I J C T A, 9(41) 2016, pp. 583-590 ISSN: 0974-5572

REAL TIME COMPUTATION OF CLUSTERING AND DISTANCE MATRIX THROUGH GPU

Vaishali Dabral*, Vikas Tripathi* and Kashaf Khan*

Abstract: **Working with GPU is a subject of great pursuit in computer vision, which is designed to rapidly manipulate and alter memory in order to quickly generate images in the buffer to display. GPUs are effective at image processing and are helpful when processing of large data needs to be done in parallel which is supported by their highly parallel structure and which makes them more efficient than CPUs for activity detection in videos. Datasets which are generated through videos using hog descriptors are of huge size and take a lot of time for processing. So we reinvigorated the algorithms for centroid calculation and dataset generation by distance calculation for HMDB-51 dataset through parallel processing using GPU. This reduced the overall time taken by clustering. Further we used random forest to classify our dataset. We have calculated the overall processing time through our approach as 35.71712 seconds and achieved 48.365% accuracy using 100 trees through random forest.**

Key Words: ***GPU, parallel processing, random forest, centroid calculation, distance calculation***

1. INTRODUCTION

GPUs became more popular and high in demand as graphic applications increased and eventually became a necessity for good performance of any computer system. GPUs have a large number of cores which helps them operate on pictures and graphics much faster than Central Processing Unit (CPU) as they are able to handle multiple tasks simultaneously because processing offered by CPU is sequential. It has less number of cores in comparison to GPU. When processing is done in parallel the computer intensive section of the code is sent to GPU for processing and remaining code runs on the CPU. While dealing with quite large datasets often there is a need to decrease the processing time taken by algorithms working on such datasets. GPU provides graphic resources which include processor and memory. Faster processing speed is achieved since the advent of GPU's as a major part of the code is sent to GPU for processing. With this the load on CPU, system bus and the main memory gets decreased to a large extent which otherwise would get overloaded with operations and I/O requests. GPU represents a 3D world in a realistic way. This paper proposes an effective approach of calculating centroid and distance through parallel processing using GPU. The paper comprises of number of sections 1. Literature Review 2. Methodology 3. Result and Discussion 4 .Conclusion

* *Department of Computer Science and Engineering, Graphic Era University, Dehradun, Uttarakhand, India*

vaishalidabral@yahoo.com , *vikastripathi.be@gmail.com*, *Kashaf17khan@gmail.com*

2. LITERATURE REVIEW

In present time work is mainly being done on enormous datasets and with it comes the need to efficiently handle them. Hence, there is a lot of work being done on clustering techniques. Scalable clustering algorithms can help in working with large datasets [4]. It is difficult at times for a processor to process such large volumes of data at once. With the recent development of affordable parallel computing platforms, scalable and high performance solutions can be readily achieved by the implementation of parallel clustering algorithms [6,12]. The ongoing research in parallel clustering algorithms has demonstrated their implementations can produce great benefits. For large datasets the use of adaptive techniques can be done in order to achieve efficient computation of clusters [13,14]. Efficiency in clustering depends on the number of clusters to be formed. A parallel k-means clustering algorithm was given by Dhillon and Modha [11]. From their analysis, working on different datasets in relation to the size of dataset and the number of clusters, they observed linear scale up and linear relative speedup.

For classification of the datasets we have used random forest [9]. Random forest is an efficient method for matching large datasets. The dataset that we used is HMDB-51 dataset [1]. The article by Stephen, William and Brucek [8] discusses the capabilities of GPU based computer systems and also the challenges faced while dealing with single chipped parallel computing systems [7]. Further working with the GPU and showing the methodology relating to event motion recognition in ATM booths, a new framework was given by Vikas Tripathi [3] that displays a strong security framework system for ATM booth using MHI and Hu moments. Stoffel and Belkoniene [2] demonstrated a linear speedup for heavy datasets, implementing k-means algorithm with around 32 computer systems on Ethernet using parallel approach. Many other approaches [10,5] considering scalability of parallel k-means approach have also been demonstrated. Our objective behind this paper is to decrease the overall processing time of algorithm working on quite large dataset through parallel processing using GPU. An efficient method of clustering has been used to support the same. Proposed method is more effective in clustering as it takes less processing time and achieves more accurate results.

3. PROPOSED METHODOLOGY

We have proposed a method to calculate centroid and distance effectively which is further being processed in GPU in order to decrease the overall processing time as shown in Figure1. Our algorithm comprises of two phases:

1. Centroid calculation using GPU and
2. Distance matrix calculation using GPU

The compute intensive code of centroid calculation is sent to GPU for faster processing. Algorithm 1 described in 1.1 is then run on GPU to compute the centroid matrix which is returned back to CPU. Following this the centroid matrix along with the train file is again sent to the GPU for distance matrix computation using Algorithm 2. Now in the subsequent iterations centroid matrix, train file along with test file will be send to algorithm 2 in the GPU for suitable modifications on the distance matrix. Once clustering is achieved, the resulting train file and test file is send for classification.

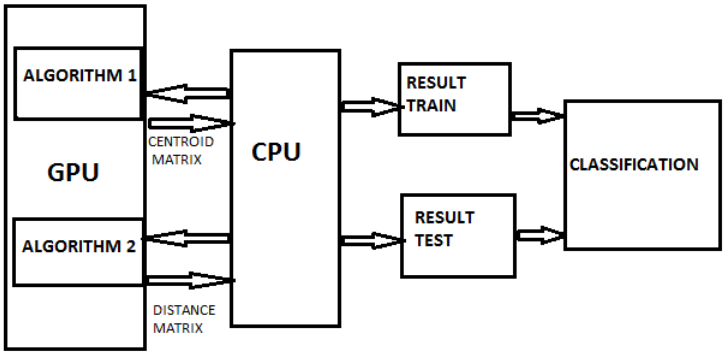

Figure 1. Parallel Processing using GPU

3.1 Centroid calculation using GPU

While working with large datasets like HMDB-51 dataset which is the largest action dataset till date, there is always a need to work on them efficiently and fluently. A processor generally is not able to process such large datasets in less time because it runs it sequentially. Hence by applying a parallel approach using GPU on these large datasets, the processing time can be significantly reduced. Through clustering a dataset can be divided into many subsets. Such that each subset of the dataset i.e. each cluster can be sent to a unique thread for processing as shown in Algorithm 1. Since original algorithm for centroid calculation was difficult to implement on GPU, therefore by making certain modifications on the original algorithm it was made suitable to run on GPU.

In our Algorithm 1 we calculated the value of centroid for each group by sending it to different threads in GPU so that the value of the entire group could be calculated simultaneously. For each group mean is calculated using fixed number of elements from each column of the dataset using different threads. After which their average is calculated to get the final value.

Algorithm 1
Centroid Calculation
Input: HMDB-51 dataset
Output: Centroid matrix

```
m=n=k=l=0;
label 1:
    LOOP if k<clusters
      sum_cen=0;
      k=0;
      label 2 :
          LOOP if n<row_size /cluster&&m<row_size
                sum_cen+=data_train[m][i];
                k++; m++;
      END label 2;
      T=(row_size/cluster);
      centroid_matrix[k][i]=sum_cen/T;

  END label 1;
```

3.2 Distance Matrix Calculation Using GPU

Distance is calculated from each element in the dataset to the centroid. The Euclidean distance formula is used to calculate the distance.

d(r,s)=d(s,r)=sqrt((r1-s1)*(r1-s1)+(r2-s2)*(r2-s2)+….)

where r and s are the element in the dataset and the point of centroid respectively. Since the HMDB-51 dataset used by us is quite large and it is time consuming to calculate the distance from each element in dataset to the centroid, therefore, a parallel approach is used for it as shown in Algorithm 2.

Algorithm 2

ALGORITHM FOR DISTANCE MATRIX

Input: Centroid and HMDB-51 train_set / test_set

```
index = blockIdx.x*blockDim.x+threadIdx.x;
i=j=0;
label 0: if index<no._of_threads
    label 1: loop if j<130
        Initialize sum to 0.0;
        Label 2: loop if z<499
            sum=sqrt(abs((data[index][j]*
                    data[index][j])-(centroid[z][j]*
                     centroid[z][j])))+sum;
        END label 2;
        Distance_ matrix[index][j]= sum;
    END label 1;
End label 0;
```

In Algorithm 2 we calculate the distance matrix by subtracting each element of data matrix row wise by the elements of centroid matrix and simultaneously adding the resulting values. All these calculations are done using threads as done in centroid calculation.

The dataset was divided in different groups and for all groups the process was repeated. Then all the groups are combined to form a single resulting distance matrix which is taken as input for classification using random forest.

4. RESULT AND DISCUSSION

The system we have used for all the computation and testing purpose is of 2.4GHz and has Intel CORE i5 processor with 4GB RAM and 92 cores .We have applied our algorithm on HMDB - 51 dataset which has been extracted from a large video database created for human motion recognition.HMDB-51 is created from the largest action video till date that has 51 action classes which contains around 7000 annotated clips gathered from a variety of sources. Table 1 shows the detailed accuracy of our approach.

The detailed accuracy report presented below compares the process of centroid and distance calculation when computed in CPU and GPU. With clustering, the three main divisions are :

- Centroid calculation using k-means and distance calculation in CPU and GPU respectively.
- Centroid calculation using k-means and distance calculation in CPU.
- Centroid calculation using Algorithm 1 and distance calculation in GPU.

The comparison is primarily based on the accuracy achieved through the various cases and the mean precision, mean recall and mean F1 thus obtained for each case. From this report we can conclude that with clustering, computation of centroid and distance in GPU gives maximum accuracy.

Table -1 Detailed accuracy report

	Accuracy	**Mean Precision**	**Mean Recall**	**Mean F1**
With clustering (centroid in CPU using K-means and distance in GPU)	**46.46%**	**51.63%**	**46.63%**	**46.65%**
With clustering (centroid using K-means and distance in CPU)	**42.087%**	**43.62%**	**40.32%**	**40.29%**
With clustering (centroid using algorithm 1 and distance in GPU)	**48.365%**	**52.87%**	**48.16%**	**48.12%**

Following this a bar graph was plotted between accuracy and the three cases, displaying the comparison of accuracies among the different cases.

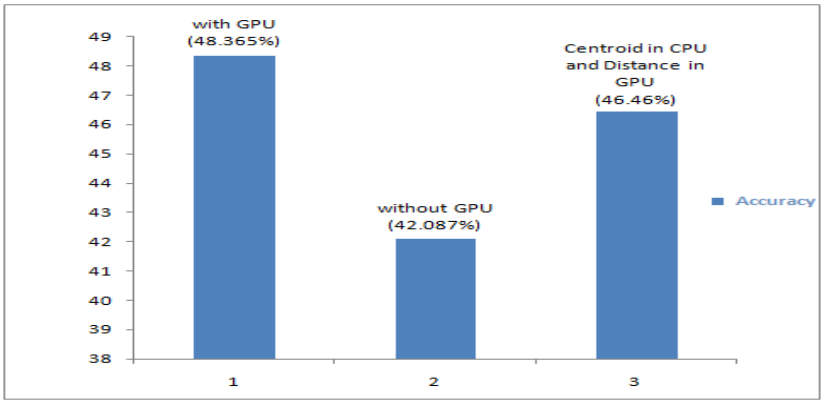

Figure 2. Comparison of accuracy

Accuracy is measured by equation 1, Precision is measured by the formula given in equation 2 and Recall is measured using equation 3. Table 2 displays the result of our approach, which comprises of the time taken by processing and the accuracy. Figure 2 shows that our approach takes less processing time in comparison to the algorithm that goes without GPU approach. Figure 3 and Table 2 shows that the accuracy of our approach is better than the one which works without GPU algorithm due to slight modifications done in the centroid calculation as shown in Algorithm 1. We have shown how making the process of centroid calculation and using GPU has decreased the overall processing time. In Figure 4 we have shown the comparison between accuracies against various classes with and without using GPU.

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN) \quad (1)$$

$$\text{Precision} = TP/(TP+FP) \quad (2)$$

$$\text{Recall} = TP/(TP+FN) \quad (3)$$

Where,

TP=>True Positive

TF=>True Negative

FP=>False Positive

FN=>False Negative

The following Table 2. analyses the processing time and accuracy achieved by computing centroid and distance in CPU and GPU. From the table it is apparent that the processing time drops significantly when centroid and distance are computed in GPU.

Table- 1 Accuracy and time report

	With GPU (centroid using Algorithm 1 and distance calculation)	**With CPU (centroid using k-means and distance calculation)**	**Centroid using k-means in CPU and distance using GPU**
Processing time (in seconds)	**35.71712**	**27977.005**	**5356.348**
Accuracy	**48.365%**	**42.087%**	**46.46%**

The bar graph given below gives the statistical representation of Table 2.

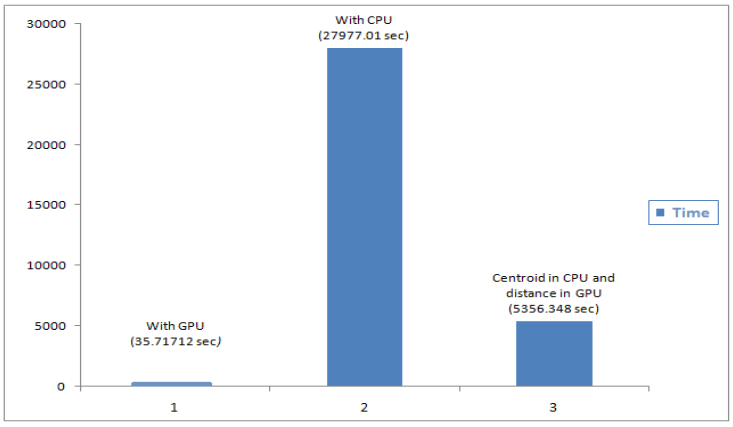

Figure 3. Comparison of processing time between algorithm

Figure- 4 gives the diagrammatic representation of the comparison among the accuracies of the various classes with or without using GPU.

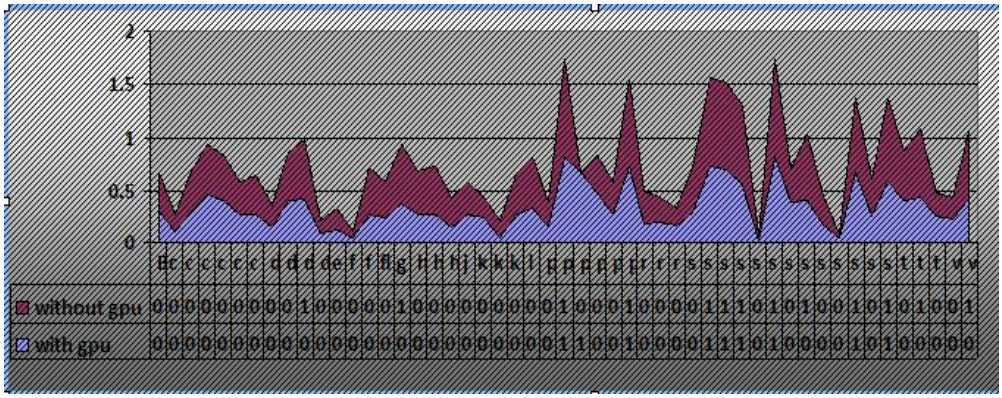

Figure- 4 Comparison between accuracies against various classes with and without using GPU.

5. CONCLUSION

We have proposed a novel approach for centroid and distance calculation in clustering. To test effectiveness of our algorithms we have used HMDB-51 video dataset. We have investigated the processing time of HMDB-51 dataset on CPU and GPU from which we discovered that processing of HMDB-51 on CPU takes substantially more time compared to its processing on GPU. Therefore centroid and distance calculations are further being processed in GPU in order to decrease the processing time of the HMDB-51 dataset. As a result we were able to decrease the processing time by 27941.29 seconds approximately and increase the accuracy by 6.278%. Thus by achieving an overall decrease in processing time and a significant increase in accuracy, we can say that our proposed methodology has been proven to be efficient. Results achieved by our framework conclusively demonstrate that it can be used for several applications. Further framework is still open for optimization which will enhance effectiveness of processing.

References

[1] Kuehne, H. Jhuang, H. Stiefelhagen, R. Serre T," HMDB: a large video database for human motion recognition," Proc. IEEE International Conference in computer vision (ICCV),pp. 2556-2563,2011.

[2] S. Kilian, and A. Belkoniene," Parallel k/hmeans clustering for large data sets, " Euro-Par'99 Parallel Processing. Springer Berlin Heidelberg, pp. 1451-1454, 1999.

[3] V. Tripathi, D. Gangodkar, V. Latta, and A. Mittal, "Robust Abnormal Event Recognition via Motion and Shape Analysis at ATM Installations," Journal of Electrical and Computer Engineering, 2015.

[4] V. Ganti, J. Gehrke and R. Ramakrishnan, " Mining Very Large Databases," Computer, vol. 32, no. 8, pp. 38–45, 1999.

[5] M. K. Ng and H. Zhexue, "A Parallel kPrototypes Algorithm for Clustering Large Data Sets in Data Mining." Intelligent Data Engineering and Learning,vol. 3, pp. 263–290, 1999.

[6] H. Nguyen," GPU Gems," Addison Wesley Professional, vol.3, 2007

[7] NVIDIA Corporation. CUDA Programming Guide Version 2.0, 2008

[8] S.W. Keckler, W.J. Dally, B. Khailany, M. Garland, and D. Glasco," Nvidia GPU'S AND THE FUTURE OF PARALLEL COMPUTING," IEEE Micro, 2011.

[9] L. Breiman, "Random forests," Machine learning, vol. 45,no.1,pp. 5-32 , 2001.

[10] H. Nagesh, S. Goil and A. Choudhary, "A Scalable Parallel Subspace Clustering Algorithm for massive datasets," proceeding of the conference on parallel processing,pp.477-483,2000.

[11] I.S. Dhillon and D.S. Modha, “A DataClustering Algorithm on Distributed Memory Multiprocessors.” Large-Scale Parallel Data Mining. Lecture Notes in Artificial Intelligence,vol.1759,pp.245–260,2000.

[12] D. Judd, P. McKinley and A. Jain, “LargeScale Parallel Data Clustering,” Proceedings of the International Conference on Pattern Recognition pp. 488–493, 1996.

[13] S. Goil,H. Nagesh,and Alok Chaudhary,”MAFIA:Efficient and scalable subspace clustering for very large datasets,”proceedings of the international conference on parallel and distributedcomputing,1999.

[14] M. Ester, H.P. Kriegel, J. Sanders, and X. Xu,”A density based algorithm for discovering clusters in large spatial databases with noise,”In proceedings of the 2nd international conference in knowledge discovery in databases and datamining,1996.