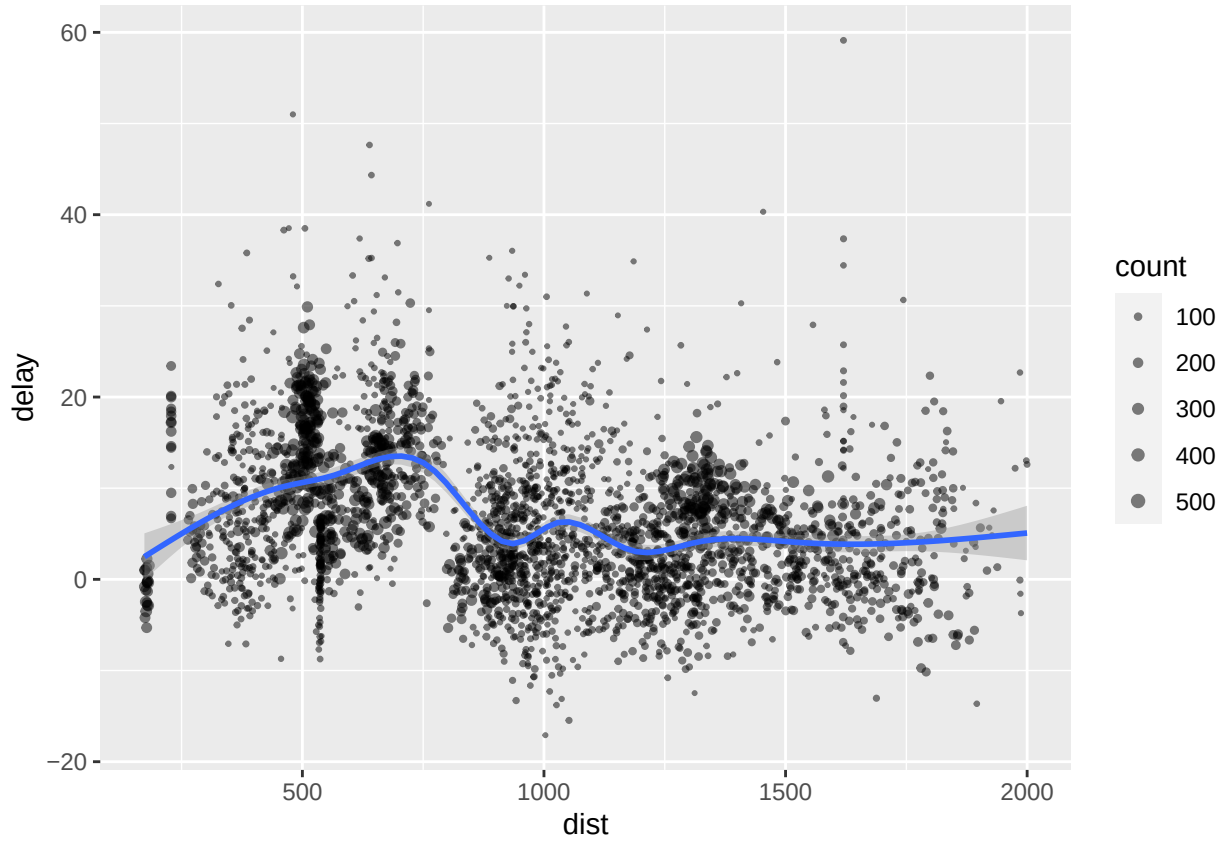
```
## [1] "tbl_spark" "tbl_sql"   "tbl_lazy"  "tbl"
```

```
class(collected_flight_delay)# return the class of the R object
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Creating visualizations

```
library(ggplot2)
ggplot(collected_flight_delay, aes(dist, delay)) +
  geom_point(aes(size = count), alpha = 1/2) +
  geom_smooth() +
  scale_size_area(max_size = 2)
```



Comparing processing time

```
system.time(
  ggplot(flight_delay, aes(dist, delay)) +
  geom_point(aes(size = count), alpha = 1/2) +
  geom_smooth() +
  scale_size_area(max_size = 2)
)
```

```
##    user  system elapsed
##    0.05    0.00    0.97
```

```
system.time(
  ggplot(collected_flight_delay, aes(dist, delay)) +
  geom_point(aes(size = count), alpha = 1/2) +
  geom_smooth() +
  scale_size_area(max_size = 2)
)
```

```
##    user  system elapsed
##       0       0       0
```

Few more data analysis tasks with `spark`

```
carrierhours <- collect(c4)
```

`collect()` executes the Spark query and returns the results to `R` for further analysis and visualization.

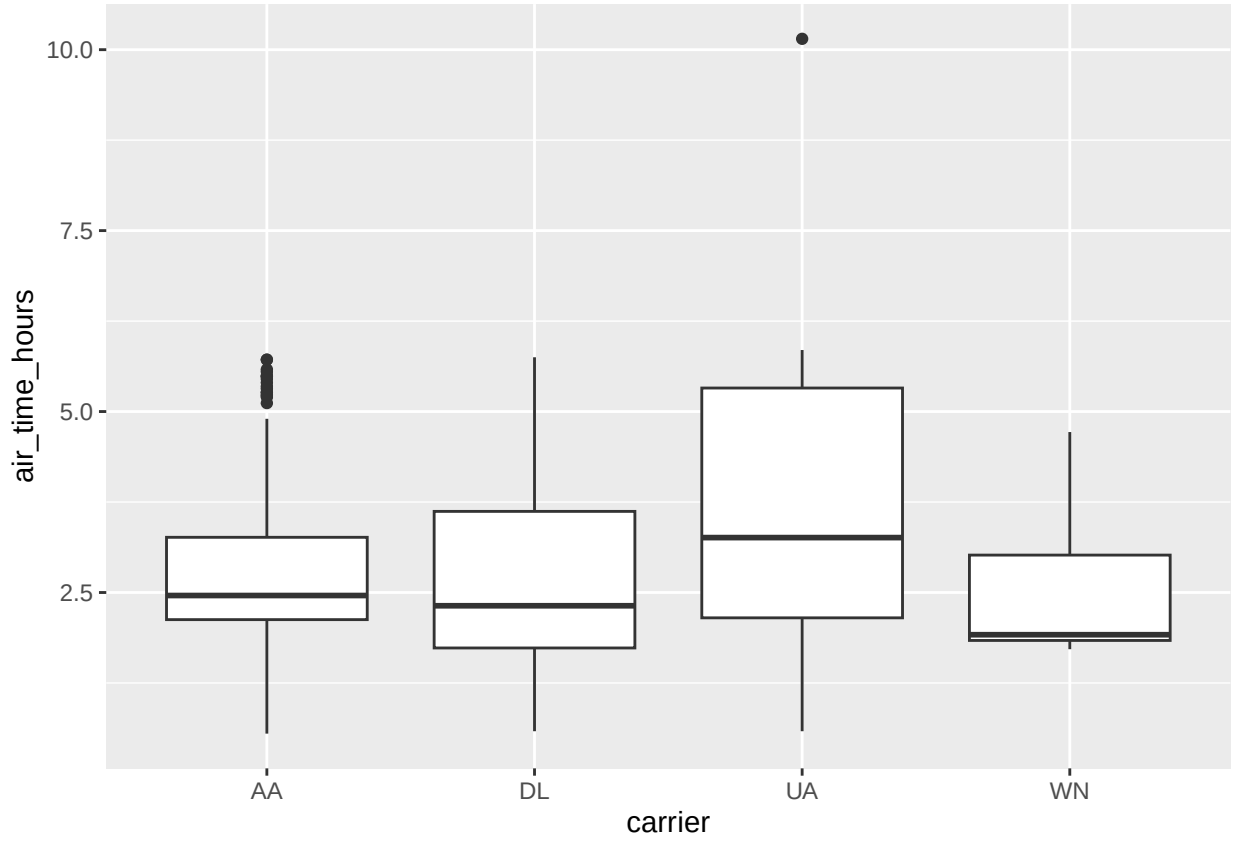
```
# Test the significance of pairwise differences and plot the results

with(carrierhours, pairwise.t.test(air_time, carrier))
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  air_time and carrier
##
##    AA      DL      UA
## DL 0.25057 -       -
## UA 0.07957 0.00044 -
## WN 0.07957 0.23488 0.00041
##
## P value adjustment method: holm
```

Statistical summary of air time over carriers

```
carrierhours %>%
  ggplot() +
  geom_boxplot(aes(carrier, air_time_hours))
```



Carrier ranking based on departure delay

```
# Rank each flight within a daily
ranked <- flights_tbl %>%
  group_by(year, month, day, carrier) %>%
  select(dep_delay) %>%
  mutate(rank = rank(desc(dep_delay)))
```

```
## Adding missing grouping variables: `year`, `month`, `day`, and `carrier`
```

```
#showing SQL query generated from dplyr command
dplyr::show_query(ranked)
```

```
## <SQL>
## SELECT
##   `year`,
##   `month`,
```

```
##   `day`,
##   `carrier`,
##   `dep_delay`,
##   RANK() OVER (PARTITION BY `year`, `month`, `day`, `carrier` ORDER BY `dep_delay` DESC) AS `rank`
## FROM `flights`
```

```
#showing carrier ranking
ranked
```

```
## # Source: spark<?> [?? x 6]
## # Groups: year, month, day, carrier
##     year month   day carrier dep_delay  rank
##    <int> <int> <int> <chr>       <dbl> <int>
##  1  2013     1     1 EV            379     1
##  2  2013     1     1 EV            290     2
##  3  2013     1     1 EV            260     3
##  4  2013     1     1 EV            216     4
##  5  2013     1     1 EV            192     5
##  6  2013     1     1 EV            155     6
##  7  2013     1     1 EV            141     7
##  8  2013     1     1 EV            121     8
##  9  2013     1     1 EV            119     9
## 10  2013     1     1 EV            115    10
## # ... with more rows
```

Disconnecting spark

```
spark_disconnect(sc)
```