# *ACCELERATION K-MEANS CLUSTERING WITH THE COMPUTE UNIFIED DEVICE ARCHITECTURE (CUDA)* (Case Study at SMA ISLAM HIDAYATULLAH)

Wahyu Cepta Gusta
Faculty of Computer Science - Dian Nuswantoro University
Jl Nakula I No. 5-11 Semarang 50131, Indonesia
cyberblackwg@gmail.com

***Abstract*— K-means algorithm is one of the unsupervised learning clustering algorithms that can be used to solve some problems such as marketing strategies, determining criteria beasiswaa receiver, classifying documents until the vehicle identification number. Computational processing time of the sequential k-means algorithm is relatively high at the time of execution. This is because the computational processes executed according to a certain order before the next computation.**

**This paper proposed an parallel to accelerate the execution time of k-means algorithm which is utilizing GPGPU that supports high speed parallel computing. It will be implemented with CUDA from NVIDIA. The experiment results indicate that parallel K-Means achieve the acceleration on GPGPU.**

*Keywords- k-means, parallel computing, GPGPU, CUDA*

## I. Introduction

Formation of clustering is one of the data mining methods that are unsupervised learning. Clustering is a tool for data analysis, which solve the problem of classification. Clustering of data mining is used to discover patterns of distribution in the dataset that can be used for the data analysis process. The similarity of the object obtained from the proximity of the values of the object attribute that describes the data, while the data object represented partially a point in multi-dimensional space [1].

There are two methods of clustering, namely hierarchical clustering and partitioning. Hierarchical clustering method consists of complete linkage clustering, single linkage clustering, average linkage clustering and centroid linkage clustering , while the partitioning method consists of k-means and fuzzy k-means. K-Means method is the most simple and common [2]. In 1967 Mac Queen developed a K-Means algorithm [3] which is one of the clustering methods are partitioning that separates data into different groups [4]. By iteratively partitioning , K - Means is able to minimize the average distance of each data to his cluster .

K - means clustering algorithm can be used to solve some problems such as marketing strategy [5], the determination of the criteria beasiswaa receiver [6], classifying documents [7] to the identification plate of the vehicle [8]. The advantages of k- means clustering algorithm capable of classifying such objects efficiently thus improving the accuracy of clustering process [9]. However, the computational time required in the process of clustering is still quite high [10]. This is because the computation is performed in a sequential manner that is sequential and gradual , there is a queue process . each computing process must be completed before the next computations , thus requiring a high execution time [10]. Execution time is also influenced by the amount of data processed .

Another Disisis, utilization of the graphics card into the era of General Purpose Graphical Processing Units ( GPGPU ) , namely the use of graphics cards to work umum.GPU computationally capable of parallel processing that can be used to improve the performance of computing.

Based on these facts, in this study the proposed parallelization of the k- means clustering to obtain acceleration algorithm execution time.

On the other hand, the use of the graphics card into the era of General Purpose Graphical Processing Units (GPGPU), namely the use of graphics cards to perform computing umum.GPU able to perform parallel processing that can be utilized to improve the performance of computing

The aim of this paper is to accelerate the execution time of k-means algorithm with a parallel. It will be implemented on GPGPU technology with a software platform which is CUDA from NVIDIA.

This paper is organized as follows. Section 1 summarizes the motivation. Section 2 outlines the preliminary concepts of k-means. Section 3 introduces the GPGPU technology. Section 4 explains the CUDA environment. The experimental result will be discussed in the section 5. The final section is a conclusion of the paper.

## II. K-MEANS ALGORITHM ON THE GPU

The main idea of GPU-based *k*-means is that data-parallel, compute-intensive portions of traditional *k*- means can be off-loaded from the host to the device to improve performance. More precisely, data objects assignment and *k* centroids recalculation executed many times, but independently on different data, can be isolated into two functions consisted of massive threads, parallel executing on the device. Actually, each function is compiled to the instruction set of the device and the target program, called a kernel, is downloaded to the device.

GPU-based *k*-means has three fundamental issues to be addressed, though the SIMD processors are accomplished in parallel computing. First, flow control and data caching of the device are weak for its more transistors are devoted to compute unit. Second, compared with the data transfer rate between the CPU and CPU's cache, the data transfer rate between GPU and GPU's memory (global memory) is much slower, then only appropriate size of block and grid is capable of winning device's power. In the end, the transfer time between the CPU's memory and GPU's memory is extra cost relative to traditional *k*-means on CPU. Performance enhancement can be obtained, as long as duty assignment for the host and the device, data storage, and parallel computing mode are reasonably designed and implemented.

In task assignment, the host is responsible for placing *k* objects into the space represented by the objects that are being clustered and rearranging all data objects and controlling iteration process, while the device for data-parallel intensive computing. In data storage, all data objects and centriods are stored as dynamic arrays on the device. We put all parameters in *global memory* as both other *constant memory* and *texture memory* are read-only and respective, which are insufficient to data. Another remarkable point is that the bandwidth between the device and the device memory is much higher than the bandwidth between the device memory and the host memory. In our approach, the cluster labels transfer between the host and the device is very small. In parallel computing mode, kernel is assigned enough computing routine and massive threads may reduce the global memory latency. This frame of GPU- based *k*-means is designed by the architectures of CPUs and GPUs, which is adapted to not only CUDA.

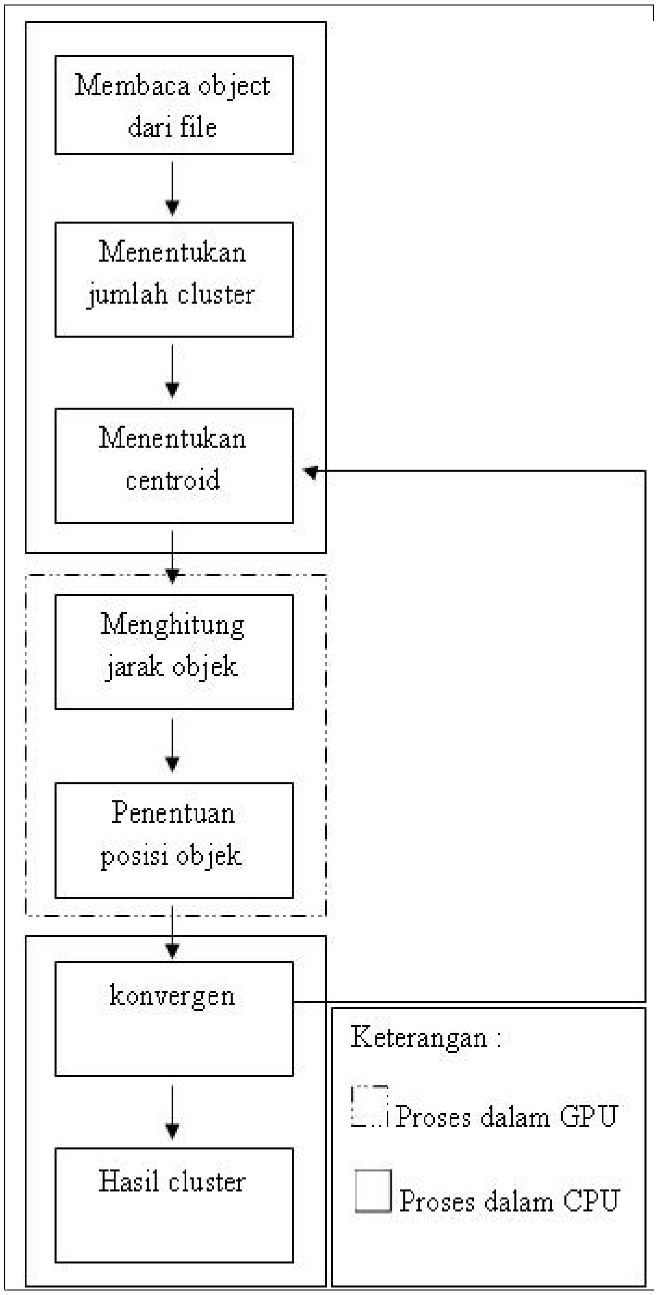

**Fig 1.** Frame GPU k-means

## III. GPGPU TECHNOLOGY

GPU's are probably today most powerfull computational hardware for the dollar. The rapid increase in the performance of graphics hardware, coupled with recent improvement in its programmability, have made graphics hardware a compelling platform for computationaly demanding task in wide variety of application domains. Alot of researches have been presented in recent years for general-purpose computing, an effort known collectively as GPGPU [11].

The performance of the GPU in computing general-purpose algorithms depends heavily on how the algorithms are arranged so as to exploit the parallel data processing power of the GPU. In our study, we have used Nvidia's GeForce GTX 650. In graphics processing, the GPU receives commands for display in the form of vertices and connectivity details from the CPU. Today’s GPUs have very high memory bandwidth and parallel internal processors, which are capable to process streams of incoming data. These processing is performed in either the vertex or the fragment processor in the GPU using specific shader programs [12]. Computations in the GPU processors are data independent, which means that the processing that occurs in each processor, is independent of the data used by other processors. Currently, there is lot of research focus in the arena of implementing general-purpose computations in the GPU (GPGPU) to leverage on speed w.r.t unit cost function. Within the GPU, the fragment processors support parallel texture operations and are able to process floating-point vectors. Implementations of GPGPU are challenging and mostly utilize the texture processing abilities of the fragment processor of the GPU.

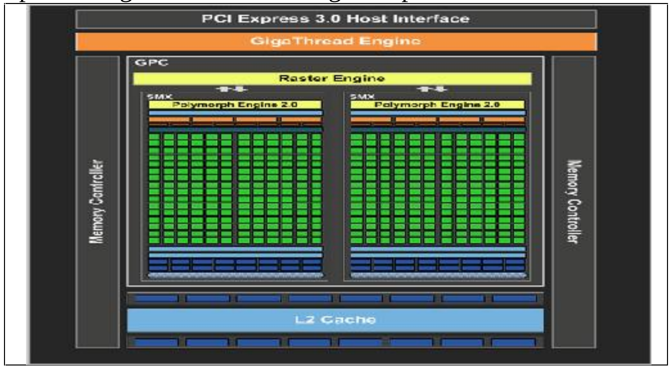

**Fig 2.** Arsitektur GTX 650

## IV. CUDA TECHNOLOGY

To be able to maintain and take advantage of the gpu which is general purpose computation, there are some kind interfaces involves Open Computing Language(Open CL), Direct Compute, and also CUDA. All those interfaces have same characteristic although in the other side it has different programming interfaces. Open CL introduces by the Khronos Group which is a parallel programming used in CPU, GPU, Digital Signal Processors (DSP), and some other processor’s types [11]. Direct Compute is an API which is standart GPU computing platform for windows, namely Windows 7 and Windows Vista, it means this API can not be used in different platform, limited library, examples, and bindings [11]. While CUDA introduces by NVIDIA as a new parallel programming model, namely the general purpose parallel computing architecture for NVIDIA’s graphic cards or GPU [12] [13]. Refers to Jayshree et.al, CUDA is better than Open CL and Direct Compute based on some considerations involve a flexibility on various platforms, availability of documentation and examples, fully supported by NVIDIA, an availability inbuilt libraries, debugging with advanced tools, support the existing construct of C/ C++ [14].

Moreover, CUDA programming model supports merging operations execution on the CPU and GPU. CUDA consists of both hardware and a software model allowing the execution of computations on modern NVIDIA GPUs in a data-parallel model.

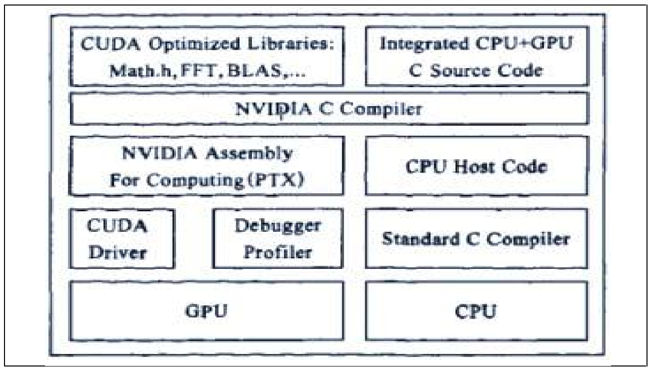
CUDA Optimized Libraries: Math.h, FFT, BLAS,...
Integrated CPU+GPU C Source Code
NVIDIA C Compiler
NVIDIA Assembly For Computing (PTX)
CPU Host Code
CUDA Driver
Debugger Profiler
Standard C Compiler
GPU
CPU

**Fig 3.** *CUDA Architecture*

“NVIDIA CUDA is a SDK (Software Development Kit) released by graphic hardware manufacture NVIDIA with the purpose of making it possible for programmers to accelerate general-purpose computations by using the computational resources available to modern GPUs”. CUDA programming is an interface to be able to access the GPU parallel computing capabilities by writing code that runs directly on the device. In term of CUDA, the GPU is called device, whereas the CPU is called host. The CUDA architecture is shown in Figure 3 which comprises from several parts [23]. A number of optimized libraries for CUDA provided by NVIDIA, such as FFT, Blas, math.h, and so on. The main think of CUDA architecture is NVIDIA C compiler (NVCC). As mentioned earlier that the CUDA program is a mix code of GPU and CPU which is isolated the code of GPU and CPU by NVCC compiler. CPU will compile the CPU code. And for the GPU code, it is compiled into GPU's computing assembly code-PTX. GPU code that runs is supported by the CUDA driver.

The emerging of CUDA as a programming model is an extension of the C language written especially for the NVIDIA graphic card. In the GPGPU environment, the basic programming language used was on C or C++ based [15]. In the other hand, object oriented programming model are widely increase, it is natural to explore how CUDA-like capabilities can be made accessible to those programmers as well.

## V. EXPERIMENTAL RESULT

In the experimental result, it is applying parallel kmeans on datasets. The experiment is implement on Intel Dual Core with NVIDIA GTX 650 and with 2GB main memory.

**Table 1.** Execution Time of Sequensial K-means CPU

| Data (N) | Waktu Eksekusi (seconds) | | | | | |
|---|---|---|---|---|---|---|
| | Uji Coba | | | | | Rata-rata |
| | 1 | 2 | 3 | 4 | 5 | |
| 10 | 0.016 | 0.062 | 0.016 | 0.047 | 0.047 | 0.0376 |
| 100 | 0.025 | 0.031 | 0.016 | 0.125 | 0.094 | 0.0582 |
| 1,000 | 0.296 | 0.031 | 0.094 | 0.047 | 0.031 | 0.0998 |
| 10,000 | 0.484 | 0.499 | 0.156 | 0.296 | 0.156 | 0.3182 |
| 100,000 | 1.763 | 1.404 | 1.451 | 1.591 | 1.482 | 1.5382 |

**Table 2.** Execution Time of Paralel K-means GPU

| Data (N) | Waktu Eksekusi (seconds) | | | | | |
|---|---|---|---|---|---|---|
| | Uji Coba | | | | | Rata-rata |
| | 1 | 2 | 3 | 4 | 5 | |
| 10 | 0.016 | 0.062 | 0.016 | 0.047 | 0.047 | 0.0376 |
| 100 | 0.025 | 0.031 | 0.016 | 0.125 | 0.094 | 0.0582 |
| 1,000 | 0.296 | 0.031 | 0.094 | 0.047 | 0.031 | 0.0998 |
| 10,000 | 0.484 | 0.499 | 0.156 | 0.296 | 0.156 | 0.3182 |
| 100,000 | 1.763 | 1.404 | 1.451 | 1.591 | 1.482 | 1.5382 |

## VI. Conclusion

In the paper, the concepts K-Means algorithm were discussed. Proposed framework of the parallel K-Means algorithm has been done within GPGPU technology. It is done on the GPU card that provided by NVIDIA, by using CUDA platform as a programming model. From the generated result, this framework is able to accelerate the execution time by parallelizing. The volume of the data to be computed effect on response time.

## VII. References

[1] J.-S. Chen, R. K. H. Ching, and Y.-S. Lin, "An extended study of the K-means algorithm for data clustering and its applications," *Journal of the Operational Research Society*, vol. Volume 55, 2004.

[2] B. Santosa, *Data Mining Teknik Pemanfaatan Data untuk Keperluan Bisnis*. Yogyakarta : Graha Ilmu, 2007.

[3] J. B. Macqueen, "Some Methods For Classification And Analysis of Multivariate Observations," *In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[4] Y. Agusta, "K-Means - Penerapan Permasalahan dan Metode Terkait," *Jurnal Sistem dan Informatika* , vol. 3, pp. 47-60, 2007.

[5] J. O. Ong, "Implementasi Algoritma K-Means Clustering Untuk Menentukan Startegi Marketing President University," *Jurnal Ilmiah Teknik Industri*, vol. 12, pp. 10-20, Jun. 2013.

[6] N. F. Hastuti, R. Saptono, and E. Suryani, "Pemanfaatan Metode K-Means Clustering Dalam Penentuan Penerima Beasiswa," 2013.

[7] M. Umran and T. F. Abidin, "Pengelompokkan Dokumen Menggunakan K-Means Dan Singular Value Decomposition : Studi Kasus Menggunakan Data Blog," *SeSINDO*, 2009.

[8] A. Unknown, I. Unknown, and F. Unknown, "Identifikasi Dengan Menggunakan Algoritma K Means Pada Plat Kendaraan," *Journal Poli Rekayasa*, vol. 6, 2010.

[9] K. Arai and A. R. Barakbah, "Hierarchical K-means: an algorithm for centroids initialization for K-means," Reports of the Faculty of Science and Engineering, 2007.

[10] P. Pacheco, *An Introduction to Parallel Programming*, 1st ed. Morgan Kaufmann, 2011.

[11] M. Zechner and M. Granitzer, "Accelerating K-Means on the Graphics Processor via CUDA," in *Intensive Applications and Services, 2009. INTENSIVE '09. First International Conference on*, Valencia, 2009, pp. 7-15.

[12] R. Xu and D. Wunsch, *Clustering (IEEE Press Series on Computational Intelligence)*. Wiley-IEEE Press, 2008.

[13] E. Wu and Y. Liu, "Emerging technology about GPGPU," in *APCCAS 2008. IEEE Asia Pacific Conference on*, Macao, 2008, pp. 618-622.

[14] A. J. van der Steen and J. Donggara, *Overview of High Performance Computers*. Kluwer Academic Publishers, 2001.

[15] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. 2006.

[16] J. B. Srivastava, R. K. Pandey, and J. Jain, "Implementation of Digital Signal Processing Algorithm in General Purpose Graphics Processing Unit (GPGPU)," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 4, Jun. 2013.

[17] J. Sirotkovic, H. Dujmic, and V. Papic, "K-Means Image Segmentation on Massively Parallel GPU Architecture," *MIPRO*, 2012.

[18] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010.

[19] M. Sadaaki and Y. Nakayama, "Algorithms of Hard C-Means Clustering UsingKernel Functions in Support Vector Machines," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. Vol. 7, p. 19–24, 2003.

[20] R. A. S. Putri and A. Suhendra, "Analisis Perbandingan Komputasi Sekuensial dan Komputasi Paralel GPU Memanfaatkan Teknologi NVIDIA Cuda Pada Aplikasi Pengurutan Bilangan Acak Menggunakan Algoritma QuickSort," *E-Journal Teknologi Industri*, 2012.

[21] S. Páll, B. Hess, and E. Lindahl, "Poster: 3D tixels: a highly efficient algorithm for gpu/cpu-acceleration of molecular dynamics on heterogeneous parallel architectures," in *Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion*, 2011, pp. 71-72.

[22] D. Padua, *Encyclopedia of Parallel Computing*. Springer, 2011.

[23] J. D. Owen, et al., "A survey of general-purpose computation on Graphics Hardware," in *Proceedings of Eurographics 2005*, 2005, pp. 21-51.

[24] NVIDIA. Developer NVIDIA Zone. [Online]. https://developer.nvidia.com/what-cuda

[25] A. Mahardito, A. Suhendra, and D. T. Hasta, "Optimizing Parallel Reduction In Cuda To Reach GPU Peak Performance," *Published Article Komputer*, 2010.

[26] J. Krüger and R. Westermann, "Linear Algebra Operators for GPU Implementation of Numerical Algorithms," in *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003*, 2003, pp. 908-916.

[27] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series)*. Morgan Kaufmann, 2010.

[28] T. Kim, S. Song, D. Choi, Y. Kwak, and B. Goo, "Sequence Data Indexing Method Exploiting the Parallel Processing Resources of GPGPU," in *The 3rd International Conference on Circuits, Control, Communication,*, 2013.

[29] A. Januarianto and A. Suhendra, "Analisis Perbandingan Komputasi Sekuensial dan Komputasi Paralel GPU Memanfaatkan Teknologi NVIDIA CUDA Pada Aplikasi Kompresi Citra Menggunakan Algoritma DCT 8X8," *E-Journal Teknologi Industri*, 2012.

[30] S. A. Horup, S. A. Juul, and H. H. Larsen, *The Art GPGPU*. 2011.

[31] J. Han and M. Kamber, *Data Mining Concepts and Techniques 2nd Edition*. San Francisco: Morgan Kaufmann, 2007.

[32] N. K. Govindaraju, B. Lloyd, W. Wang, M. Lin, and D. Manocha, "Fast computation of database operations using graphics processors," in *SIGGRAPH '05 ACM SIGGRAPH 2005 Courses*, 2005.

[33] f. Gebali, *Algorithms and Parallel Computing*. Wiley, 2011.

[34] R. Farber, *CUDA Application Design and Development*. Morgan Kaufmann, 2011.

[35] J. Fang, A. L. Varbanescu, and H. Sips, "A Comprehensive Performance Comparison of CUDA and OpenCL," in *Parallel Processing (ICPP), 2011 International Conference on*, 2011, pp. 216-225.

[36] Z. Fan, F. Qiu, A. Kaufman, and S. Yoakum-Stover, "GPU Cluster for High Performance Computing," in *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, 2004, p. 47.

[37] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs (Applications of GPU Computing Series)*. Morgan Kaufmann, 2012.

[38] D. Pettinger and G. Di Fatta, "Scalability of Efficient Parallel K-Means," in *E-Science Workshops, 2009 5th IEEE International Conference on*, Oxford, 2009, pp. 96-101.

[39] K. Rupp, F. Rudolf, and J. Weinbub, "ViennaCL - A High Level Linear Algebra Library for GPUs and Multi-Core CPUs," in *Proceedings of GPUScA*, 2010, pp. 51-56.