From Docker to Kubernetes: Scaling Machine Learning Algorithms

E. J. Harner[a] and M. Lilback[b]

[a] *West Virginia University,* [b] *Rc²ai*

Rc² (R Cloud Computing) is a collection of Docker containers for running R and Spark with various persistent data stores, including PostgreSQL, HDFS (Hadoop Distributed File System), Hive, etc. The core components of Rc² are an app server, a database server, and 0 or more R-based compute servers. The user interacts with the app server through a native Rc² client. Currently, the client supports R markdown documents and notebooks. Rc² can be run in single-user mode (running on a local machine or server) or in multi-user mode, typically run on a scalable cloud service.

Kubernetes orchestrates these sets of running Docker containers and volumes, called pods, which are the smallest deployable unit within a cluster. Applications running in the same pod share the same IP address, the same port space, and the same namespace. Although deployment of pods can be done in different ways, e.g., as a traditional high-performance computing environment, the focus here is in scaling the number of Rc² users.

Rc² uses Ingress for managing access by multiple users to the various supported services. This involves user authentication and the spawning of requested Rc² services. The canonical Rc² service is a pod running R and other tightly coupled applications, e.g., Java. Other services, e.g., PostgreSQL and Spark, are shared, stateful services and are made available to the user depending on the login request and the user's permissions. Kubernetes not only ensures that the dependencies among and within the requested pods are enforced, but also that the number of available pods always exceeds the number of running pods by a specified number.

The power and flexibility of the Rc²architecture is illustrated with several machine learning examples, which are available here: https://github.com/jharner/DSSV2019Rc2.

Keywords: R, Spark, Docker containers, Kubernetes, Ingress.