DOI: 10.1051/itmconf/20171203048

Parallelization of K-Means Clustering Algorithm for Data Mining

Hao JIANG[a], Liyan YU[b]

College of Computer Science and Engineering, Southeast University, Nanjing, China
[a]hjiang@seu.edu.cn, [b]yly.sunshine@qq.com

Abstract: In this paper, we studied the parallelization of K-Means clustering algorithm, proposed a parallel scheme, designed a corresponding algorithm, and implemented the algorithm in GPU environment. The experimental result shows that the GPU-based parallelization algorithm has a good acceleration effect compared with the CPU-based serialization algorithm.

1 Introduction

Cluster analysis is an important research topic in the field of data mining.[1] Clustering is the process of classifying a set of physical or abstract objects into similar object classes. The objects in the same cluster have high similarity, and the differences in the different clusters are large. The automatic clustering is able to identify dense and sparse regions in the object space, and then find interesting correlations between the global distribution pattern and the data attributes. [2] In the context of the application of large data and increasing mass data, the data size has reached TB or even PB level, which puts forward a higher requirement of the cluster. Massive data and huge processing tasks can not be completed by the general computer within the specified time. In order to improve the ability to deal with massive data and enhance the real-time data processing, parallelization of the clustering algorithm become an attractive choice. Today's widely used methods are distributed computing systems, such as Hadoop[3] and Spark[4], which combine multiple computers into a unified distributed system, and each computer processes user data to improves processing efficiency. In order to fully tap the computing power of a single computer, you can transplant the appropriate parallel computing algorithm to GPU platform, and with the GPU's powerful parallel computer capability the computer's data computing capabilities will be improved. Similarly to the CPU, you can easily add a number of GPU to the same computer, to further improve the efficiency of a single computer data processing. It is not difficult to imagine that if a multi-GPU computer with powerful computing power is organized into a distributed cluster, it will greatly improve the data processing capacity compare to the same level of computer clusters. Therefore, the parallelization of GPU-based K-Means algorithm has a wide range of practical application value.

2 Overview of K-Means Algorithm

The K-Means[5] algorithm is one of the most famous and most commonly used clustering algorithms proposed by MacQueen. The algorithm is simple, fast and easy to implement. The K-Means algorithm uses K as the input parameter to divide the N sets of objects into K clusters, which makes the similarity in the same cluster high, and the similarity between different clusters is low. The similarity of clusters is about the mean measure of the objects in the cluster, and can be regarded as the centroid or center of gravity of each cluster. The K-Means algorithm can be described as follows: Given a sample set *D={x_1,x_2,⋯, x_N}*, and $x_i \in \chi \subseteq R^n$ is the eigenvector of the instance, the K-Means algorithm is designed to map N samples to *k(k≤N)* clustering centers *C={c_1,c_2,⋯, c_k}*, and make sure that the square sum of the distance between each sample and its nearest clustering center is minimized, and square sum of that distance is called The Square Error Function, marked as E, as shown in equation (1).

$$\mathrm{E} = \sum_{i=1}^{k} \sum_{x_i \in C_i} \| x_i - C_i \|^2 \qquad (1)$$

The processing flow of the K-Means algorithm:

Input: Data set $D=\{x_1, x_2, \cdots, x_N\}$, and $x_i \in \chi \subseteq R^n$; k is the number of clusters.

Output: a collection of k clusters.

Step:

1) Arbitrary select K samples from D as the center of initial cluster;
2) Repeat;
3) Calculate all the distance between each data sample and the center of the cluster;
4) Assign each object to the most similar cluster according to the above distance;
5) Calculate the mean of all the objects in each cluster and update the cluster mean;
6) Calculate the Squared Error Function;
7) Until the criterion function satisfies the threshold;

8) The algorithm terminates.

The K-Means algorithm attempts to determine the k divisions of the minimized Squared Error Function. When the cluster shape is compact, the difference between cluster and cluster is obvious, and size of all the clusters is similar, clustering result is ideal. The time complexity of the K-Means algorithm is O(NKt), where N is the sample set size, K is the number of clusters, and t is the number of times of iteration. [6] Specifically, the most of time, in the process of the K-Means algorithm's each iteration, spent on calculating the distance between each data set objects and calculating the Square Error Sum of all objects. Although the process of alloction each data object and the update of all k clusters center need to be executed many times, but the most heavy calculation in each time is to repeat the calculation of the Square Error Sum between data and different center points, and calculate the new center point and find the new center of each cluster. So, it can be separated into two single kernel functions. Then, it can be sent to the GPU processing core for parallel computing.

3 Parallel Design of K-Means Algorithm

First, this paper presents a GPU-based K-Means parallel algorithm--G-K-Means algorithm. The main idea of that algorithm is to improve the performance by moving the part which data is independent and computation is intensive in the traditional K-Means algorithm from the host to the GPU. Since the K-Means algorithm is an iterative convergence process which will calculate the distance between the data and the center of the cluster every time, the main parallel scheme includes the following two aspects:

(1) Parallel calculation of all the distance between data objects and cluster center point

In order to facilitate the data calculation on the GPU, we construct the data set matrix T[N][D], the center point matrix C[k][D] and the distance matrix Dis[N][k]. N is the size of the data set, k is the number of clusters to be classified, D is the dimension of the data sample, Dis[N][k] is the result from T[N][D] and C[k][D] set the C^T [D][k] matrix. Each row in the distance matrix Dis[N][k] represents the distance between a data object and center points of k clusters. This step can be transplanted to GPU, and use similar block matrix multiplication to parallel computing. Finally, it can improve the efficiency of distance calculation between the data samples.

(2) Parallel calculation of the Square Error Sum for all objects in the data set

In the process of the iterative convergence of the K-Means algorithm, each iteration needs to calculate the Squared Error Sum of all data objects in order to determine whether it is converge. First, for each object of each cluster, the operation of calculating the square sum of the distance from the object to its cluster center is independent, so parallelism can be achieved here. Second, the calculation of Square Error Sum for each object in each cluster is independent, parallelism can be achieved too. In the specific implementation, GPU grid is divided into N/1024 one-dimensional parallel blocks, each block has 1024 one-dimensional thread. Each thread corresponds to an object for calculating the Square Error Sum with its cluster center, and then use Reduction Thought to sum each thread result. During the Reduction, the thread in each block is first summed, and the Square Error Sum of the object in each block is obtained. Then, the same method is used to Reduction Sum each block and get the final result.

Although the calculation model of SIMD in GPU is good at parallel computing, the GPU-based K-Means algorithm has three important principles[7]. Firstly, GPU branch control and data cache mechanism are very weak, because a large number of computing processing unit occupies most of the space within GPU. Secondly, the data transfer rate between GPU and GPU's global memory is much slower than the data transfer rate between CPU and CPU cache, so with the appropriate thread block and thread bundle size, GPU can reflect powerful computing speed. Thirdly, the GPU-based K-Means algorithm increases the time when transfering data between GPU global memory and CPU memory compared to the traditional K-Means algorithm. Therefore, in order to optimize the performance of the algorithm, we must reasonably allocate the responsibilities of the host and the device, and design or implement the data storage and parallel computing model. The main flow chart of GPU-based G-K-Means algorithm is shown in Fig. 1.

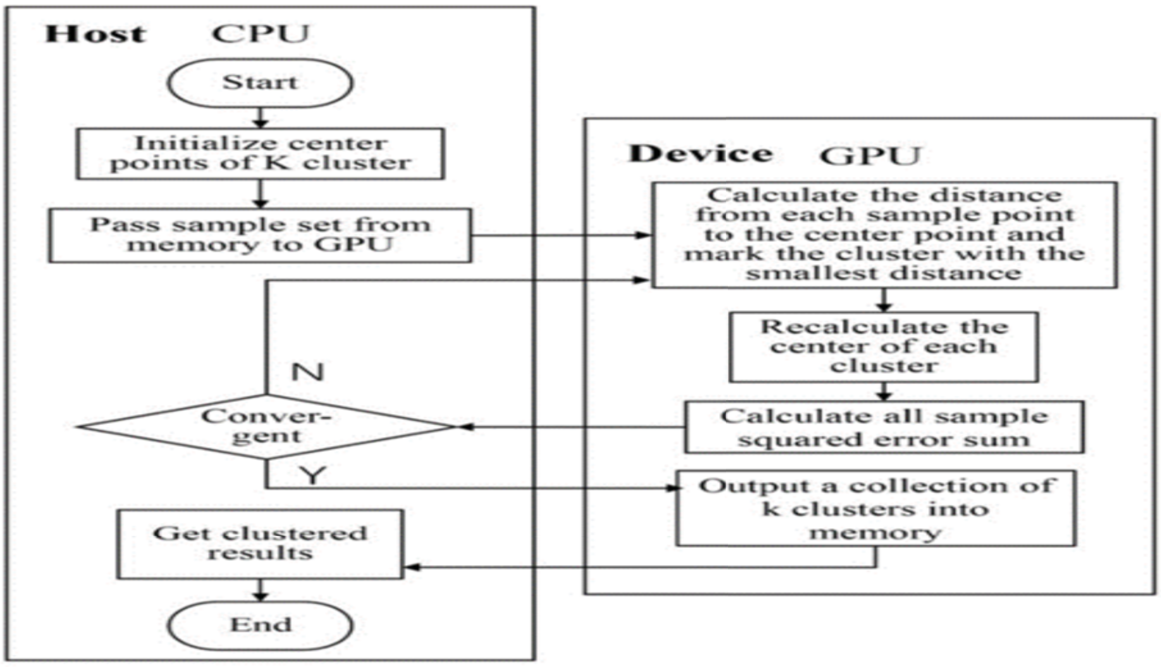
Host CPU
Start
Initialize center points of K cluster
Pass sample set from memory to GPU
N
Conver-gent
Y
Get clustered results
End
Device GPU
Calculate the distance from each sample point to the center point and mark the cluster with the smallest distance
Recalculate the center of each cluster
Calculate all sample squared error sum
Output a collection of k clusters into memory

Figure 1. G-K-Means algorithm flow.

Algorithm process is as following:

1) Initialize the convergence threshold ε, the data set matrix $T[N][D]$ and the random selected k samples' central point matrix $C[k][D]$, where D is the data sample dimension. And calculate Squared Error Sum E_1;

2) Pass the matrix T and transposed C^T into the GPU;

3) In the GPU, use the sample matrix T and C^T to calculate the distance matrix $Dis[N][k]$ of each sample and center point in parallel by matrix operation;

4) According to the distance matrix $Dis[N][k]$, each data sample is marked to the smallest distance cluster, and the center point of each cluster is recalculated, and then the center point transpose matrix C^T is updated;

5) According to matrix $Dis[N][k]$ and matrix C^T, calculate the sample Square Error Sum E_2, then determine whether it is convergent, if $|E_2 - E_1| < \varepsilon$ is satisfied, then turn to 6), otherwise turn to 3);

6) Output cluster results then end.

4 Experiment and Analysis

The experimental data are randomly generated when comparing the GPU-based G-K-Means and the CPU-based K-Means. The experiment is divided into two groups. In the first group, data set T has a size N=100000, a data dimension D = 100.And in the other group, there are several data set ranging from hundreds KB to hundreds MB. Since the efficiency of the K-Means algorithm is affected by the value of k, the first group of tests is taken k from 2 to 1024, and the second group keep 128 as the value of k in order to observe the speedup ratio of parallel algorithm compare to serial algorithm.

In the experiment, the convergence threshold ε is 0.001 and the number of iterations is less than 500. Table 1 describes the results of the first set of experiments, including the convergence time of the two algorithms, the number of iterations and the speedup ratio. Table 2 describes the results of the second set of experiments.

Table 1. N=100000, D=100 G-K-Means&C-K-Means clustering time compare

K	time/s		Iteration times	SpeedUp
	G-K-Means	K-Means		
2	11.386	11.186	133	0.98
4	15.446	24.914	177	1.62
8	19.089	55.645	222	2.92
16	12.339	67.049	141	5.43
32	13.236	136.298	148	10.30
64	12.580	268.150	149	21.32
128	8.307	330.629	93	39.80
256	5.61	417.006	58	74.33
512	3.607	527.611	36	146.27
1024	3.212	689.322	23	214.61

Table 2. k=128 G-K-Means&K-Means clustering time compare

	time/s		Iteration times	SpeedUp
	G-K-Means	K-Means		
0.3	2.013	1.357	33	0.67
0.8	0.907	0.437	14	0.48
1.8	1.472	1.420	24	0.96
5.0	1.656	4.259	26	2.57
11.2	2.798	18.159	47	6.49
23.5	5.61	90.714	93	17.58
50.0	8.396	234.688	121	27.95
116.0	7.804	316.495	87	40.56
232.0	8.834	516.551	73	58.47

From the experimental results above, we can see that the clustering time of the algorithm is influenced by the value of k and the number of iterations. The G-K-Means parallel algorithm has a significant improvement over the clustering convergence rate compared with the K-Means algorithm. In particular, with the increase of k in clustering and the increase of clustering data, the speedup effect of parallel algorithm is more obvious. It can be seen from Fig. 2 that the clustering time difference between the two algorithms is very small when the value of k is small, and the advantage of G-K-Means algorithm is not obvious, but with the increase of k, the clustering time of G-K-Means algorithm increase slowly compared with the K-Means algorithm, and it converges slowly. When the k is 1024, G-K-Means algorithm achieves 214.64x faster than the K-Means algorithm. The K-Means algorithm has nothing to do with the parallel running time on the GPU and the K value.

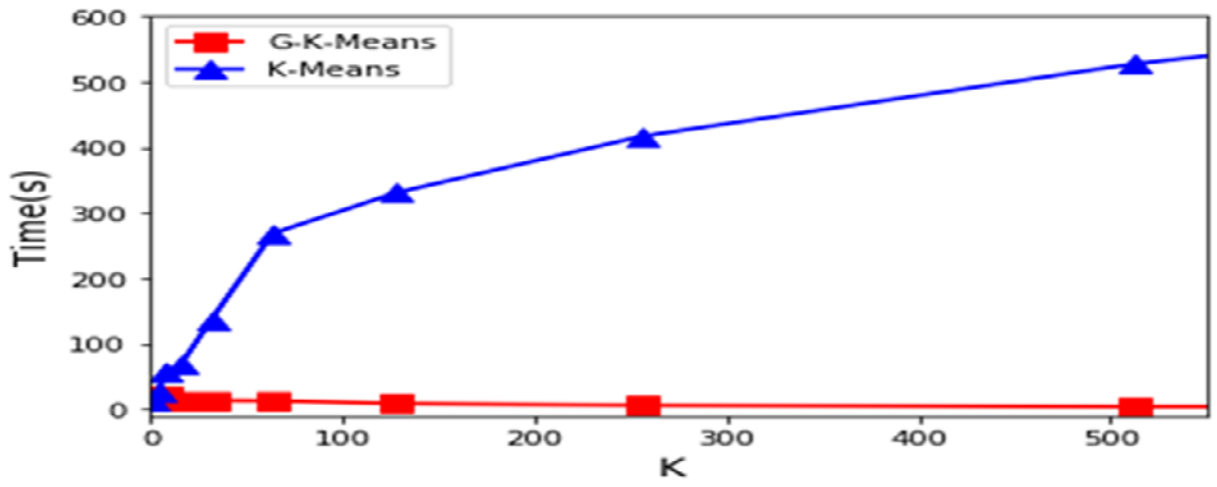

Figure 2. Clustering time comparing between G-K-Means&K-Means when N=100000,D=100-K-Means algorithm flow.

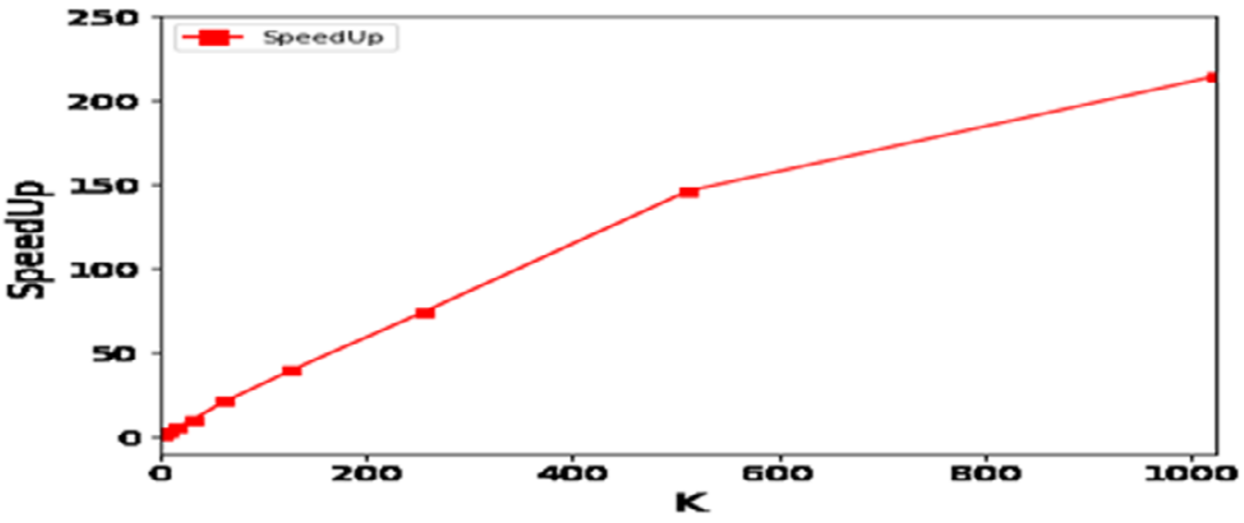

Figure 3. Algorithm Speedup when N=100000, D=100-K-Means.

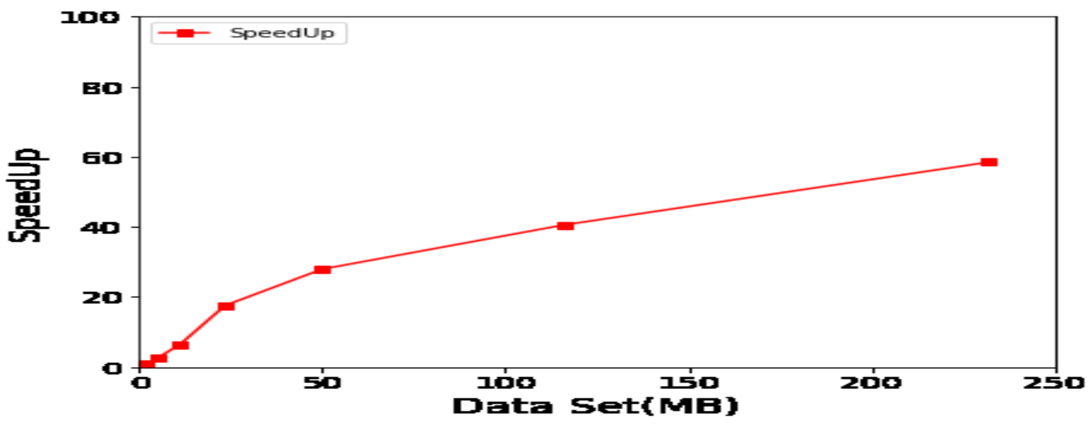

Figure 4. Algorithm Speedup for different dataset when k=128.

Fig. 4 depicts the clustering time comparison between the G-K-Means algorithm and the K-Means algorithm over a variety of data sets when k is 128. It can be seen that the acceleration effect of the G-K-Means algorithm on the smaller data set is not ideal, but with the increase of the data set, the clustering time of the G-K-Means algorithm is faster than that of the K-Means algorithm. When the set comes to 232MB, the acceleration ratio reached 58.47x.

The experimental results show that the G-K-Means algorithm, proposed in this paper, converts the process of multiple iterations and the updating process of k cluster centers to the GPU to improve the convergence speed of the algorithm. In particular, the greater the k value, the more obvious the effect of algorithm speedup. To illustrate the scalability of the algorithm, the experiment is carried out on a variety of data sets with different sizes. The results show that the speedup effect of G-K-Means parallel algorithm becomes more and more obvious as the data set scale increases.

5 Conclusion

In this paper, we deeply studied the parallelization of K-Means clustering algorithm of data mining. The experimental results show that GPU-based parallelization algorithm has greatly improved the efficiency when compares to the CPU-based serial algorithm. Especially, when data increase, the effect of acceleration becomes more obvious.

Acknowledge

This work was supported by Collaborative Innovation Center of Wireless Communications Technology and the National Natural Science Foundation of China (Grant No. 6157060279).

References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When reference

l footnote at the bottom of the column in which it was cite

1. Radiner L R.A tutorial on hidden Mardov model and selected applications in speech recognition[J].Proceedings of the IEEE,1989,77(2):257-286.
2. Rabiner L R,Juang B H.An Introduction to hidden Mardov models[J].IEEE ASSP Magazine,1986,3(1):4-16.
3. Dean J, Gheawat S. MapReduce: Simplified data processing on large clusters[J]. Communications of the ACM-50th Anniversary Issue: 1958-2008, 2008, 51 (1): 107-113.
4. Spark: Cluster Computing with Working Sets. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Technical Report No.UCB/EECS- 2010 -53 . 2010.
5. Han Jiawei, Kamber Micheline. Data Mining: Concepts and Techniques [M]. Second Edition. MingFan, Xiaofeng Meng translatrion. Beijing: Machinery Industry Press, 2007: p1-449.
6. M. Zechner and M. Granitzer, Accelerating K-Means on the graphics processor via CUDA, apr.2009, pp. 7-15.
7. Anguo Ma. Research on Key Technologies of High Performance GPGPU Architecture. National University of Defense Science and Technology. 2011:03-01.