ggplot2.SparkR: Rebooting ggplot2 for Scalable Big Data Visualization

Jonghyun Bae†, Sangoh Jeong*, Wenjing Jin† and Jae W. Lee†

†Sungkyunkwan University
*SK Telecom
Speakers

• Sangoh Jeong (sangoh.jeong@sk.com)
  – Senior Manager at SK telecom in Korea
  – Interested in R, Spark

• Jonghyun Bae (jonghbae@skku.edu)
  – Graduate student at Sung Kyun Kwan University
  – Interested in R, JavaScript and Spark
Big Data Analytics Pipeline

Log data
Logstore
Spark
Hadoop
Store
Extract & Transform
API
Query
SQL
Analyze
Visualization
Interactive dashboard & app
Spreadsheets
This Work!

Structured data
Datastore

Unstructured data
Cloud storage

Source: http://www.slideshare.net/BigDataCloud/big-data-analytics-with-google-cloud-platform
Why Big Data Visualization?

● Case of a business unit at SK Telecom
  – Typical DB size: 70M records with 330 columns
  – Analyzes the DB using R on a single-node scale-up server
  – Has much bigger DBs that cannot be handled by this server

● The business unit’s visualization needs
  – Use of R
  – Easy-to-use APIs
  – Scalable solution for the bigger DBs
R Has Great Visualization Packages

But, these packages cannot process Spark DataFrames.
ggplot2

• (Arguably) the most popular visualization package for R
  – Based on the “layered” grammar of graphics
  – Making it easy to produce high-quality graphs
  – Limited to single node processing

“Base graphics are good for drawing pictures; ggplot2 graphics are good for understanding the data”

(Hadley Wickham, Creator of ggplot2, 2012)
Top 10 packages in R

Top package for visualization

ggplot2: Example Plots

Source: ggplot2 documentation, http://docs.ggplot2.org/current
ggplot2.SparkR = SparkR+ggplot2!

• An R package extending ggplot2 to visualize big data represented in Spark DataFrame
Example: Draw a histogram using DataFrame
ggplot2.SparkR Simplifies Plotting (2)

BEFORE

# Pre-processing Spark DataFrame using SparkR API
range <- select(df, min(df$Price), max(df$Price))
breaks <- fullseq(range, diff(range / binwidth))
left <- breaks[-length(breaks)]; right <- breaks[-1]
breaks_df <- createDataFrame(sqlContext, data.frame(left = left,
right = right))
histogram_df <- join(df, breaks_df, df$Price >= breaks_df$left &
df$Price < breaks_df$right, "inner")
histogram_df <- count(groupBy(histogram_df, "left", "right"))

# Draw histogram chart using ggplot2 API
ggplot(collect(histogram_df), aes(xmin = left, xmax = right, ymin = 0, ymax = count)) + geom_rect()

AFTER

# It just takes one line!
ggplot(df, aes(x = Price)) + geom_histogram()


**ggplot2.SparkR: Features**

- **Scalable**
  - Beyond the capacity of single node (cf. ggplot2)
  - Performance scales to the number of nodes

- **Easy to use**
  - No changes to ggplot2 API
  - No training required for existing ggplot2 users

- **Readily deployable**
  - No modifications required for Spark
  - Using SparkR API only
The Rest of This Talk

Overview
How to Use It?
Architecture
Performance
Status & Plan
Summary
How to Use It?
Using ggplot2.SparkR is as easy as 1-2-3!

1. Install
2. Create
3. Draw
devtools::install_github ("SKKU-SKT/ggplot2.SparkR")

1. Install from Github
df <- read.json(sqlContext, "hdfs://localhost:9000/dataset")

2. Create DataFrame
3. Draw it (using ggplot2 API)!

```
ggplot(df, aes(x = Item, fill = Payment)) + geom_bar() + coord_flip()
```

Note that `df` is a Spark DataFrame object (not R data.frame).
Demo

1. Install
2. Create
3. Draw
## Demo: Data Set

- **Schema:** Sales record from a department store chain
  - Source: [http://content.udacity-data.com/course/hadoop/forum_data.tar.gz](http://content.udacity-data.com/course/hadoop/forum_data.tar.gz)

<table>
<thead>
<tr>
<th>Date</th>
<th>Item</th>
<th>Payment</th>
<th>Place</th>
<th>Price</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-01-11</td>
<td>Baby</td>
<td>Discover</td>
<td>Houston</td>
<td>426.32</td>
<td>09:05</td>
</tr>
<tr>
<td>2012-07-23</td>
<td>Books</td>
<td>MasterCard</td>
<td>Lexington</td>
<td>47.20</td>
<td>17:30</td>
</tr>
<tr>
<td>2012-04-21</td>
<td>Consumer Electronics</td>
<td>MasterCard</td>
<td>St. Louis</td>
<td>234.95</td>
<td>15:01</td>
</tr>
<tr>
<td>2012-10-26</td>
<td>Garden</td>
<td>Cash</td>
<td>Spokane</td>
<td>469.47</td>
<td>12:46</td>
</tr>
</tbody>
</table>
### Supported Graph Types & Options

<table>
<thead>
<tr>
<th>Graph types</th>
<th>Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>geom_bar</td>
<td>Bars, rectangles with bases on x-axis.</td>
</tr>
<tr>
<td></td>
<td>geom_histogram</td>
<td>Histogram.</td>
</tr>
<tr>
<td></td>
<td>stat_sum</td>
<td>Sum unique values.</td>
</tr>
<tr>
<td></td>
<td>geom_boxplot</td>
<td>Box and whiskers plot.</td>
</tr>
<tr>
<td></td>
<td>geom_bin2d</td>
<td>Heatmap of 2d bin counts.</td>
</tr>
<tr>
<td></td>
<td>geom_freqpoly</td>
<td>Frequency polygon.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Positions</th>
<th>Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>geom_bar</td>
<td>Stack overlapping objects on top of one another</td>
</tr>
<tr>
<td></td>
<td>geom_histogram</td>
<td>Same as above, but the range is standardized.</td>
</tr>
<tr>
<td></td>
<td>geom_boxplot</td>
<td>Adjust position by dodging overlaps to the side</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Facets</th>
<th>Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>facet_null</td>
<td>Facet specification: a single panel</td>
</tr>
<tr>
<td></td>
<td>facet_grid</td>
<td>Lay out panels in a grid</td>
</tr>
<tr>
<td></td>
<td>facet_wrap</td>
<td>Wrap a 1d ribbon of panels into 2d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scales</th>
<th>Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>scale_x_log10</td>
<td>Put x-axis on a log scale</td>
</tr>
<tr>
<td></td>
<td>scale_y_log10</td>
<td>Put y-axis on a log scale</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coords</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coord_cartesian</td>
<td>Cartesian coordinates</td>
</tr>
<tr>
<td></td>
<td>coord_flip</td>
<td>Flip cartesian coordinates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ranges</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xlim</td>
<td>Set the ranges of the x axis</td>
</tr>
<tr>
<td></td>
<td>ylim</td>
<td>Set the ranges of the y axis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Texts</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xlab</td>
<td>Change the label of x-axis</td>
</tr>
<tr>
<td></td>
<td>ylab</td>
<td>Change the label of y-axis</td>
</tr>
<tr>
<td></td>
<td>ggtitle</td>
<td>Change the graph title</td>
</tr>
</tbody>
</table>
Supported Graph Types

- **stat_sum**
- **geom_boxplot**
- **geom_bin2d**
- **geom_freqpoly**
Supported Graph Options

- **position_stack**
- **position_dodge**
- **position_fill**
- **facet_grid**
- **facet_wrap**
Architecture
ggplot2.SparkR: Architecture (1)

Three-stage pipeline:

- **Parameter extraction**
  - x, y, colour, facet, scale, geom, etc.

- **Data processing**
  - Get data from the original source
  - Process data using graph parameters

- **Plotting**
  - Draw graphs on display windows
ggplot2.SparkR: Architecture (2)

Input Checking

Parameter extraction

Data processing (R data.frame)

Plotting

Parameter extraction

Data processing (Spark DataFrame)

R data.frame
ggplot2.SparkR: Architecture (3)

INPUT CHECKING

Parameter extraction

Data processing (R data.frame)

Plotting

Spark DataFrame

Extract x, y, colour, facet, scale, geom, etc

Parameters

Graph type

Layer option
ggplot2.SparkR: Architecture (4)

- Use 12 internal stages to process Spark DataFrame and finally convert it to R data.frame
- Calculated R data.frame

Parameter extraction

Data processing (R data.frame)

Plotting

Input Checking

Parameter extraction

Data processing (Spark DataFrame)

Spark DataFrame

Parameters

Graph type

Layer option

ggplot2

ggplot2.SparkR
ggplot2.SparkR: Data Flow (1)

ggplot2.SparkR: Data Flow (2)

Performance
Experimental Setup (1)

- Cluster setup:
  8-node Spark Cluster

<table>
<thead>
<tr>
<th>Node Parameters</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel® i7-4790 (Haswell) 4GHz 8 cores</td>
</tr>
<tr>
<td>Memory</td>
<td>32GB DDR3 1600MHz</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu 14.04 LTS</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Ver. 1.2.1 (stable)</td>
</tr>
<tr>
<td>Spark</td>
<td>Ver. 1.5.0</td>
</tr>
<tr>
<td>R</td>
<td>Ver. 3.2.2</td>
</tr>
<tr>
<td>JDK</td>
<td>Ver. 1.8.0_60</td>
</tr>
<tr>
<td>Spark Worker</td>
<td>8 cores + 30GB / Worker</td>
</tr>
<tr>
<td>Network</td>
<td>Gigabit Ethernet</td>
</tr>
</tbody>
</table>
Experimental Setup (2)

- Workload: Bar graph (ggplot(df, aes(x=Item))+geom_bar())
Performance: Scalability

Performance scales to the number of cluster nodes.

Input size: 460M rows x 6 columns

Based on 12 runs

- Maximum
- Average
- Minimum
Throughput (inverse of slope) remains relatively stable.

Based on 12 runs

Largest data for R single node
Status & Plan
ggplot2.SparkR Project Page

To Report Bugs or Request Features

● Report using our github issue page
  https://github.com/SKKU-SKT/ggplot2.SparkR/issues

Or

● Email to the ggplot2.SparkR mailing list
  ggplot2-sparkr@googlegroups.com
API Coverage & Future Plan

• ggplot2 API Coverage
  – Total: 135
  – Primary target: 45* (100%)
  – Implemented: 21 (47%)

• Future Plan
  – Register the project to spark-packages.org (and CRAN)
  – Improve API coverage
  – Optimize performance

*Suitable for big data visualization
Summary: ggplot2.SparkR

• R package extending ggplot2 to take Spark DataFrame (as well as R data.frame) as input

• Scalable, easy to use, and readily deployable

• Feedback and contributions from Spark Community will be greatly appreciated.
THANK YOU.