Rc²: an Environment for Running R and Spark in Docker Containers

UseR! 2017

E. James Harner¹ and Mark Lilback²

¹Department of Statistics
West Virginia University
²Rc2ai.io

July, 2017
Outline

Introduction

Rspark

Rc^2 Server

Rc^2 Client

Summary
Big Data Architectures

What data architectures are needed for big data analytics?

Two principal architectures:

**HDFS/Hadoop**  A software framework for distributed storage (HDFS) and distributed processing (MapReduce).

**Spark**  A cluster computing environment using in-memory primitives rather than Hadoop’s two-stage, disk-based MapReduce approach.

These environments are generally part of a much larger ecosystem. A frontend language is used to access their services, typically R or Python, rather than their native languages: Java for Hadoop and Scala for Spark.

The following two slides from O’Reilly show typical production environments for batch and streaming data, respectively.
Batch Architecture
Streaming Architecture
Implementation

How do we build these systems?

How do we integrate all the components with the correct versions for these systems?

How do we update these systems?

What about the data sources? (ETL)

How do we preprocess unstructured data?

Many believe containerization is the future:

- Docker containers: containerize the components of your system
- Kubernetes: orchestrate containers for production at scale
Rspark: a Simple Approach

The GitHub repo for rspark:
https://github.com/jharner/rspark

Rspark constructs containers (currently) for Postgres, HDFS, Hive, and RStudio. Spark and H2O are built into the RStudio container. Other containers can be added as needed, e.g., for HBase.

Rspark runs on a single machine which can be a server, but it supports multiple cores. RStudio is the web-based client using port 8787.

Principal deployment issue: building or launching containers is done simultaneously and thus dependency relationships may be violated, e.g., launching hive before hdfs completes.

Rc² attempts to solve dependency relationships programmatically.
Rc² Overall Architecture

The GitHub repo for rc2server:
https://github.com/rc2server

Rc² has an underlying 4-tier architecture:

- **client** native application (Swift) for macOS and iOS
  (web-based version planned)
- **app server** Perfect/Swift app-/web-server using technologies
  such as WebSockets and RestKit
- **compute engine** JSON over BSD sockets for R
- **databases** PostgreSQL for primary data storage, including
  meta-data, user profiles, files, .Rdata (as blobs), etc.
  Apache CouchDB (NoSQL—key-value) for logging
  client/server JSON messages

The three backend tiers run in Linux containers.
Server Architecture Diagram

Client -> WebSession 2 -> Swift/Perfect -> Postgres

RSession 1 -> RSession Cluster -> rcompute -> Kubernetes

RSession N -> Spark Cluster -> HDFS/HBase/S3 -> Kubernetes

WebSession 1 -> WebSession N -> CouchDB
Server Architecture Components

The repo for both the server and client will soon be moved to Rc2ai.

**Client:** end-user application communicating via REST and WebSockets

**Perfect:** App/Web server running a restful application and WebSessions

**WebSession:** an in-memory object that connects multiple clients with a single RSesssion

**RCompute:** application written in C++ that forks RSessions

**RSession** application that contains an R execution environment via RInside. It manages interchange between R and a WebSession
Server Databases

Postgres: stores all persistent data, including file content (excluding hdfs)

CouchDB: stores logs of various kinds, including session playback capability

HDFS/Hadoop: allows access from WebSession/Client and RSession

Spark allows access from WebSession/Client and RSession

HDFS/Hadoop is accessed from RSession using RHadoop and RHIPE (for divide and recombine statistical algorithms).

Spark is accessed from RSession using SparkR, sparklyr (for dplyr interfaces and workflows), and rsparkling (for H2O).
File Change Monitoring

- RSession fetches files from the database on init
- RSession monitors files via inotify and sends those changes to the database
- The database has triggers to send appropriate file-changed notifications
- WebSession notes those changes and sends those changes to the client
- RSession notices changes made elsewhere and updates files on the file system
The client application includes:

- a launch window for predefined or new containerized sessions;
- sharable and tabbed workspaces;
- a text editor for .R, .Rmd, .Rnw, and .txt files;
- a command line for R;
- styled text for console output, image display, and WebKit for other file types, e.g., html and pdf;
- file, workspace, and package displays;
- a graphics display supporting multiple plots;
- a control window for Docker containers.

WebSockets are used for client/server communications with minimal overhead.
Files and Workspaces

A workspace is a superset of an R workspace. It has a list of associated files (no directories currently) along with all objects stored in an .Rdata file.

Workspaces can be shared with other users for collaboration.

A workspace contains source code, shared project files, and other files. The .Rdata file, usually associated with a workspace, is hidden, but visible through a separate tab in which: the R objects in .Rdata are displayed in a variable list; a ‘data.frame‘ is displayed as a spreadsheet.

Source files are created in the text editor or imported from the local filesystem or Dropbox.
Client Interface (Session Selection View)
Client Interface (Output View)

### Regression demo

Suppose we are interested in studying a possible linear relationship between age and blood pressure in adult males. First, we obtain a random sample of men and measure the age and blood pressure on each. Then we fit a straight line to the resulting data.

The BP data is loaded into R with the `data` function:

```r
BP <- data.frame(age = c(28, 23, 52, 42, 27, 29, 43, 34, 40, 28),
                  systolic = c(78, 68, 90, 75, 68, 80, 78, 70, 80, 72))
```

```r
# Warning in data(BP): data set 'BP' not found
attach(BP)
```

The R code for plotting the data and the fitted line:

```r
BP.lm <- lm(systolic ~ age)
plot(age, systolic, xlab="Age", ylab="Systolic blood pressure")
abline(BP.lm)
```

The `print` method for the `BP.lm` object gives the regression coefficients:

```r
BP.lm
```

Thus the fitted regression line is:

$$\hat{y} = 53.69 + 0.62x.$$  

The residuals are the vertical distances between the observed and predicted (fitted) values, i.e., the $y - \hat{y}$'s. The systolic blood pressure residuals are represented by vertical lines in the figure below:

```r
systolicFit <- fitted(BP.lm)
plot(age, systolic, xlab="Age", ylab="Systolic Blood Pressure (BP)")
abline(BP.lm)
```
Client Interface (File View)

---

```r
# regression demo
output: html_document
---

```(r

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```(r

Suppose we are interested in studying a possible linear relationship between age and blood pressure in adult males. First, we obtain a random sample of men and measure the age and blood pressure on each. Then we fit a straight line to the resulting data.

The BP data is loaded into R with the `data` function:

```{r}
BP <- data.frame(age = c(28, 23, 52, 42, 27, 29, 43, 34, 40, 28),
                 systolic = c(120, 68, 90, 75, 68, 80, 70, 80, 90, 72))
data(BP)
attach(BP)
```

The R code for plotting the data and the fitted line:

```{r}
BP.lm <- lm(systolic ~ age)  
plot(BP.lm, xlab="Age", ylab="Systolic blood pressure")  
abline(BP.lm)
```

The `print` method for the `BP.lm` object gives the regression coefficients:

```{r}
BP.lm
```

Thus the fitted regression line is:

$$
\hat{y} = 53.69 + 0.62x.
$$

The residuals are the vertical distances between the observed and predicted (fitted) values, i.e., the $y - \hat{y}$ residuals. The systolic blood pressure residuals are represented by vertical lines in the figure below:

```{r}
```
Client Interface (Code Search View)

```
linear relationship between age and blood pressure in adult males. First, we obtain a random sample of men and measure the age and blood pressure on each. Then we fit a straight line to the resulting data.

\[
\begin{align*}
\text{BP data is loaded into R with the ‘data’ function:} \\
\text{BP } & \leftarrow \text{data.frame (age = c(28, 32, 52, 42, 27, 29, 43, 34, 40, 28),} \\
& \text{ systolic } = \text{ c(70, 68, 90, 75, 68, 80, 78, 70, 80, 72))} \\
& \text{ data BP} \\
& \text{ attach (BP)}
\end{align*}
\]

The R code for plotting the data and the fitted line:

\[
\begin{align*}
& \text{BP.lm } \leftarrow \text{ lm (systolic } \text{ ~ age, plot (age, systolic, xlab} = \text{ “Age”, ylab} = \text{ “Systolic blood pressure”)} \\
& \text{ abline (BP.lm)}
\end{align*}
\]

The ‘print’ method for the ‘BP.lm’ object gives the regression coefficients:

\[
\begin{align*}
& BP.lm \\
& \text{Thus the fitted regression line is:} \\
& \hat{y} = 53.69 + 0.62x
\end{align*}
\]

The residuals are the vertical distances between the observed and predicted (fitted) values, i.e., the $\hat{y}$ - $\hat{y}$. The systolic blood pressure residuals are represented by vertical lines in the figure below:

\[
\begin{align*}
\text{Error (residuals) for systolic blood pressure residuals} \\
& \text{plot (age, systolic, xlab} = \text{ “Age”, ylab} = \text{ “Systolic Blood Pressure (BP)”)
\end{align*}
\]

A density histogram of the residuals with a
```
Client Interface (Workspace View)

Suppose we are interested in studying a possible linear relationship between age and blood pressure in adult males. First, we obtain a random sample of men and measure the age and blood pressure on each. Then we fit a straight line to the resulting data.

The BP data is loaded into R with the `data` function:

```r
BP <- data.frame(age = c(28, 33, 52, 62, 27, 29, 43, 34, 40, 20),
                  systolic = c(70, 68, 90, 75, 68, 80, 70, 80, 70, 72))
```

The R code for plotting the data and the fitted line:

```r
BP.lm <- lm(systolic ~ age)
plot(age, systolic, xlab="Age", ylab="Systolic blood pressure")
abline(BP.lm)
```

The `print` method for the `BP.lm` object gives the regression coefficients:

```r
BP.lm
```

Thus the fitted regression line is:

\[ y = 53.69 + 0.62x. \]

The residuals are the vertical distances between the observed and predicted (fitted) values, i.e., the \( y - y_\text{hat} \). The systolic blood pressure residuals are represented by vertical lines in the figure below:
Client Interface (Package Help View)
Client Interface (Console View)

```r
# R code for plotting data and fitted line:
systolic = c(70, 68, 90, 75, 68, 80, 70, 80, 70, 80, 72)
data(BP)
attach(BP)
BP$lm <- lm(systolic ~ age)
plot(age, systolic, xlab="Age", ylab="Systolic blood pressure")
abline(BP$lm)
The 'print' method for the 'BP.lm' object gives the regression coefficients:

BP$lm

Thus the fitted regression line is:

\[ \hat{y} = 53.69 + 0.62x \]

The residuals are the vertical distances between the observed and predicted (fitted) values, i.e., the \( y - \hat{y} \). The systolic blood pressure residuals are represented by vertical lines in the figure below:

systolicFit <- fitted(BP$lm)
plot(age, systolic, xlab="Age", ylab="Systolic Blood Pressure (BP)"
  abline(BP$lm)
segments(age, systolic, age, systolicFit)

A density histogram of the residuals with a superimposed kernel density estimate is now shown.

hist(systolicRes, prob=T, breaks=c(-9, -4.5, 0, 4.5, 9),
     main="",
     xlim=c(-10, 10), ylim=c(0, 0.1), xlab = "Systolic BP Residuals")
lines(density(systolicRes, bw="SJ"))
rug(systolicRes)
```

---

**Client Interface (Console View)**
Client Interface (Container Control Window)
Client Interface (Container Log Window)
Images are written consecutively to files; the app server moves these files to the database as blobs, and sends the client a list of image URLs.

The client displays icons for each plot and any one, two, or four can be simultaneously displayed.
Client Interface (Plot View)

The R code for plotting the data and the fitted line:

```r
systolic = c(70, 68, 90, 75, 68, 80, 70, 70, 80, 72)
data(BP)
attach(BP)

BP.lm <- lm(systolic ~ age)
plot(age, systolic, xlab="Age", ylab="Systolic blood pressure")
abline(BP.lm)
```

The `print` method for the `BP.lm` object gives the regression coefficients:

```r
BP.lm
```

Thus the fitted regression line is:

$$\hat{y} = 53.69 + 0.62x.$$  

The residuals are the vertical distances between the observed and predicted (fitted) values, i.e., the $y - \hat{y}(y)$. The systolic blood pressure residuals are represented by vertical lines in the figure below:

```r
systolicFit <- fitted(BP.lm)
plot(age, systolic, xlab="Age", ylab="Systolic Blood Pressure (BP)"
abline(BP.lm)
segments(age, systolic, age, systolicFit)
```

A density histogram of the residuals with a superimposed kernel density estimate is now shown.

```r
systolicRes <- resid(BP.lm)
hist(systolicRes, prob=T, breaks=c(-9, -4.5, 0, 4.5, 9), main = "",
xlim=c(-10, 10), ylim=c(0, 0.1), xlab = "Systolic BP Residuals")
lines(density(systolicRes, bw="SJ"))
rug(systolicRes)
```
Security

A 3-value token is used for auto-logins, which:

- disables an account if someone attempts to hijack a session;
- logs all activity for reports and security auditing.

All communications are done over SSL.
Summary

Rc² is an accessible IDE for writing R code that runs in Docker containers through a client/server architecture. Sessions can be created that support different container configurations, including containers for Hadoop, Hive, Spark, etc. Workflows can be facilitated by constructing containers based on flexible topologies.