REVOLUTION ANALYTICS

Module 3: Architectural Options of Working with RRE and Hadoop

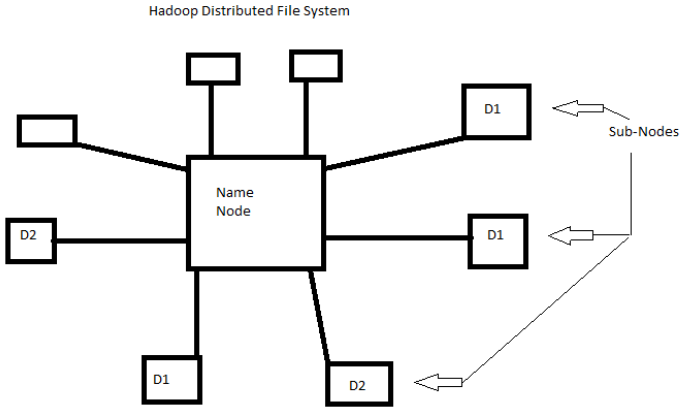
Revolution Analytics

Hadoop and ScaleR

As we have seen before, Hadoop stores information across a number of compute nodes in a cluster. Consequently, data may be split up into many different components, and accessing data is limited by contacting the name node and requesting the location of data within the system.

In the following figure, for aesthetic reasons the name node is located in the center of the sub-nodes on an example Hadoop cluster. Two data sets, DS1 and DS2, are broken up across the system, and require a call to the name node to access their locations. Note that this is a very simplified example; for instance, multiple copies of data are usually stored to defend again hardware failure, and that is not reflected in the figure.

Hadoop Architecture

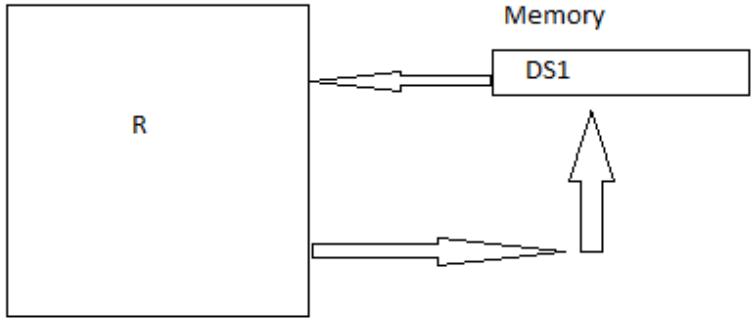


Open Source R Architecture

Conventionally, the data set being used in R commands is defined within the function call. In open source R, this is done by storing the data in memory and running your analysis by defining the data consistent to its location in memory. However, this can be restrictive when dealing with big data.

In the following figure, data is stored in memory on a single node or computer, and is transferred to the Central Processing Unit running open source R in another overly simplified figure (nothing else is running on this computer!). In this case, data is imported from memory, computed, and the result is written back out to memory once again.

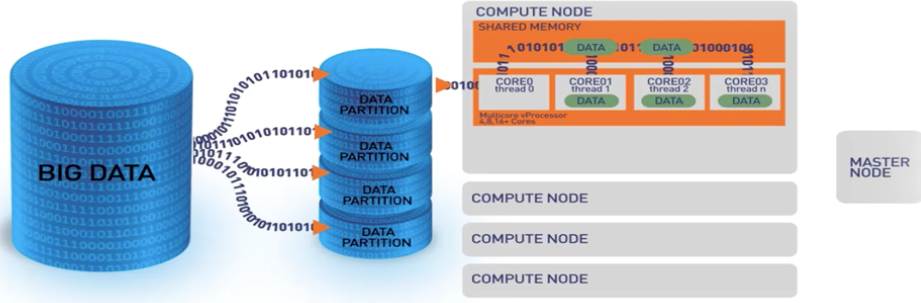
Open Source R Architecture



Revolution R Enterprise Architecture

ScaleR improves this process of handling big data by storing data in an XDF format in hard drive space rather than memory. This improves big data computations on a single computer, but how can we handle computations across a distributed network such as Hadoop?

Revolution R Enterprise Architecture



The Data Source

To solve this problem of distributed computing, ScaleR version 7.0 points to data on the Hadoop cluster by using the data source.

The data source points to data on the Hadoop cluster by establishing a connection with the Name Node which then provides ScaleR with not only the location of the data file, but also in its proper restructured format. Remember that data can be split into many portions, and the name node also contains information on how to properly reconstruct the data to retain its original format.

The Data Source

So how does the data source work in relation to ScaleR?

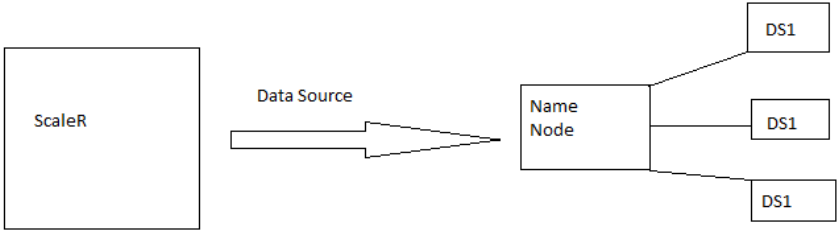
Just as in prior versions where XDF data is loaded as a part of the function call, ScaleR version 7.0 is no different. However, instead of loading a standard XDF data file (which is of course still compatible), the user instead “loads” the data source; that is, the user defines the input data as the data source.

The data source then points to the location and proper restructuring format by linking to the name node, and ScaleR runs executes the function call on this data. Afterwards, the original data set is either over-written, or a new data set is declared on the Hadoop Distributed File System.

The Data Source

The following plot shows a simplified version of the interaction between ScaleR and the Name Node via the data source. The data source accesses the information on the name node containing the data location and proper restructuring format. Consequently this data is reconstructed, computed, and the result is sent back to the specified output format (i.e. R output, an additional file on the HDFS, overwriting the original file, and so forth).

The Data Source



Recap

Let's review some of the concepts covered in this module:

- How does Revolution's ScaleR interact with big data?
- How does the data source relay information from Hadoop to the user?
  - How does ScaleR handle this information and carry out its computation?