IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-11, Nov 2018 E-ISSN: 2347-2693

A Review on Various Nearest Neighbor Searching Algorithms Using Graphical Processing Units

Sneha Jacob, Anuj Mohamed

School of Computer Sciences, Mahatma Gandhi University, Kerala, India

Available online at: www.ijcseonline.org

Accepted: 24/Nov/2018, Published: 30/Nov/2018

***Abstract*—**The demand for Graphical Processing Units or GPUs, gained a tremendous hike during the past few years as a result of its migration from processing and representation of mere high dimensional graphical patterns to a heterogeneous high performance computing capability. The future generation data science requirements like Big Data Analysis and Deep Learning increased the popularity of GPUs to a wide extend. Graphical Processing Units or GPUs are well suited for parallel processing which enables visualization of vast amount of real time processed data in a more significant manner than CPU. From processing mere graphical algorithms, GPU has gone through numerous advancements in the past few decades. They can be used to improve the performance and efficiency of any algorithm nowadays. The expenditure of installation and use of GPUs have come down to a great extent from the initial huge amount. Data classification tasks like kNN classification can be done more efficiently and cost effectively by applying parallelism using GPU. kNN algorithms are the most popular data classification algorithm, because of its simplicity, high accuracy and versatility. This paper studies four major kNN algorithms developed for GPU processing and compares the techniques and methodologies used in them.

***Keywords*—** GPU, BF CUDA, CUBLAS, CUKNN

I. INTRODUCTION

Classification is the process of categorizing a newly arrived data to its original class label, by studying the characteristics of previously trained data. The classification models are categorized into two- Eager Learning methods and Lazy Learning methods. The different eager learning classification methods include Classification by Decision Tree Induction, Bayesian Classification, Rule Based Classification, Classification by Back propagation, Support Vector Machines (SVM), Associative Classification etc. K Nearest Neighbor classifier and Case-based reasoning classifiers are examples of Lazy Learners. Even though Lazy Learners are more expensive and require high storage space, they are well suited for parallel implementation. kNN (k Nearest Neighbour) classification [1] is the simplest and the most popular technique in the Lazy Learner classification methods. They are well suited to model high dimensional data.

Over the past years, many authors have proposed a number of optimal algorithms for kNN classification. For example dW-ABC-kNN(distance Weighted kNN using Artificial Bee Colony) algorithm [2], kTree method [3], coefficient Weighted kNN classifier, Residual Weighted kNN classifier[4] etc.

The main objective of this paper is a thorough study of the different algorithms for implementing kNN classification, in Graphical Processing Units. The four different Nearest Neighbour Searching algorithms taken for this study are BF CUDA (Brute Force Compute Unified Device Architecture), GPU kNN, CUBLAS (CUDA Basic Linear Algebra Subprograms) and CUKNN (CUDA kNN). These algorithms are compared with regard to their speed, accuracy and the sorting techniques used.

This paper is organized into 5 different sections. Section I gives an introduction about the importance of classification algorithms in Graphical Processing Units. Section II illustrates a brief description about GPUs and CUDA programming. Section III presents related works in this area. Section IV depicts a simplified GPU memory architecture and Section V illustrates the observations of the study. Conclusion and future research directions are specified in section VI.

II. GRAPHICAL PROCESSING UNITS AND CUDA PROGRAMMING

When the same set of code is to be applied on a large amount of data elements, Graphical Processing Units (GPU) provide an efficient platform for the same through parallelism. Hence the distance calculation and sorting steps which are independent, in kNN classification, can be done parallel using GPUs. GPUs achieve high degree of parallelism by dedicating most of their transistors for data processing rather than control and data flow management. Graphical

Processing Systems provide assistance for parallel processing which in turn helps to achieve high degree of speed acceleration in High Performance Computers. With its multiple processors, working in parallel, GPU is highly useful for both graphics and non-graphics processing [5]. Different companies developing GPUs include AMD (Advanced Micro Devices), ARM Holdings, Broadcom Limited, Imagination Technologies, Intel, Matrox, NVIDIA, Qualcomm etc.

CUDA is a general purpose computing architecture that helps to fully utilize the parallel processing capability of GPU. CUDA stands for Compute Unified Device Architecture. CUDA was introduced by NVIDIA in 2006. It supports various high level programming languages like C, C++, FORTRAN, Open CL, Direct Compute etc [6].

III. RELATED WORKS

Many researches are being done in the classical kNN search to empower it to do advanced classifications like Multi Label (ML-kNN) learning in which each object instance will be associated with more than one class label [7]. Applying parallelism in kNN using GPU, will increase the performance of algorithms to several multiples of the original one.

Due to high cost of GPU purchase, installation and maintenance and limited languages in GPU computing, many manufacturing companies even suspended their production and maintenance supports of GPUs. Thus there came a gap in the research and use of High Performance Computers for few years.

Advances in Cloud computing, Big data analysis and Internet of Things again demanded the capacities and capabilities of HPCs for their efficient processing using GPUs. Studies have proved that the use of GPUs helps to solve many high end computing problems many times faster than using conventional CPUs. This became handier by the development of CUDA programming language. For instance, the project named GAP utilizes the high end computing power of GPU in various real time applications [8].

IV. GPU CUDA MEMORY ARCHITECTURE

GPUs were initially developed to accelerate floating-point calculations[9]. Later, programming languages like CUDA by NVIDIA has increased the usability of GPUs by making it suitable for heterogeneous processing of numerous real time data within a short period of time.

The major challenge in utilizing the high end features of GPU is the thorough understanding of the GPU architecture. Deciding the number of threads and blocks is one of the most important tasks that affect the efficient utilization of GPU. The GPU memory is being divided into Grids, Blocks and Threads. Each Grid contains a number of blocks and each block consists of a number of Threads. Kernel is the function that invokes GPU. 'Host' represents the local system we are using for programming and 'Device' refers to the GPU.

Several novel GPU memory architectures have been proposed by various authors [10] to overcome the difficulty in memory management of GPU. Many advanced algorithms like Spatial Preprocessing (SPP) have been developed to increase the efficient memory utilization of GPUs [11].

The major difference between CPU and GPU is that, GPU dedicates most of its transistors for data processing rather than flow control and temporary storage. A combination of GPU and CPU can present the most efficient and cost effective high performance computing.

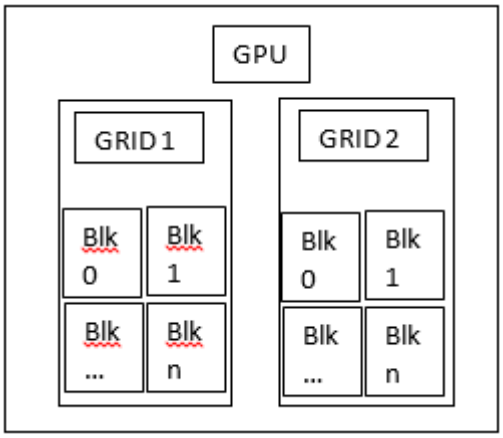

Figure 1. GPU Memory Architecture

V. STUDY AND OBSERVATIONS

BF-CUDA is a method for k Nearest Neighbour Search using Brute Force method with CUDA in GPU. A comparison of the performance of this new algorithm with three other algorithms is also presented. They proved that the BF CUDA algorithm is faster than the other algorithms named BF-Matlab, BF-C and ANN (Approximated Nearest Neighbor) C++ [12]. The Datasets used in this experiment were a synthetic dataset and a real dataset. The Brute Force Method employs four steps to find k Nearest Neighbours among 'm' reference and 'n' query points in a 'd' dimensional data. Let $Q\{q_1,q_2,\ldots,q_n\}$ be the set of query points and $R\{r_1,r_2\ldots,r_m\}$

- Find out distance between the query point (q_i) with each reference point (r_j) in the set R.
- Sort these distance values.
- Select 'k' smallest distances from this sorted list.
- Repeat all the above steps for each element in the set Q.

The sorting technique used here is the insertion sort. High complexity is the major disadvantage of Brute Force approach. In this paper, they have compared both "Comb Sort" and "Insertion Sort" implementations in CUDA. The result states that the computation time for "Comb Sort" remained constant whereas that of "Insertion Sort" increased linearly.

Kuang, Quansheng, and Lei Zhao [13] proposed a GPU based kNN algorithm in which they used a data segmentation method. They utilized the thread model of the GPU memory hierarchy. Euclidean distance calculation method was used for finding out the distance between query and reference points. Instead of computing the final square root, squared distances are used in this system. This will reduce the computing time. They used a batch loading strategy while reading the global memory. The sorting technique used here is a CUDA based Radix Sort. After the analysis, it had been found that, the distance calculation phase can be highly paralleled and can reach a high speed up ratio in GPU implementation. The sorting step can also be accelerated and the rest of the steps are not much time consuming. For finding out the classification labels, this method used a simple statistical election, depending upon the frequency of classification label. This step is accomplished using CPU instead of GPU to avoid complexity. They have used the Adult dataset from UCI Machine Learning Repository. In the performance analysis, they found that GPU algorithm is 32.61 times faster than CPU algorithm and 14.98 times faster than ANN BF method while using a1a dataset. In a2a dataset, GPU algorithm reached a speedup of 34.91 times compared with CPU algorithm and 15.36 times with ANN-Brute method.

Garcia and Debreuve [14], proposed two fast GPU-based implementations of the Brute Force KNN search algorithm using CUDA and CUBLAS APIs. They proved that CUDA and CUBLAS implementations are up to 64 times and 189 times faster on synthetic data than ANN C++ library and up to 25 times and 62 times faster on high dimensional SIFT matching. Search method used in this paper is the Insertion sort. They have used the CUDA implementation in BLAS library and named it as CUBLAS. This new method is 189 times faster than ANN and up to 4 times faster than their previous approach. The distance calculation method they used in their previous method had been modified and rewritten in this paper. A modified insertion sort is used for sorting. In the implementation stage, they had reformulated the distance computation using CUBLAS library. They state that for Synthetic datasets, CUBLAS is 189 times faster than ANN and 4 times faster than CUDA. In case of HD SIFT matching, CUBLAS is 62 times faster than ANN and 2.5 times faster than CUDA.

A CUDA based parallel implementation of KNN named CUKNN is being proposed by Liang, Liu, Wang and Jian [15]. They used a synthetic dataset and a real physical simulation dataset. Techniques like CUDA stream and Memory coalescing were utilized here. The sorting method applied is the Quicksort. The results show that CUKNN is 46.71 times faster than serial KNN on synthetic dataset and 42.49 times faster on physical simulation dataset. CUKNN achieved 21.81 times speed up over quicksort based KNN and 46.71 times faster than Insertion sort based KNN.

Table1 shows a comparison chart of the different algorithms executed in the GPU.

Table 1: A comparison of sorting techniques and speed of different kNN algorithms using GPU

Algorithm	Comparison of Algorithms			
	Dataset	*Sorting Technique*	*Speed*	*Reference*
BF CUDA	Synthetic	Insertion Sort	407 X than BF-MATLAB, 295 X than BF-C & 148 X than ANN C++	[12]
	Real			
GPU kNN	a1a	Radix Sort	32.61 X than CPU & 14.98 X than ANN-BF	[13]
	a2a		34.91X than CPU&15.36 X than ANN-BF	
CUBLAS	Synthetic	Insertion Sort	189 X than ANN & 4 X than CUDA	[14]
	HD SIFT Matching		62 X than ANN & 2.5 X than CUDA	
CUKNN	Synthetic	Quicksort	46.71 X than Serial KNN	[15]
	Real Physical Simulation		42.49 X than Serial KNN	

VI. CONCLUSION AND FUTURE SCOPE

In this paper, a comparison study of various GPU based algorithms for kNN classification is presented. The comparison of results presented by the works in [12], [13], [14] and [15] shows that BRUTE FORCE CUDA (BF CUDA) is the most efficient and faster algorithm for GPU based data classification. CUDA programming has reduced much of the complexities in GPU programing with its scalability and extensibility, which in turn increased the

scope of GPU usage to a great extent. Further studies in this area will surely benefit future real time data processing especially in the field of Big data analysis and deep learning as well.

ACKNOWLEDGEMENT

The authors acknowledge the support extended by DST.PURSE (Phase II), Government of India.

REFERENCES

[1] Han, Jiawei, Jian Pei, and Micheline Kamber. *"Data mining: concepts and techniques"*. Elsevier, 2011.

[2] Yigit, Halil. *"A weighting approach for KNN classifier"*. Electronics, Computer and Computation (ICECCO), International Conference on. IEEE, 2013.

[3] Zhang, Shichao, et al. *"Efficient knn classification with different numbers of nearest neighbors"*. IEEE transactions on neural networks and learning systems 29.5, 2018.

[4] Ma, Hongxing, Jianping Gou, Xili Wang, Jia Ke, and Shaoning Zeng. *"Sparse Coefficient-Based k-Nearest Neighbor Classification."* IEEE Access 5: 16618-16634, 2017.

[5] Buck, Ian. *"Gpu computing: Programming a massively parallel processor."* International Symposium on IEEE, 2007.

[6] NVIDIA CUDA C Programming Guide Version 4.2, 2012.

[7] Zhang, Min-Ling, and Zhi-Hua Zhou. *"ML-KNN: A lazy learning approach to multi-label learning."* Pattern recognition , 2007.

[8] Bauce, M., et al. *"The GAP project-GPU for real-time applications in high energy physics and medical imaging."* 19th IEEE-NPSS Real Time Conference. IEEE, 2014.

[9] McClanahan, Chris. *"History and evolution of gpu architecture."* A Survey Paper, 2010.

[10] Kim, Youngsok, Jaewon Lee, Donggyu Kim, and Jangwoo Kim. *"ScaleGPU: GPU architecture for memory-unaware GPU programming."* IEEE Computer Architecture Letters, 2014.

[11] Delgado, Jaime, Gabriel Martín, Javier Plaza, Luis-Ignacio Jiménez, and Antonio Plaza. *"On the optimization of memory access to increase the performance of spatial preprocessing techniques on graphics processing units."* IEEE International Geoscience and Remote Sensing Symposium, 2016.

[12] Garcia, Vincent, Eric Debreuve, and Michel Barlaud. *"Fast k nearest neighbor search using GPU."* at arXiv preprint arXiv:0804.1448 , 2008.

[13] Kuang, Quansheng, and Lei Zhao. *"A practical GPU based kNN algorithm."* Proceedings. The International Symposium on Computer Science and Computational Technology (ISCSCI 2009). Academy Publisher, 2009.

[14] Garcia, Vincent, et al. *"K-nearest neighbor search: Fast GPU-based implementations and application to high-dimensional feature matching."* Image Processing (ICIP), 17th IEEE International Conference, 2010.

[15] Liang, Shenshen, et al. *"Design and evaluation of a parallel k-nearest neighbor algorithm on CUDA-enabled GPU."* Web Society (SWS), IEEE 2nd Symposium, 2010.

