Intro to sparklyr

Yitao Li

RStudio PBC

Agenda

- Quick overview of Apache Spark and sparklyr
- Installing sparklyr & connecting to Spark
- Mastering Spark with R (therinspark.com)
- ML & streaming demos
- Overview of some recent sparklyr features
- Comparison between sparklyr and SparkR

What is sparklyr?

Sparklyr is an R interface for Apache Spark

- **It is available on CRAN and is simple to install**
- **It supports many versions of Spark (from version 1.6 of the distant past to the most recent version of 3.0)**
- **It combines familiar constructs and best practices from other R packages such as broom, dplyr, DBI, magrittr, and tidyr (as of recent) with the distributed in-memory data analytics engine offered by Apache Spark**

Installing sparklyr

- **Run**

  ```
  install.packages(“sparklyr”)
  ```

  to install the latest version that is released to CRAN

- **Or alternatively, run**

  ```
  remotes::install_github("sparklyr/sparklyr", ref = "main")
  ```

  to experiment with a version that is not released yet

Connecting to Apache Spark

- Installing and running Apache Spark locally for prototyping purposes

```
library(sparklyr)
spark_version <- “3.0.0”
spark_install(version = spark_version)
sc <- spark_connect(master = “local”, version = spark_version)
```

- Connecting to an existing Spark cluster (e.g., see https://github.com/sparklyr/sparklyr/wiki/EMR)

```
library(sparklyr)
sc <- spark_connect(master = "yarn", spark_home =
"/usr/lib/spark")
```

Mastering Spark with R

Before going ahead with some more demos, I would like to first mention https://therinspark.com

In my opinion it is a great book for getting started with Apache Spark and sparklyr. It explains a number of topics fairly well with examples.

It is freely available online and contains examples of pretty much everything that Sparklyr has to offer.

I will also aim to update the book with some recent new features in Sparklyr fairly soon.

Demo: Connecting to Apache Spark

```
library(sparklyr)

sc <- spark_connect(...) # I will demonstrate connecting to Spark locally
and connecting to an EMR cluster

# I will also create a Spark dataframe with 5 partitions and then print
# its number of partitions and content to show the Spark connection is
# fully functional
sdf <- sdf_len(sc, 10, repartition = 5)
print(sdf %>% sdf_num_partitions())
print(sdf)
```

Demo: Building and running a ML pipeline in sparklyr

We will use a subset of the Shakespeare dataset at https://www.kaggle.com/kingburrito666/shakespeare-plays?select=Shakespeare_data.csv to try to train a ML model that predicts character said what line

https://github.com/yitao-li/sparklyr_demos/tree/main/ml/shakespeare

Demo with streaming dataframe

https://github.com/yitao-li/sparklyr_demos/tree/main/streaming/simple

Some major sparklyr features from recent releases

- sparklyr 1.2
  - Databricks connect
  - Support for nested list types in copy_to() and collect(), converting unnamed list columns in R into Spark SQL array columns, and named list columns in R into Spark SQL struct columns, and vice versa
  - Support for Spark as a foreach parallel backend
- sparklyr 1.3 & 1.4
  - All Spark SQL higher order functions supported through dplyr, with lambdas specified by formulae in R (e.g., `dplyr::mutate(arr = array_sort(arr, ~ as.integer(sign(.y - .x))))` to sort each array in an array column named ‘arr’ in descending order)
  - Improvements in dplyr::sample_n and dplyr::sample_frac -- both now support efficient (parallelized & one-pass) weighted sampling with and without replacements on Spark dataframes
  - Specialized implementations of tidyr verbs for Spark dataframes (including pivot_wider, pivot_longer, nest, unnest, separate, unite, and fill)

Sparklyr vs SparkR

Sparklyr:

Sparklyr is definitely more geared towards people who use R for data science and who are accustomed to interfaces provided by R packages such broom, dplyr, and tidyr

SparkR:

SparkR appears to simply contain R bindings for Spark functionalities and feels a bit less connected with the rest of the R ecosystem.

It’s probably great for people who already know Spark quite well.

Because SparkR is part of the Apache Spark release, it could provide R interface to some new Spark features before Sparkly does.

Engagement with the open source community

- Many new features and bug fixes in sparklyr are inspired by feed backs from a large community of sparklyr users
- Please don’t hesitate to post any ideas, questions, bug reports, or feature requests you have at http://github.com/sparklyr/sparklyr/issues
- We are also more than happy to review your pull requests at https://github.com/sparklyr/sparklyr/pulls

The End

EMR

This slide documents how the EMR (Elastic MapReduce) cluster that was used in the demo was created (accurate as of October 2020). This is just for your reference. You are more than welcome to try out similar steps with your favorite cloud provider (not necessarily EC2).

1. Login to your EC2 account and go to https://console.aws.amazon.com/elasticmapreduce/home, click “create cluster” on top-left
2. Under “software configuration” choose “emr-6.1.0” as the release (no reason to not go with the latest of everything), and choose “Spark: Spark 3.0.0 on Hadoop 3.2.1 YARN with and Zeppelin 0.9.0-preview1“ as applications
3. Instance type: I went with the default “m5.xlarge” for the purpose of this demo
4. EC2 key pair: proceeding with a key pair is usually the simpler and more secure option -- I went with my existing key pair on EC2

EMR (cont’d)

5.
Edit security group rules of the relevant security group(s): you should at least allow SSH traffic from your IP to the driver node -- notice you do this during or after cluster creation, the change will take effect almost instantaneously, and there is no need to bounce any of the nodes for the change to take effect. You should also do the same for the Spark Web UI port if you plan to use the Spark Web UI to see resource utilization on the EMR cluster.
6.
Copy-paste the “Master public DNS” of your cluster and login to the master node using SSH (e.g., `ssh -i ~/.ssh/id_rsa hadoop@<hostname>`)
7.
Install some version of R -- There are many possible ways to accomplish that. The steps for compiling R from source were included in the next slide. They are included solely for the purpose repeatability. Notice compiling from source is not necessarily the best way to install R.

EMR (cont’d)

8. Install R from source:

```
sudo yum install git
git clone https://github.com/yitao-li/r-cloud-scripts.git
cd r-cloud-scripts
bash ./install_r_on_emr.sh
```

9. Launch R, install sparklyr, connect to Apache Spark in YARN client mode

```
R
> install.packages(“sparklyr”)
> library(sparklyr)
> sc <- spark_connect(master = "yarn", spark_home = "/usr/lib/spark")
```