International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

GPU Accelerated Clustering Techniques

Komal D. Nistane[1], Shailendra W. Shende[2]

[1]Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India

[2]Professor, Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India

Abstract: ***Clustering is the data mining technique which is used to place the data elements into related groups. With the development of technologies the size of the data is increasing rapidly. Graphics Processing Units (GPU) are the high performance parallel processors. GPU has the ability to execute different tasks independently however at the same time with the help of every single processor. This computational feature of GPU can be used to increase the speedup of the data mining algorithms. This paper shows the comparative study of the k-means and k-medoid clustering on GPU.***

Keywords: k-means clustering, k-medoid clustering, graphics processing unit (GPU), CUDA

1. Introduction

The process of grouping a set of physical or abstract objects into classes of similar objects is known as clustering. Clustering analysis has many applications for data processing [7]. In machine learning, cluster analysis is considered as the unsupervised learning. k-means and k-medoid are commonly used partitioning algorithms and are applied in pattern recognition, data analysis, information retrieval, market analysis, image processing and feature learning [8][9].

In order to have faster execution of data miming algorithms parallel computation is useful to reduce overall computation time. Graphics processors (GPUs) have developed very rapidly in recent years. At present time, many computers are assembled with programmable graphics processing units (GPUs). GPUs have powerful Single Instruction Multiple Data (SIMD) processors that can support parallel data processing and high-accuracy computation. CUDA device has Multithreaded architecture which runs many threads parallel and makes the optimum use of available computation power of GPU.

The paper is organized as follows. Section 2 presents the related works of GPGPU. Section 3 describes GPU-based k-means and k-medoid algorithms. The experimental results of our approach are reported in Section 4. Finally, conclusions are drawn in Section 5.

2. Related work

BAI Hong-tao[2] presented the approach for the parallelization of the k-means clustering in which the data object assignment and k centroid recalculations are done on the GPU in parallel manner.

In [1], the algorithm is executed in master slave model. Master thread prepares the data points and uploads them on GPU. In labeling stage the nearest centroid for each data set is calculated. The index of these centroids are stored and used by the centroid update stage on CPU. Hence each thread works on each data point and calculates the nearest centroid. The task of each thread is to calculate and iterate over data points belong to the thread according to partitioning schema and performs the actual labeling for the current data point.

Kaiyong Zhao [4] designed two different GPU-based algorithms: one for low-dimensional data sets and another one for high-dimensional data sets. One data point is dispatched to one thread and then the thread is responsible for the calculation of the distance from one data point to all the centroids, and maintains the minimum distance and the corresponding centroid. As *n* data points has been delivered to *n* threads working in parallel decreases the time consumption significantly. The result of the first step, the data points belonging to the same centroid constitute one cluster. The new centroids are calculated by taking the mean of all the data points in each cluster. The final centroids are calculated on the CPU. When the size of dataset grows larger than a single GPU's on-board memory, a divide-and-merge method is adopted as follows: load the dataset group by group, then compute the temporary results and merge them into the finally results.

D Roberts [5] has given the CUDA implementation of k-medoid clustering. According to the auther the CUDA implementation of this algorithm is nearly similar to the sequential algorithm except the code is parallelised with the help of CUDA cores.

3. Graphics Processing Unit

GPU is a set of multiprocessors executing concurrent threads in parallel. The threads are grouped in thread blocks. Grid is a collection of blocks. Thread and blocks are given an id according to its position in block and grid respectively. The thread and the block id can access specific memory during run time. Each thread on the GPU executes the same procedure known as a kernel.

Each multiprocessor has on-chip memory which include a set of local 32-bit *registers* per processor, Each processor shares parallel data cache or *shared memory* and is where the shared memory space resides. A read-only region of device memory i.e. a read-only *constant cache* is shared by all scalar

processor cores and speeds up reads from the constant memory space.

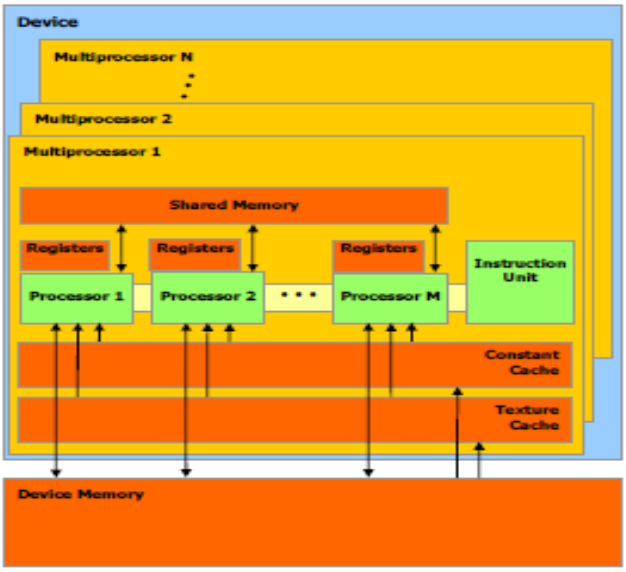

Figure 1: GPU based CUDA architecture

All scalar processor cores shares a read only texture cache and speeds up reads from the texture memory space, which is a read-only region of device memory; each multiprocessor accesses the texture cache via a *texture unit* that implements the various addressing modes and data filtering [10].

4. GPU Based Clustering Algorithm

4.1 GPU-based k-means Clustering Algorithm

k-mean is a clustering algorithm arranges the data points into k clusters having the maximum similarity function. The algorithm assigns each data point to the cluster whose center is nearest. The center is the mean of all the data points in the cluster.

The k-means algorithm does not yield the same result after each run, as the resulting cluster depends upon the initial random assignments. The algorithm minimizes the intra-cluster distance but does not provide the result with the global minimum value [6].

The parallel k-means algorithm is implemented as follows:

1) Enter the k value
2) Randomly select the k clusters from dataset X
3) Copy all the dataset on GPU
4) Repeat
 4.1) calculate the distance of each data item
  for all xi ε Xthread do
  li<- arg min D(cj,xi)
  End for
 4.2) for all xi ε X do
  Add xi to appropriate cluster (ci)
  Calculate the no of elements in ci
  End for
 4.3) for all cj ε C do

$$Cj = \frac{1}{m_i} c_i$$

End for

4.4) until same clusters are found

5) Copy the results on CPU.

In GPU, processors are called threads and a master-slave model is used. Thread 0 worked as the master thread while all other threads are slaves.

The master thread initializes the centroids. Next X is partitioned into subsets Xi. All threads execute the labeling stage for their partition of X. The label of each data point Xi is stored in a component li of an n-dimensional vector. Depending on the distance calculated the xi is assigned to the appropriate clusters. A k dimensional vector m is updated in every iteration where each component mj holds the number of data points assigned to cluster Cj. Next another loop over all centroids is performed scaling each centroid cj by 1/mj giving the final centroids. Convergence is also determined by the master thread by checking whether the last labeling stage introduced any changes in the clustering. Slave threads are signaled to stop execution by the master thread as soon as convergence is achieved [1].

4.2 GPU-based k-medoid clustering algorithm

k-medoid is representative object-based technique. Partitioning Around Method (PAM) was one of the first k-medoid algorithm. It attempts to determine k partitions for n objects. After an initial random selection of k medoids, the algorithm tries to make a better choice of medoid repeatedly. Instead of taking the mean value of the data object in a cluster as a reference point, a medoid is used which is most centrally located in a cluster [11].

k-medoid algorithm is as follows:

1) Enter the number of clusters
2) Copy the data and the cost calculator on GPU
3) Calculate the cost for each medoid on GPU
4) Select the non-medoids randomly and calculate the new cost for the non medoids
5) Calculate the membership value on GPU
6) Copy the membership values on host

For each event in single cluster each thread block calculates the membership value.

5. Experimental Results

In this chapter the results are taken on 2.0 Ghz four core Intel Nehalem, 16 GB RAM with NVIDIA quadro FX 380LP graphics processor. The results are taken on GROCERY attribute with data transfer time and without data transfer time using k-means and k-medoid algorithm. WITHOUT DT means speedup taken without considering data transfer time and WITH DT means speed taken with considering data transfer time.

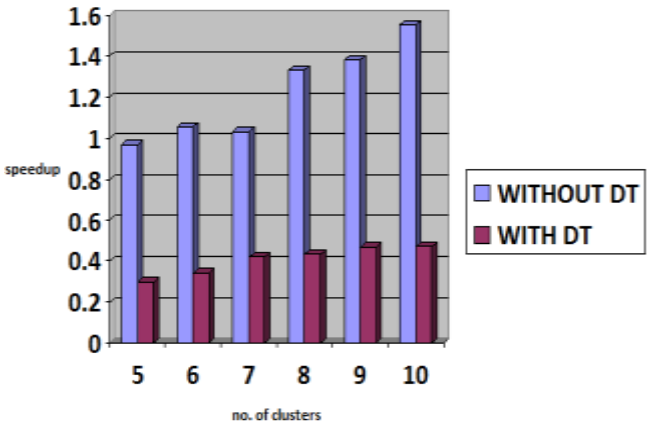

Figure 2: Speedup of k-means algorithm with and without data transfer time

The graph shown in figure 2 is the graph of speedup for k-means clustering algorithm with data transfer time and without data transfer time. The levels on the X-axis indicates that the no of clusters. On the Y-axis there is a speedup factor. The blue bar indicates the gpu speedup without data transfer on gpu and the magenta bar indicates the speedup with data transfer. So here we can see that as the levels are increased the speedup also increases.

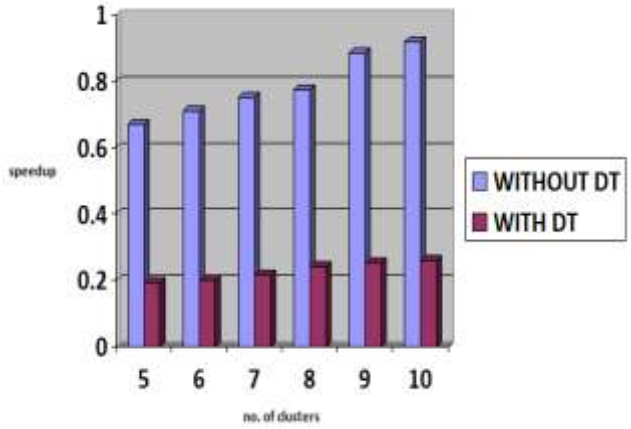

Figure 3: Speedup of k-medoid algorithm with and without data transfer time

The graph shown in figure 3 is the graph of speedup for k-medoid clustering algorithm with data transfer time and without data transfer time. The levels on the X-axis indicates that the no of clusters. On the Y-axis there is a speedup factor. The blue bar indicates the gpu speedup without data transfer on gpu and the magenta bar indicates the speedup with data transfer. So here we can see that as the levels are increased the speedup also increases.

The graph shown in figure 4 is the graph of speedup for k-means and k-medoid clustering algorithm without data transfer time. The levels on the X-axis indicates that the no of clusters. On the Y-axis there is a speedup factor. The blue bar indicates the gpu speedup for k-means clustering algorithm on gpu and the magenta bar indicates the speedup for k-medoid clustering algorithm. More speedup is achieved for k-means clustering as compared to k-means clustering.

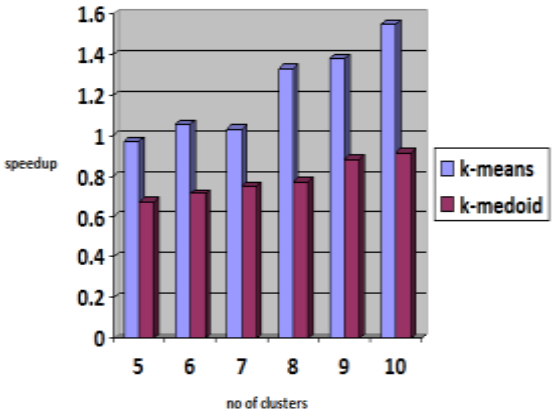

Figure 4: Speedup of k-means and k-medoid algorithm without data transfer time

6. Conclusion

In this project, the serial and parallel performance of k-means and k-medoid clustering algorithm is compared with respect to time. From the investigation and results we conclude that Parallel results are better than serial results with respect to time. Parallelizing has increased the performance. Speedup factor for k-means clustering is 1.55 for k=10 .Speedup factor for k-medoid clustering is 0.91 for k=10.

References

[1] M. Zechner and M. Granitzer, "Accelerating K- Means on the Graphics Processor via CUDA," Proc. First Int'l Conf. Intensive Applications and Services (INTENSIVE '09), pp. 7-15 , 2009, doi: 10.1109/INTENSIVE.2009.19)
[2] BAI Hong-tao, HE Li-li, OUYANG Dan-tong , LI Zhan-shan ,LI He "K-Means on commodity GPUs with CUDA" 2009 World Congress on Computer Science and Information Engineering.
[3] Reza Farivar, Daniel Rebolledo, Ellick Chan, Roy Campbell, "A Parallel Implementation of K-Means Clustering on GPUs" Proceedings of 2008 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 08)
[4] Kaiyong Zhao ; Xiaowen Chu ; Jiming Liu," Speeding up K-Means Algorithm by GPUs" , Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on June 29 2010-July 1 2010
[5] Roberts, Douglas. "k-medoids: CUDA Implementation." (2009).
[6] https://en.wikipedia.org/wiki/Cluster_analysis
[7] Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. "A roadmap of clustering algorithms: finding a match for a biomedical application. Brief Bioinform", pages bbn058+, February 2009.
[8] X.J. Wang, "K-means clustering for multispectral images using floating-point divide". Proceedings 2007 IEEE Symposium on Field-Program Custom Computing Machines (FCCM 2007), pp.151-59.
[9] H. Zhou, Y.H. Liu, "Accurate integration of multi-view range images using k-means clustering". Pattern Recognition 2008, 41(1), pp.152-75.

[10] Nvidia "NVIDIA CUDA Programming Guide" version 2.1 2008.
[11] Dr Batra, Aishwarya. "Analysis and Approach: K-Means and K-Medoids Data Mining Algorithms." *5th IEEE International Conference on Advanced Computing & Communication Technologies [ICACCT-2011] ISBN*.
[12] Nvidia "NVIDIA CUDA Getting Started Guide for Microsoft Windows" version 5.0 2012.
[13] J. Han, M. Kamber. Data Mining: concepts and techniques, Beijing: China Machine Press, 2006.