Running Hadoop and Spark from R Using Docker Containers Interface 2015

E. James Harner and Mark Lilback

Department of Statistics
West Virginia University

June 11, 2014
Outline

Introduction

Rc² Server

Rc² Client

RStudio

Summary
What data architecture is needed for big data analytics?
A story of two 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.

How do access these big data processing architectures from R?
Rc² Overview

Rc² (R cloud computing) is an iPad and OS X front-end to R which is:

- cloud based with local caches for performances;
- highly scalable;
- collaborative (via shared sessions and workspaces);
- output formatted appropriately to platform;
- mobile interface tailored for the iPad.

Researchers can collaborate over the Internet without concern for code becoming out of sync.

Users can start long-running computations and Rc2 will notify the user(s) when the process is complete.
Overall Architecture

Rc² has a 4-tier architecture:

- **client** iPad and OS X native clients
- **app server** Jetty app-/web-server with Java servlets using technologies such as JPO, WebSockets, and RestKit
- **compute cloud** JSON over BSD sockets for R
- **database** 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, including audio

The three backend tiers run on Linux—clustered or not.
Server Architecture Components

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

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

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

**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

HDFS/Hadooop is accessed from RSession using RHadoop and RHIPE.
In the future, Spark will be accessed using SparkR.
File Change Monitoring

- RSession fetches files from database on init
- RSession monitors files via inotify and sends those changes to the database
- The database has triggers to send appropriate file changed notification
- WebSession notes those changes and sends those changes to client
- RSession notices changes made elsewhere and updates files on the filesystem
Hosting

- $\text{Rc}^2$ currently uses two containers: 1 for Hadoop. 1 for the rest.
- Ideally we should have Jetty, Postgres, CouchDB in one container, which can be scaled using traditional web app scaling methods.
- We envision running each instance of RSession on its own container, ideally managed by Mesos.
- We plan to use ZooKeeper to manage configuration information.
Client

Native client interfaces (UIKit for iOS; AppKit for OS X) are comparable in speed and functionality to desktop R interfaces and include:

- sharable project and workspaces;
- a text editor for .R, .Rmd, .Rnw, .sas, and .txt files;
- a command line for R;
- styled text for console output, native image display, and WebKit for other file types, e.g., html and pdf;
- file and workspace displays;
- a graphics display supporting multiple plots;
- voice chat capability.

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

Projects contain workspaces and shared files and:

- provide the setup of sharing permissions for individual workspaces (defaults to read/write for each user);
- can be flagged as a class (defaults to shared workspaces).

A workspace is a superset of an R workspace. It has a list of associated files (no directories) along with all objects that would be 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. 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 (by dragging in OS X and by importing in iOS). Source files in classroom mode are automatically cloned. Cloning greatly reduces setup and complexity for new users (e.g., students).
Client Interface

$Rc^2$ has three principal screens:

1. a project screen for adding and deleting projects and for adding or deleting shared users;
2. a workspace screen for adding and deleting workspaces and for setting workspace-specific permissions;
3. a work-environment screen for text editing and viewing output.

See the demo.
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.
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.

Rc² has a fine-grained permission system so a student in one class can be a GTA in another.
RStudio Overview

RStudio is a powerful, open-source IDE for R.

RStudio

- provides a productive user interface for R;
- works on all major platforms;
- has a server version for code development over the web;
- supports both Sweave and R Markdown;
- supports interactive web application development using Shiny and Shiny Server.
IDE Features

As an IDE, RStudio:

- supports syntax highlighting, code completion, and smart indentation;
- allows code go be directly executed from the source editor;
- supports integrated R help;
- has a workspace browser;
- has an interactive debugger allowing the developer to find and fix errors quickly;
- has extensive support for developing packages
Projects

RStudio allows the creation of projects.

RStudio projects can be created:

- in a new directory;
- from an existing directory containing R code and data;
- from a version control Git or Subversion directory.

RStudio has support for multiple simultaneous projects.

Version control allows the coordination of team work and benefits individual work.
RStudio supports many tools for package development, including:

- a Build pane with package development commands and a view of build output and errors;
- Build and Reload commands for rebuilding the package and reloading it in a fresh R session;
- R documentation tools including previewing, spell-checking, and Roxygen aware editing;
- integration with devtools package development functions;
- support for Rcpp including syntax highlighting for C/C++ and gcc error navigation.
Summary

Rc² and RStudio target different audiences.

Rc² is an accessible IDE for students and researchers who have limited technical skills. Rc² sessions allow real-time collaboration which is ideal for students taking distance-based courses and researchers in different locations. On the other hand, Rc² is not yet platform independent.

RStudio is a powerful IDE, but its completeness necessarily involves complexity. It does not support collaboration although users could share information using group permissions on the Linux server version.