Parallelizing Existing R Packages with SparkR

Hossein Falaki
@mhfalaki
About me

• Former Data Scientist at Apple Siri
• Software Engineer at Databricks
• Started using Apache Spark since version 0.6
• Developed first version of Apache Spark CSV data source
• Worked on SparkR & Databricks R Notebook feature
• Currently focusing on R experience at Databricks
What is SparkR?

An R package distributed with Apache Spark:
- Provides R frontend to Spark
- Exposes Spark DataFrames (inspired by R and Pandas)
- Convenient interoperability between R and Spark DataFrames

distributed/robust processing, data sources, off-memory data structures
+ Dynamic environment, interactivity, packages, visualization
SparkR architecture

Spark Driver

Worker

JVM

Worker

JVM

Data Sources

cassandra

HDFS

HBASE

databricks
SparkR architecture (since 2.0)
Overview of SparkR API

http://spark.apache.org/docs/latest/api/R/

**IO**
read.df / write.df /
createDataFrame / collect

**Caching**
cache / persist / unpersist /
cacheTable / uncachetable

**SQL**
sql / table / saveAsTable /
registerTempTable / tables

**ML Lib**
glm / kmeans / Naïve Bayes
Survival regression

**DataFrame API**
select / subset / groupBy /
head / avg / column / dim

**UDF functionality (since 2.0)**
spark.lapply / dapply /
gapply / dapplyCollect
SparkR UDF API

**spark.lapply**
Runs a function over a list of elements

**dapply**
Applies a function to each partition of a SparkDataFrame

**gapply**
Applies a function to each group within a SparkDataFrame

spark.lapply()
dapply()
gapply()
dapplyCollect()
gaplyCollect()
Simplest SparkR UDF pattern
For each element of a list:

1. Sends the function to an R worker
2. Executes the function
3. Returns the result of all workers as a list to R driver

spark.lapply(1:100, function(x) {
  runBootstrap(x)
})
spark.lapply control flow

1. Serialize R closure
2. Transfer over local socket
3. Transfer serialized closure over the network
4. Transfer over local socket
5. De-serialize closure
6. Serialize result
7. Transfer over local socket
8. Transfer serialized closure over the network
9. Transfer over local socket
10. Deserialize result
For each partition of a Spark DataFrame

1. collects each partition as an R data.frame
2. sends the R function to the R worker
3. executes the function

\[
dapply(\text{sparkDF}, \text{func}, \text{schema})
\]
combines results as DataFrame with provided schema

\[
dapplyCollect(\text{sparkDF}, \text{func})
\]
combines results as R data.frame
dapply control & data flow

local socket

cluster network

local socket

input data ser/de transfer

result data ser/de transfer
dapplyCollect control & data flow

Driver JVM

Worker JVM

local socket

cluster network

local socket

input data ser/de transfer

result transfer

result deser

R
gapply

Groups a Spark DataFrame on one or more columns
1. collects each group as an R data.frame
2. sends the R function to the R worker
3. executes the function

\[ \text{gapply}(\text{sparkDF}, \text{cols}, \text{func}, \text{schema}) \]

combines results as DataFrame with provided schema

\[ \text{gapplyCollect}(\text{sparkDF}, \text{cols}, \text{func}) \]

combines results as R data.frame
gapply control & data flow

Driver JVM

Worker JVM

local socket

cluster network

local socket

data shuffle

input data ser/de transfer

result data ser/de transfer

R

R

local socket

cluster network

local socket

R

input data ser/de transfer

result data ser/de transfer

R

local socket
# dapply vs. gapply

<table>
<thead>
<tr>
<th>gapply</th>
<th>dapply</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature</td>
<td></td>
</tr>
<tr>
<td>gapply(df, cols, func, schema)</td>
<td></td>
</tr>
<tr>
<td>gapply(gdf, func, schema)</td>
<td></td>
</tr>
<tr>
<td>user function</td>
<td></td>
</tr>
<tr>
<td>signature</td>
<td></td>
</tr>
<tr>
<td>function(key, data)</td>
<td></td>
</tr>
<tr>
<td>function(data)</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
</tr>
<tr>
<td>controlled by grouping</td>
<td></td>
</tr>
<tr>
<td>not controlled</td>
<td></td>
</tr>
</tbody>
</table>

Parallelizing data

- Do not use `spark.lapply()` to distribute large data sets
- Do not pack data in the closure
- Watch for skew in data
  - Are partitions evenly sized?
- Auxiliary data
  - Can be joined with input DataFrame
  - Can be distributed to all the workers
Packages on workers

• SparkR closure capture does not include packages
• You need to import packages on each worker inside your function
• If not installed install packages on workers out-of-band
• spark.lapply() can be used to install packages
Debugging user code

1. Verify your code on the Driver
2. Interactively execute the code on the cluster
   - When R worker fails, Spark Driver throws exception with the R error text
3. Inspect details of failure reason of failed job in spark UI
4. Inspect stdout/stderror of workers
Demo

http://bit.ly/2krYMWc
http://bit.ly/2ltLVKs
Thank you!