Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fast density-based clustering through dataset partition using graphics processing units[☆,☆☆]

Woong-Kee Loh[a], Hwanjo Yu[b,*]

[a] Department of Software, Gachon University, Republic of Korea
[b] Department of Computer Science & Engineering, POSTECH, Republic of Korea

A R T I C L E I N F O

Article history:
Received 26 July 2013
Received in revised form 26 August 2014
Accepted 6 October 2014
Available online 19 October 2014

Keywords:
Density-based clustering
Graphics processing unit
Massively parallel algorithm
Dataset partition
On-chip shared memory

A B S T R A C T

Graphics processing units (GPUs) have been utilized to improve the processing speed of many conventional data mining algorithms. DBSCAN, a popular clustering algorithm that has been often used in practice, was extended to execute on a GPU. However, existing GPU-based DBSCAN extensions still have impediments in that the distances from all objects need to be repeatedly computed to find the neighbor objects and the objects and intermediate clustering results are stored in costly off-chip memory of the GPU. This paper proposes *CudaSCAN*, a novel algorithm that improves the efficiency of DBSCAN by making better use of the GPU. CudaSCAN consists of three phases: (1) partitioning the entire dataset into sub-regions of size of an integer multiple of the on-chip shared memory size in the GPU; (2) local clustering within sub-regions in parallel; and (3) merging the local clustering results. CudaSCAN allows an overlap between sub-regions to ensure independent, parallel local clustering in each sub-region, which in turn enables for objects and/or intermediate results to be stored in on-chip shared memory that has an access cost a few hundred times cheaper than that of off-chip global memory. The independence also enables for merging to be parallelized. This paper proves the correctness of CudaSCAN, and according to our extensive experiments, CudaSCAN outperforms CUDA-DClust, a previous GPU-based DBSCAN extension, by up to 163.6 times.

1. Introduction

A graphics processing unit (GPU) is a specialized processor that receives display data from the CPU, creates 30–60 frames per second, and then outputs the result to the computer display. It is usually a chip that is independent of the CPU, and it has a dedicated memory. In recent years, the GPU has been considered to be a high-performance vector processor that consists of a number of simple cores which conduct elementary operations like arithmetic and logic units (ALUs). Each parallel core produces output images for different portions in the same frame. Due to rapid improvements in modern GPU technologies, various attempts have been made to apply GPUs not only to advanced display but also to general problems or applications. The GPU is used as a massively parallel computer that uses a number of cores in which the instructions for graphics and

☆ This research was supported by the Basic Science Research Program through the National Research Foundation (NRF) funded by the Ministry of Education, Science and Technology (MEST) (No. 2010-0025001).
☆☆ This work was supported by Mid-career Researcher Program through NRF Grant funded by the MEST (No. NRF-2013R1A2A2A01067425).
* Corresponding author.
E-mail addresses: wkloh2@gachon.ac.kr (W.-K. Loh), hwanjoyu@postech.ac.kr (H. Yu).

http://dx.doi.org/10.1016/j.ins.2014.10.023
0020-0255/© 2014 Elsevier Inc. All rights reserved.

general applications can be executed. Recent studies have introduced various techniques to improve the processing speed of conventional data mining algorithms such as Apriori, *k*-means, DBSCAN, and support vector machine (SVM) using a GPU [1,4,6,9,16,24,27]. The GPU-accelerated data mining algorithms have achieved speed-up of up to two orders of magnitude better than the previous sequential ones. As the number of mining objects keeps increasing dramatically, faster GPU algorithms are also gaining prominence.

Clustering is a partitioning task that divides objects in a dataset into several groups (or clusters) without any prior knowledge, each cluster comprising objects with similar features. Clustering is a popular data mining task, and it has many applications in various domains, including physics, biology, social sciences, and marketing. Specifically, density-based clustering has been popularly used in practice; it creates clusters of objects in dense areas and divides the clusters in sparse areas. It has the following advantages over other clustering algorithms [12,13,19]. First, density-based clustering detects clusters of any shape, unlike most other methods that find clusters with convex shapes. Second, it efficiently filters out noises. Third, it does not require a pre-defined number of clusters, unlike *k*-means [12]. Fourth, its clustering results are not sensitive to the order of objects in the dataset. Fifth, it can handle cases where only the distances (differences) between the objects are given instead of their actual positions.

Representative density-based clustering algorithms are DBSCAN [8], OPTICS [2], and DENCLUE [15]. DBSCAN has been adopted for a few real-world applications that deal with very large datasets. Edla and Jana [7] used DBSCAN to cluster genes with similar expression levels, and each gene expression level was measured by the microarray. Clustering the enormous genes helps discover gene functions and regulatory mechanisms [7]. He et al. [13] analyzed 1.9 billion GPS location records obtained from about 6000 taxis for two years in Shanghai. They used DBSCAN to identify hot regions within the city and to distinguish erroneous GPS records. The analysis is helpful in designing city traffic policies and in detecting drivers with faulty behavior.

Various methods have been developed to improve the efficiency of the clustering algorithms [1,4,5,14,21,26]. In particular, CUDA-DClust [4] used the GPU to improve the efficiency of DBSCAN by up to 15 times relative to the sequential, CPU-based method. CUDA-DClust* was also proposed [4] to further improve the speed of CUDA-DClust by up to 11.9 times using a simple index structure, even though its performance should be highly dependent on the object distribution in the datasets. CUDA-DClust creates multiple sub-clusters simultaneously in order to maximize the utility of GPU cores. However, it needs to compute the distances among all objects in the dataset to find the neighbor objects, and the objects and computation results are stored in costly off-chip memory of the GPU. CUDA-DClust* greatly reduces the number of distance computations using an index, but it needs to first construct the index from the dataset before clustering can commence, and it does not effectively utilize the large quantities of GPU cores while traversing the index. It also still has to store the objects and the intermediate clustering results in the costly off-chip memory.

In this paper, we propose *CudaSCAN* (*CUDA-based DBSCAN*), a novel algorithm that improves the efficiency of DBSCAN using the GPU. CudaSCAN first divides the entire dataset into sub-regions with sizes that are not larger than pre-specified *S*. Then, local clustering within the sub-regions is performed in parallel. Clustering within each sub-region substantially reduces the computational cost, as the distances among the objects need to be computed only within the sub-region whereas DBSCAN needs to compute the distances among all the objects in the entire dataset. Finally, the local clustering results are merged in order to construct the final clustering result. We prove that the final clustering result is equal to that of DBSCAN. CudaSCAN further improves the execution speed by storing the objects and/or the intermediate local clustering results in the fast on-chip shared memory in the GPU.

Our extensive experiments reveal that CudaSCAN outperformed CUDA-DClust and CUDA-DClust*. CudaSCAN produces the clustering results by up to 163.6 times faster than CUDA-DClust and 2.83 times faster than CUDA-DClust*. CudaSCAN can be easily extended to parallel programming environments using multicore CPUs and distributed computing environments using the MapReduce framework.

This paper is organized as follows. In Section 2, we briefly explain the structure of Nvidia GPUs. We present existing density-based clustering methods along with their strengths and weaknesses in Section 3. In Section 4, we give a detailed account of the CudaSCAN algorithm, and we present the experimental results for CudaSCAN in Section 5. Finally, we conclude our study in Section 6.

2. Graphics processing units

Nvidia announced the 8800 series, codenamed G80, in 2006, and this series had a fundamentally different architecture from that of previous GPUs. Since then, Nvidia has released more sophisticated architectures such as Tesla, Fermi, and Kepler [17]. Fig. 1 shows an overview of Nvidia's GPU architecture. A GPU chip has several *multi-processors* (*MPs*),[1] and each MP consists of many *stream processors* (*SPs*).[2] An SP is an execution unit that has a simple structure, like an ALU in a CPU. Each MP performs a mutually independent task, and every SP contained in an MP executes the same operation for different data. Since modern GPUs consist of hundreds or thousands of SPs, a GPU can be treated as a massively parallel computer.

Nvidia's GPU has a complex memory structure to maximize the processing speed of applications. As Fig. 1 shows, each SP has its own private register that stores variables and temporal data for executing instructions on the SP. An SP cannot access

[1] An MP is also called a *streaming multi-processor* (*SM*).

[2] An SP is also called a *core*, but it has much simpler structure than CPU cores.