Spark In 

For vagrant clusters

Run these scripts in on your edge node and control node respectively. Each may take as much as 30-40 minutes to run, so start now.

- *install-r-studio-edge.sh:* This script installs both R and RStudio Server on the edge node. Once it has been run, you should be able to get to RStudio at the address `http://edge:8787` and to log in with user name `vagrant` and password `vagrant`. Because of the need to download all the packages and compile them, this will take almost an hour to run.
- *install-r-control.sh:* We don't need RStudio server on the control node, so this script only installs R and the packages. It won't take quite as long as the edge node, but still requires some time to run.

Agenda

- What's different about Spark when using R
- Getting set up
- Adapting to the SparkR API
- Rethinking the Spark interface
- Comparisons with Scala and Python interfaces

What's Different about R on Spark?

- Think Data Science instead of Data Engineering
  - No RDD API
  - High level operations focused on DataFrames and SQL
  - Easy access to Machine Learning libraries
- Multiple libraries with which to use Spark
  - SparkR
  - sparklyr

Getting Set Up

If you ever need to do a manual upgrade, you do this

The following assumes you have downloaded a version of Spark into your Vagrant directory. In this case, we will assume it to be spark-2.0.1-bin-hadoop2.6

1. **Set SPARK_HOME:** export SPARK_HOME=/vagrant/spark-2.0.1-bin-hadoop2.6
2. **Add Spark to your path:** export PATH=$SPARK_HOME/bin:$PATH
3. **Copy over your hive-site.xml to the Spark conf directory:**
   cp /etc/hive/conf/hive-site.xml $SPARK_HOME/conf/
4. **Set your HADOOP_CONF_DIR variable so that you can use yarn mode:**
   export HADOOP_CONF_DIR=/etc/hadoop/conf
5. (**optional) For Spark versions before 2.0, reduce your Spark logging from INFO to WARN.**
   sudo sed -i.bak 's/INFO/WARN/' /usr/lib/spark/conf/log4j.properties
6. **(optional) If you are running 2.0 or later, the logging default is WARN and you can adjust this at the command line:**
   sc.setLogLevel(newLevel)

Spark R smoke test

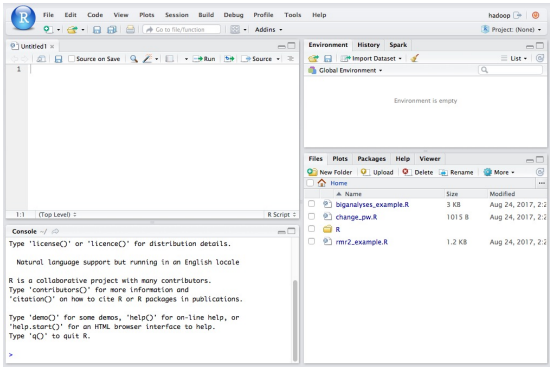
- On the edge node, type: `sparkR --master yarn`
- You should see the following:

```
[vagrant@edge vagrant]$ sparkR --master yarn
R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Launching java with spark-submit command /vagrant/spark-2.0.1-bin-hadoop2.6/bin/spark-submit   "sparkr-shell" /tmp/RtmpPBYVP6/
backend_port52b86fd11cb1
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).

 Welcome to
    ____              __
   / __/__  ___ _____/ /__
  _\ \/ _ \/ _ `/ __/  '_/
 /___/ .__/\_,_/_/ /_/\_\   version  2.2.0
    /_/


 SparkSession available as 'spark'.
>
```

RStudio Server: Point Your Browser To Port 8787

- The RStudio Server Integrated Development Environment is available as a Web page at $MY_IP_ADDRESS:8787
- Log in using user name hadoop, password hadoop
- You should see what's shown on the next page.

RStudio Server Screen Shot



RStudio Server Environment Variables

RStudio Server doesn't necessarily have all the necessary Spark environment variables, so you'll have to start your programs with the following preamble

```
if (nchar(Sys.getenv("SPARK_HOME")) < 1) {
  Sys.setenv(SPARK_HOME = "/usr/lib/spark")           # or wherever your Spark
install lives
}
if (nchar(Sys.getenv("HADOOP_CONF_DIR")) < 1) {
  Sys.setenv(HADOOP_CONF_DIR = "/etc/hadoop/conf")    # or wherever your Hadoop
lives
}
if (nchar(Sys.getenv("YARN_CONF_DIR")) < 1) {
  Sys.setenv(YARN_CONF_DIR = "/etc/hadoop/conf")      # or wherever your YARN config
lives
}
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sparkR.session(master = "yarn", sparkConfig = list(spark.driver.memory = "3g"))
```

sparkR.session is what creates your Spark context for you. :

To Run under RStudio Server, Go To Port 8787

To run sparkR under RStudio Server, simply run these commands after pointing your browser to port 8787:

```
if (nchar(Sys.getenv("SPARK_HOME")) < 1) {
  Sys.setenv(SPARK_HOME = "/usr/lib/spark")            # or wherever your Spark
install lives
}
if (nchar(Sys.getenv("HADOOP_CONF_DIR")) < 1) {
  Sys.setenv(HADOOP_CONF_DIR = "/etc/hadoop/conf")     # or wherever your Hadoop
lives
}
if (nchar(Sys.getenv("YARN_CONF_DIR")) < 1) {
  Sys.setenv(YARN_CONF_DIR = "/etc/hadoop/conf")       # or wherever your YARN confi
lives
}
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sparkR.session(master = "yarn", sparkConfig = list(spark.driver.memory = "3g"))
```

sparkR.session is what creates your Spark context for you. :

Comparing DataFrames in Scala, Python, and R


```
val df = spark.read.json("examples/src/main/resources/people.json")

// Displays the content of the DataFrame to stdout
df.show()
```

This is the Python version

```
df = spark.read.json("examples/src/main/resources/people.json")

# Displays the content of the DataFrame to stdout
df.show()
```

And here's R

```
df <- read.df("examples/src/main/resources/people.json", "json")

# Displays the content of the DataFrame to stdout
showDF(df)
```

Comparing SQL queries in Scala, Python, and R


```scala
val df = spark.sql("SELECT * FROM STOCKS")

// Displays the content of the DataFrame to stdout
df.show()
```

This is the Python version

```python
df = spark.sql("SELECT * FROM STOCKS")

# Displays the content of the DataFrame to stdout
df.show()
```

And here's R

```r
df <- sql("SELECT * FROM STOCKS")

# Displays the content of the DataFrame to stdout
showDF(df)
```

What happened to RDDs?

- Spark's R API doesn't support RDDs because they are too low level
- R focuses its API on Datasets, DataFrames, and other higher-level objects

You Have Two Ways To Use Spark From R

- SparkR: Library that uses R interfaces to the same Spark APIs used by Scala
- sparklyr: Library that translates dplyr operations to SQL statements operated by the cluster

We'll look at the first, which is more common. We'll then walk through the second with in code.

Using the SparkR Library and API

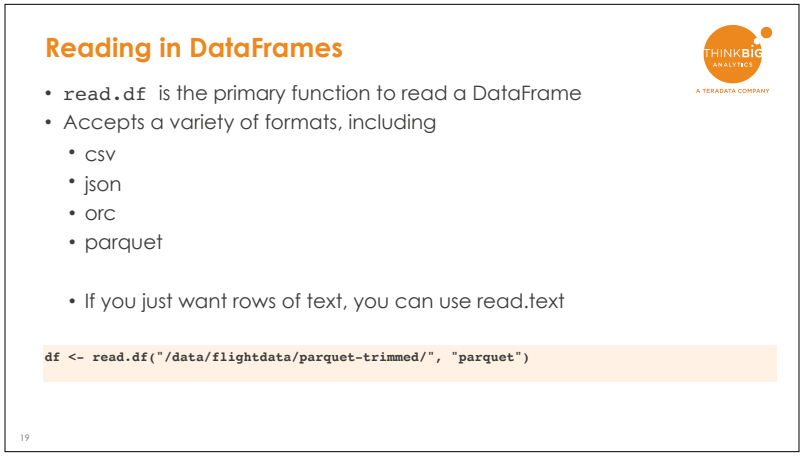
The SparkR API

- Provides most of the same grouping and aggregation functions most R programmers are familiar with in packages such as `plyr` and `dplyr`.
- Some of the functions are spelled differently, so pay close attention to capitalization, underscores, and dots in the examples.

You Must Distinguish Two Types of Dataframes

- R natively supports a ***data.frame*** type, which is implemented as a list of columns.
- SparkR supports a different dataframe type called a ***DataFrame***, which is implemented within the Spark Cluster as an RDD of Rows.
- You must use different functions for the two different types!

```
head(faithful)
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55

str(faithful)
## 'data.frame': 272 obs. of  2 variables:
##  $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
##  $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...

df <- as.DataFrame(faithful) # note this is
as.Dataframe, not as.data.frame
# Displays the first part of the SparkDataFrame
head(df)
## you see the same numbers but it takes longer.
##  eruptions waiting
##1     3.600      79
##2     1.800      54
##3     3.333      74
```



Do not use the dplyr package, since it conflicts with sparklyr

Actual sparkR code

This is all a lot like SparkSQL in Scala, but with different syntax

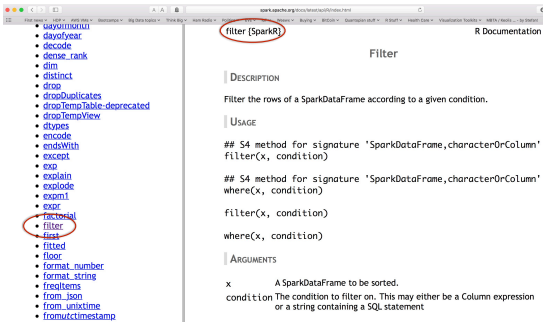
```
> df <- read.df("/data/spark-resources-data/people.json", "json")
17/07/20 10:26:36 WARN datasources.DataSource: Error while looking for metadata directory.
> showDF(df)
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+

> printSchema(df)
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
> names(df)
[1] "age"  "name"

> df %>% select("name") %>% showDF()
+-------+
|   name|
+-------+
|Michael|
|   Andy|
| Justin|
+-------+
```

Library magrittr gives you a pipeline operator similar to how we would use "." in Python or Scala

```
library(magrittr)
> df %>% select(df$name, df$age + 1) %>% showDF()
+-------+-----------+
|   name|(age + 1.0)|
+-------+-----------+
|Michael|       null|
|   Andy|       31.0|
| Justin|       20.0|
+-------+-----------+


> df %>% filter(df$age > 21) %>% showDF()
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+


> df %>% groupBy("age") %>% count() %>% showDF()
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  30|    1|
+----+-----+
```

WARNING: Don't be fooled into thinking that these are dplyr operations

These are SparkR Functions



22

SparkR functions are all on the website for Apache

Lab: 02-SparkR-API.{md,R}

Using Sparklyr

Sparklyr Hides Spark Behind dplyr Primitives

- Developed by the RStudio team
- Provides a complete *dplyr* backend that connects to Spark
- Filters and aggregates Spark DataSets and brings them into R for analysis and visualization.
- Allows use Spark's of distributed machine learning library from R.
- Allows extensions that call the full Spark API and provide interfaces to Spark packages.
- Still relatively early in its maturity

- Less than 1 year old

Example sparklyr code

- Sparklyr works as a back end to `dplyr`.
- Acts as if it is part of the Hadley Wickham *tidyverse* packages for Data Science
- Once your Spark connection is set up, it mostly just looks like *dplyr* pipelines
- However, due to lazy evaluation, you must *collect* your results before printing or graphing them.

```
library(sparklyr)

sc <- spark_connect(master = "yarn", version = "2.2.0")

install.packages(c("nycflights13")
library(dplyr)

# Copy nycflights to cluster

flights_tbl <- copy_to(sc, nycflights13::flights,
"flights")

flights_tbl %>% filter(dep_delay == 2) %>% head()

delay <- flights_tbl %>%
  group_by(tailnum) %>%
  summarise(count = n(), dist = mean(distance), delay =
mean(arr_delay)) %>%
  filter(count > 20, dist < 2000, !is.na(delay)) %>%
  collect
```

Sparklyr Supports Machine Learning Too

- Most of the machine learning libraries work

Lab: 03-Sparklyr.{md,R}

Summary

- Python and R environments for Spark exist and work
- Many of the same principles you've learned in Scala work similarly in these languages
- The dataFrame API is more likely to be compatible across languages than the RDD one
- All non-Scala languages have some "gotchas" in either functionality or performance

x1 machine from Amazon will be $6 an hour