Framework Proposal

Songmeng Wang

April 2021

1 What is the problem?

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is known for its prominent performance handling massive data, not including images. Image has relatively small size, which makes it not as suitable as other data type for Hadoop. Each data node in Hadoop has 64M storage. Regular image implementation will put only one image per data node (2M/64M used, 62M/64M unused). Such operation decreases the efficiency of Hadoop. When handling massive images(especially for large-scale images), this deficiency will be magnified.

2 Why is it important?

Images are everywhere nowadays. It's used in social media, videos, monitors and so on. Image processing on massive images has become significant important. For instance, a simple face recognition application requires training on massive images to enhance its accuracy. Images search on website also required searching in massive image data base (more than 300 million images are posted on social media each day). Running time is the core problem dealing with this huge number of images. Since Hadoop provide such a beneficial framework handling large data across clusters, it's time-saving and essential to improve the performance with images processing on Hadoop.

3 Why is it difficult to solve?

First, Hadoop is relatively hard to get started compare to many other framework. Users will need hours to install and create their first program on Hadoop. To create more specific and complex operation on Hadoop, users will need to understand how it functions, especially about MapReduce, which is the most difficult part. Second, images are hard to handle in Java, compared to languages like Python, Ruby, etc. ByteWritable for images in Hadoop is not a friendly/familiar data type for many images processing techniques. Third, framework should provide a compatibility for different types of techniques. The pixel values should be merged for better transaction through HDFS, but on the same time, provide enough information for the Mapper (ex. RGB values).

4 Benefit

Hadoop Image Processing will be beneficial to people who want to deal with massive large scale images. It will be beneficial to face/object recognition, which required training. Search-based processing technique will be easy to implement on this framework, and it will have a better performance (value/string labels are faster to communicate between Mapper and Reducer). It can also benefit image processing starter when dealing with large scale images. It will provide a faster run time than normal processing even when the number of samples is not large (¡100). It can also perform as a convenient implementing tool for Hadoop. It can lower the difficulty of using Hadoop with images.

5 Proposed Solution

First, sort and merge multiple images into one data file. Merging the data into one file for data node without messing with RGB data. File name included in the combined file, but each filename will corresponded to its data through out the process.

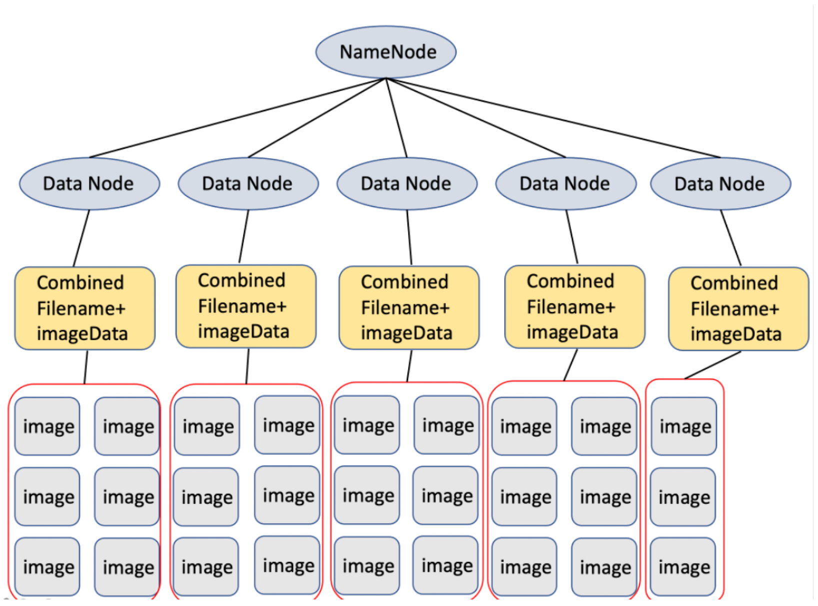
NameNode
Data Node
Data Node
Data Node
Data Node
Data Node
Combined Filename+ imageData
Combined Filename+ imageData
Combined Filename+ imageData
Combined Filename+ imageData
Combined Filename+ imageData
image

Figure 1:

Second, send the merged data file to Data Node. The merged data file should be read, usable image value will be fetch in the Mapper. Reducer should help collecting the values corresponding each files in a combined file.

Third, provide different types of Mapper for different type of image processing. Type differentiate by their input/output data type. Key value remains text for fileName.

Fourth, provide a value search base class for search based processing, enable the searching mapper to convert different data type. Enable string label/integer value from searches to communicate between Mapper and Reducer. Reducer should be able to search based on the preferred output.
Fifth, provide a suitable data type for the filter class. All RGB value should be fetch to the custom filter class. Value should be stored and transport to the Reducer. Reducer converted the data to a desired data type. Additional converter should be included in the file to provide a reader for the ByteWritable data.