A Survey on Hadoop Assisted K-Means Clustering of Hefty Volume Images

Anil R Surve
Dept. of Computer Science and Engineering
Walchand College of Engineering
Sangli, India.
a_nilsurve@rediffmail.com

Nilesh S Paddune
Dept. of Computer Science and Engineering
Walchand College of Engineering
Sangli, India.
nilesh.paddune@gmail.com

***Abstract*— The objects or the overview of the objects in a remote sensing image can be detected or generated directly through the use of basic K-means clustering method. ENVI, ERDAS IMAGINE are some of the software that can be used to get the work done on PCs. But the hurdle to process the large amount of remote sensing images is limitations of hardware resources and the processing time. The parallel or the distributed computing remains the right choice in such cases. In this paper, the efforts are put to make the algorithm parallel using Hadoop MapReduce, a distributed computing framework which is an open source programming model. The introductory part explains the color representation of remote sensing images. There is a need to convert the RGB pixel values to CIELAB color space which is more suitable for distinguishing colors. The overview of the traditional K-means is provided and in the later part programming model MapReduce and the Hadoop platform for K-Means is described. To achieve this parallelization of the algorithm using the customized MapReduce functions in two stages is essential. The map and reduce functions for the algorithm are described by pseudo-codes. This method will be useful in the many similar situations of remote sensing images.**

Keywords- K-Means, color space, remote sensing, parallel, Hadoop, MapReduce, HIPI

I. INTRODUCTION

The K-means clustering is the most recognized method when it comes to the data mining, image processing, etc. The method has been used for the processing of the hefty size remote sensing images. The objects in the remote sensing images can be detected and clustered together. The similarity in the spectrum values of the object is the basis for the clustering of the objects [1]. The K-means algorithm has become the basis of unsupervised classification [2]. But the processing of the hefty size remote sensing images is time consuming process. Though the time complexity of the K-means algorithm is considerable, the execution of hefty volume images is a time and memory consuming process. So the need to optimize the algorithm is arisen. The way to optimize the algorithm can be defined by parallelizing the algorithm. This paper describes the way to migrate the sequential process to parallel computation through the use of Hadoop MapReduce, a distributed framework [1]. The Hadoop is a computing model based on distributed computing [4] and first proposed by Google in 2004 as GFS (Google File System). Nowadays it has made its footprints in many domains in the market.

The rest of the paper will describe the sequential K-means process, the traditional in brief to cluster the remote sensing images. It also describes the color space we used that is RGB, CIE XYZ, CIELAB color spaces [8]. The related work is introduced in the next section. In the fourth section the parallel process is described. Finally the paper is concluded.

II. RELATED WORK

Hadoop is a distributed framework[4] and the MapReduce is the programming model[3] which is implemented using Hadoop. The map and reduce functions can be customized and be written by the users. The Hadoop jobs are carried out using three functions, they are configure, map and close. The process is listed out in below steps.

In paper [9], the algorithm proceeds with following steps:

Step 1: generate initial guess.

Step 2: Map (null, data-instance) -> list (cluster-id, (data- instance, 1)), where the in-key is null, and the in-value is the data instance vector.

Step 3: Reduce: (cluster-id, (data-instance, count)) -> list (cluster-id, (sum-of-data-instances, number-of-instances))

In the output of reduce, the sum-of-data-instances divided by number-of-instances is the mean of the cluster.

III. SEQUENTIAL PROCESS

The proposed work is the serial K-means clustering for the hefty volume images i.e. RS images and the parallel K-means clustering using Hadoop platform.

A color image is used as input for the clustering. The color image is represented using the RGB space. The CIELAB color space is better to understand the similarity between colors than the RGB. Thus it is better to use Lab values than to use the RGB values for clustering. The expected K-means output is the k subset of pixels and within each subset the pixels have most similar color. So for the purpose of clustering the color transformation from RGB to CIELAb is essential. The similarity is represented by the distance between the Lab values.

A. Transformation of colors

The RGB color values first need to be converted to the CIEXYZ values [8]. Then the CIE XYZ color values are converted to CIE LAB values i.e. L*a*b*[7]. L* is the brightness layer, a* is the chromaticity layer which indicates where color fall along red-green axis and b* is the chromaticity layer which indicates where color fall along blue-yellow axis [5].The formula to convert RGB to LAB is given by [6].The MATLAB code to implement the conversion is available on website. The Java function for color conversion is available on [8].

P(L,a,b) is a pixel value in LAB color space. We will make use of only a* and b* values to find the difference of the two colors as the color information is present in only ‘a*b*’ space. The difference can be measured using Euclidean distance. The distance and the difference are directly proportional.

B. K-Means clustering

Let C_1, C_2,….,C_k be the k number of clusters for the algorithm, where k is the input parameter. Let P1, P2,…, Pn be the N number of pixels, where Pi is the i^{th} pixel consisting of a pair (a,b), where a, b are the color components in L*a*b color space. Since the pixels in the image files are arranged along widths and heights like rectangular shapes and each pixel has a two dimensions coordinates, a conversion of pixels to sequential pairs is required. Let m_1,m_2,…,m_k be the centroids of the clusters respectively, where $1\leq i \leq$k.

$$Je = \sum_{j=1}^{k} \sum_{Pi\in Ci} \left|\left|Pi - mi\right|\right|^2 \qquad (1)$$

$$m'i = mi + \frac{1}{Ni-1}[mi - Pi] \qquad (2)$$

$$m'k = mk + \frac{1}{Nk+1}[Pi - mk] \qquad (3)$$

The Je is the error variance defined as (1) [10].To get the optimized solution Je should be minimum. The $m'i$ is the new cluster centroid when the pixel Pi is moved out from cluster Ci (2). Again the new centroid $m'k$ when there is increase in the centroid value when a Pi is moved to a new cluster Ck (3).

C. Steps for K-Means clustering

The steps are as follows:

Step 1. Generate an initial K clusters.

Step 2. Put the pixels in another cluster when the sum of error variance is going to decrease. Select each pixel P, then let Ci be P’s cluster, Ni be the number of pixels in this cluster and mi be the centroid. If Ni =1,then ignore this pixel and select another one, otherwise calculate the variation by (4):

$$Vj = \begin{cases} \frac{Nj}{Nj+1}\left|\left|P - mj\right|\right|^2 \; j \neq i \\ \frac{Ni}{Ni-1}\left|\left|P - mi\right|\right|^2 \; j = i \end{cases} \qquad (4)$$

In the above equation, i is the number indicating the cluster P belongs to, while j varies from 1 to K. $Vj(j = i)$ is a decrease in the sum of error variance when P has be moved out of the cluster Ci . Vj increases if $j \neq i$ ($P\ is\ assigned\ to\ Cj$). If a $j(\ j \neq i)$exists satisfying $Vj < Vi$, that means moving P from Ci to Cj gives rise to a decrease in sum of total error variance, otherwise jump to the beginning of the step. If there exist several j satisfying such condition, the one with minimal increase would be chosen.

Step 3. Move the pixel from Ci to Cj and update mi to mj.

Step 4. Iterate this procedure for N times.

This algorithm stops only when there is no variation of Je took place or maximum number of iterations specified.

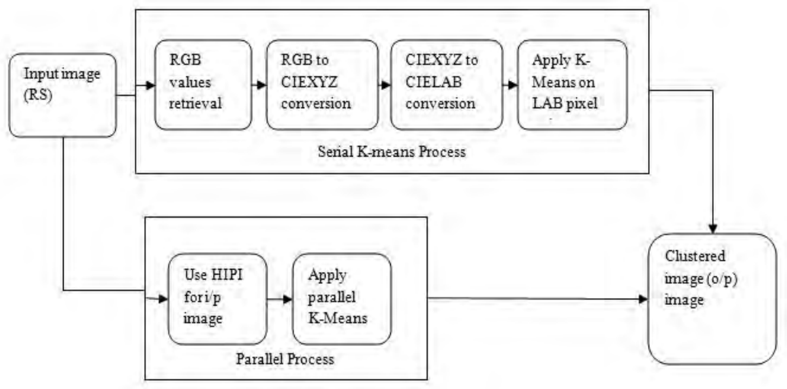

Figure 1. Functional Diagram

IV. PARALLEL PROCESS

The parallelization is done to optimize the time utilized by the clustering process. We are using Hadoop jobs for the parallel processing of the remote sensing images. The dynamic updating of the centroids needs global communication between node and it should be avoided. Hence, in our method pixels are clustered partially. In general, there are two Hadoop jobs for calculations, one is to distribute each pixel into its initial cluster, and another is to move pixels minimizing the error variances. To simply suit the framework, input images are translated into text file. Each line represents a pixel like (file_id, pixel_id, r, g, b), where file_id identifies the file, pixel_id is the line number in each file, and r, g, b are the values [1].

The algorithm is described in two steps [1]:

Step 1: Initial clustering

Step 2: Minimize the total error variance for each cluster

Step 1: Initial clustering includes DETA which is increase in centroid when a pixel is moved into a new cluster and calculated by the 3rd equation provided in the above part.

The input ‘CountsInCluster’ is a counter list in which each item represents the number of pixels that a cluster contains. It is initialized at the configure function. The ‘lab_distance’ method is to calculate Euclidean distance of two pixels.

Step 2: Minimize the total error variance for each cluster. This job reads its input from the files that are produced by the first job. At the configure function of this job, it merges some partial results. For example, centroids of each cluster which are produced by each map task are collected together. And some ‘CountsInCluster’ values are also added together. The map function of the job ‘Minimize Error’ is shown in minimize error step. The idea is to move the pixel to another cluster if the sum of error variance decreases. The error variance of a cluster will decrease when a pixel of it moves out and will increase then a pixel moves in. If the amount of increase is less than that of decrease, the pixel is of course worth moving.

1. InitalClusteringJob.map

```
Input: (in_useless, in_pixel), where ‘in_pixel’ is (file_id, pixel_id, r, g, b);
Centroid{K} = getFromConfiguration(); // assign K initial centroid
CountsInCluster{K} = {1}; // each cluster has 1 pixel at the beginning.
Output: (out_cluster_id, out_ pixel)
rgbPixel = parseFrom(in_pixel);
labPixel = Transform(rgbPixel);
minDistance = Double.MAX_VALUE;
closestClusterId = -1;
for (i = 0; i < K; ++i) {
dist = lab_distance(input, Centroid[i]);
if (dist < minDistance) { minDistance = dist; closestClusterId = i;
 }
}
```

```
++ CountsInCluster [closestClusterId];
Centroid [closestClusterId] = Centroid [cid] + DETA;
out_cluster_id = closestClusterId;
output = {closestClusterId, file_id, pixel_id, labPix}
```

2. MinimizeErrorJob.map

```
Input: (cluster_id, labPixel)
Centroid{K}; //a set of centroids
Counts{K};//Counts[i] indicates the count of pixels in cluster i
Output: (cluster_id, out_ pixel)
// move the pixel to another cluster if error variance decreases
icount = Counts [cluster_id]; idestCluster = -1;
cen = Centroid [cluster_id]; double tempMin = decrease;
decrease = icount / (icount - 1) * lab_distance(labPixel, cen);
for (int k = 0; k < ClusterNumber; ++k) {
if (k != cluster_id)
{
 increase = icount / (icount+1) * lab_distance(input, cen);
 if (increase < tempMin)
{ idestCluster = k; tempMin = increase;}
}
}
if ( idestCluster != -1)
{
cen = cen + (cen - labPixel) / (icount - 1);
cen2 = Centroid[idestCluster];
cen2 = cen2 + (labPixel – cen2) / (icount + 1);
Centroid[cluster_id] = cen;
Centroid[idestCluster] = cen2;
--m_Counts[cluster_id]; ++m_Counts[idestCluster];
output = { idestCluster, labPixel}
}
```

The Hadoop jobs are thus completes the clustering process with customized MapReduce functions. The HIPI interface stays useful instead of writing the pixel values to the text file and then process for the clustering. The Mahout platform which is more scalable for large data processing than Hadoop is also a good choice found in the survey.

V. Conclusion

In this experiment the pixels are clustered and merged to form the clustered image which is different from the standard clustering algorithm in a mathematical perspective. However as the input size increases the deviation slows down enhancing the clustering work. The Hadoop interface HIPI is a very useful in this context as it was the major concern in earlier research work. The main objective of clustering remote sensing images is to obtain broader perspective of the data that has to be used for the further image processing. As stated in the article, primitive tool for visualizing the hefty size images is developed. It has been noted that the Mahout platform that will be better in terms of resource utilization and time optimization as it scalable to large datasets.

References

[1] Zhenhua Lv, Yingjie Hu, Haidong Zhong, Jianping Wu, Bo Li and Hui Zhao, “Parallel K-Means Clustering of Remote Sensing Images Based on MapReduce,” F.L. Wanget al.(Eds.): WISM 2010, LNCS 6318,pp.162-170,2010.
[2] K-Means for unsupervised algorithm: http://www.sciencedirect.com/science/article/pii/S0031320301001388.
[3] The MapReduce programming model: http://research.google.com/archive/mapreduce.html
[4] Hadoop Website: http://hadoop.apache.org/
[5] Matlab R2008a Product Help. Demos/Toolboxes/Image Segmentation/Color-Based Segmentation Using the L*a*b* Color Space.
[6] Kartikeyan, B., Sarkar, A.,et al.: A segmentation approach to classification of remote sensing imagery. International Journal of Remote Sensing 19,1695-1709(1998).

[7] Color space conversions: http://www.cs.rit.edu/~ncs/color/t_convert.html
[8] Color Inspector 3D-color space conversions,
http://www.f4.fhtw-berlin.de/~barthel/ ImageJ/ColorInspector/HTMLHelp/farbraumJava.htm#rgb2lab
[9] Zhao, W., H., et al.: Parallel K-Means Clustering Based on MapReduce. In: CloudCom 2009.LNCS, vol.5931, pp.674-679(2009).
[10] Zhaoqi, B., Xuegong, Z.: Pattern Recognition, 2nd edn., pp. 235-237.Tsinghua University Press, Beijing (2000).