KACU: K-means with Hardware Centroid-Updating

Wei-Chuan Liu, Jiun-Long Huang[†] and Ming-Syan Chen

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC

[†]Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan, ROC

E-mail: weichuan@arbor.ee.ntu.edu.tw, jlhuang@csie.nctu.edu.tw, mschen@cc.ee.ntu.edu.tw

***Abstract*— In this paper, we propose a framework, KACU (standing for K-means with hArdware Centroid Updating), to enhance the speed of K-means clustering algorithm by integrating a hardware centroid updating mechanism into the procedure of continuous K-means algorithm. To facilitate performance measurement, KACU is implemented in a commercial Field Programmable Gate Array (abbreviated as FPGA) device. The experimental results show that KACU is able to achieve considerably higher performance.**

I. Introduction

Data mining is the process of extracting features, discovering patterns and clustering data from large volumes of raw data. For conducting such data archives, there is a growing need for rapid processing ability. In many algorithms of data mining, a great amount of computing power is spent on some specific routines with repeated computation. For these algorithms, dramatic speedups can often be obtained with specialized hardware. Although processing can also be accelerated by adopting a more sophisticated algorithm or heuristic ones, these algorithms are often too complicated to be implemented in hardware. Fortunately, it is sometimes possible to combine software and hardware designs in such a way that each complements the other, and the final implementation is able to achieve higher speedup than a hardware-only or software-only solution [8]. Utilizing the concept of HW/SW co-design is a trend to accelerate applicable algorithms. With the incremental development of reconfigurable hardware, and in particular the commercial FPGA [1][9], it creates a new scope for the design space, changes the view on algorithmic problem solving and has the advantage of being extremely powerful for many applications.

In this paper we design a specific hardware solution to accelerate the processing speed of K-means clustering algorithm. Clustering has been recognized as an important unsupervised learning problem which deals with finding a structure from a collection of unlabeled data. A loose definition of clustering is the process of grouping objects into clusters whose members are similar in some way. The most popular clustering algorithm is K-means due to its simplicity. In [4], the authors applied HW/SW co-processing technique on a hybrid processor for K-means algorithm. In that processor, a systolic process array was designed to speedup the mutual distance calculation loop which is the most time consuming operation in standard K-means algorithm. In this paper, we consider a variant of K-means algorithm, named continuous K-means [3], in which faster convergence is generally achieved when the centroids are updated on the fly. We apply systolic process array into our design and embed the updating scheme in the flow of pipeline. Therefore, the procedure of centroid updating becomes a number of stages in pipeline and can be accomplished as fast as possible to meet the data arrival rate.

The rest of this paper is organized as follows. The preliminaries of hardware enhanced mining are given in Section 2. The proposed hardware solution of K-means is described in Section 3. Experiment results are shown in Section 4. Finally, this paper concludes with Section 5.

II. Preliminaries

Hardware enhancement is sometimes thought as a brute force approach to speed up algorithms. However, implementing an algorithm in hardware is not a trivial task, and usually requires more effort than implementing in software. Furthermore, not all algorithms can be efficiently implemented in pure hardware, since implementing an algorithm in hardware includes a different set of design strategies than implementing the same algorithm in software. For example, a software implementation may intend to employ intelligent branching to avoid some computation. But in hardware, it is often attempted to simplify the fundamental operations, like the MAC instruction in DSP, to accelerate the calculations and to provide more parallelism. Mapping these functions, such as file I/O, outer loop management and other house-keeping tasks, onto hardware is time-consuming and usually not profitable in terms of speedup. It is better to use hardware to unroll an inner loop for the maximum data flow. The granularity of pipeline is considered as well. The pipeline of hardware is achieved in register level while there is only limited instruction level pipelining in software implementations on traditional CPU-based frameworks.

III. KACU: K-means with Hardware Centroid Updating

A. K-means Algorithm

Basically, K-means is a partition-based algorithm to group objects based on certain features into K number of clusters. The grouping is done by minimizing the distance between

Algorithm Standard K-means

```
1: $closet\_centroid[] \leftarrow rand();$
2: while ($object\_move \neq 0$) do
3:     $object\_move \leftarrow 0;$
4:     $ACC[] \leftarrow 0;$
5:     $NUM[] \leftarrow 0;$
6:     for ($i = 0$ to $n$) do
7:         $min\_distance \leftarrow MAX_INT;$
8:         for ($j = 0$ to $k$) do
9:             $dist \leftarrow$ calculate distance between $object[i]$ and $centroid[j]$;
10:            if ($dist \leq min\_distance$) then
11:                $min\_distance \leftarrow dist;$
12:                $closest\_centroid[i] \leftarrow j;$
13:                $object\_move = 1;$
14: for ($i = 0$ to $n$) do
15:     $ACC[closest\_centroid[i]] += object[i];$
16:     $NUM[closest\_centroid[i]]++;$
17: for ($j = 0$ to $k$) do
18:     $centroid[j] \leftarrow \frac{ACC[j]}{NUM[j]};$
```

Fig. 1. Standard K-means algorithm

each data object and the corresponding cluster centroid. The standard K-means algorithm is shown in Figure1. First, in line 1, we randomly choose K values as the initial cluster centroids. Second, from line 6 to 13, each object is assigned to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster. Then, the centriods are updated from line 14 to 18, i.e. calculate the mean value of the objects for each cluster. Finally, K-means stops when no more new assignment occurs. Specifically, there are two major phases in K-means, i.e. *mutual distance calculation* (line 6-13) and *centroid updating* (line 14-18). While being updating after all objects are completely scanned in standard K-means algorithm, the centroids can be updated on the fly as well in continuous K-means [3] in which faster convergence is generally achieved. Formally, centroids are recomputed after each new assignment. That is, centroids must be updated when a new object arrives. Unfortunately, efficiently updating the centroids on the fly is difficult. The number of iterations is reduced due to faster convergence, but overall execution time may increase due to the massive routine incurred from frequent centroid updating. Consequently, in [7], in order to achieve the best performance, the authors presented the block updates approach which recomputed the centroids as long as B objects are reassigned. How to choose the value of B is a case by case question. $B = \sqrt{N}$, where N is the quantity of object data, is suggested in [7].

B. Motivation

As our study, there are some researches to develop hardware accelerators for K-means, such as [2][4][5] and [6]. In [2], the authors proposed an a pixel comparison device based on K-means clustering algorithm. Although they are the first to address the acceleration framework of K-means with dedicated hardware, the brute force hardware design is not suitable for the implementations in practical applications. Recently, there is one effective design proposed in [4] and the heart of the proposed design is systolic process array (abbreviated as SPA) which employed massive parallel concept and independent character of data to accelerate the processing speed. However, while mentioned two designs are in the light of the most time-consuming task, mutual distance calculation, in standard K-means, centroid updating would become a very heavy task in "continuous K-means" algorithm.

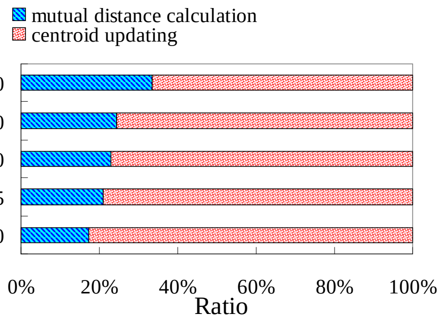

Fig. 2. Clock consumption ratio between the two procedures, mutual distance calculation and centroid updating, in different B. (After applying SPA, quantity of data=10000, dimension of data=2 and K=4)

Referring to Figure 2, when SPA is applied, the most part of time consumption in continuous K-means algorithm is the procedure of centroid updating instead of the procedure of mutual distance calculation. In view of this, we propose a framework to accelerate the speed of continuous K-means algorithm. To the best of our knowledge, we are the first to implement a hardware centroid-updating in continuous K-means algorithm. This feature distinguishes our work from others.

C. Hardware Design

Procedures, including mutual distance calculation and centroid updating, are all mapped to hardware implementation. The block diagram of the framework, KACU, is shown in Figure 3. The SPA can perform mutual distance calculation and the other design is used to execute the centroid updating process. With pipeline architecture, the control in KACU receives one data object and its closet centroid once. Then the control will add the value of the data object into its closest centroid's accumulator. Finally, a new centroid is generated and stored as a temporary. Note that although a centroid is recomputed while an object data arrives, it does not mean the centroids change all the time. The centroids update only when getting a command from CPU. Furthermore, due to pipeline construction, we only need a divider because only one centroid changes at a time. Traditionally, the results of each data object have to be returned to CPU after passing the

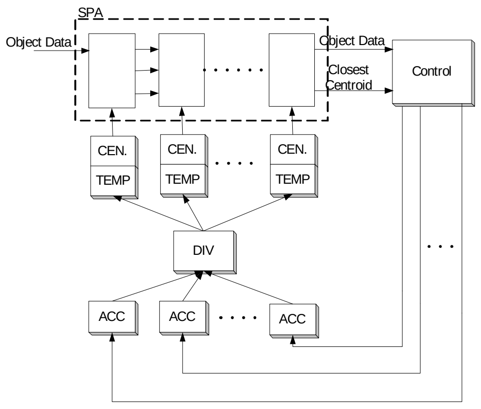

Fig. 3. Block diagram of KACU

process of mutual distance calculating. Our design does not need the return trip. Specifically, the procedure of centroid updating is accomplished by a dedicated hardware instead of CPU, so the communication overhead caused by returning results can be left out. After exploiting this simple hardware, we find that the time consumption of centroid updating can almost be ignored, as opposite to that of other procedure. Therefore, KACU is best suitable for continuous K-means algorithm which involves massive updating.

IV. Performance Study

The hardware is implemented and verified with Altera's design software QuartusII and is executed on an Altera's Stratix device with a NiosII 50MHz CPU and 16MB of SDRAM. Software implementation of the algorithm is executed on the same device. The synthetic dataset has ten thousands data objects with Gaussian distribution and be grouped into five clusters.

The following experiments are conducted to compare the clock consumption between SPA with software centroid updating and KACU, both in continuous K-means algorithm. Note that the number of iterations depends on the method used for initialization. Therefore, the clock number we measured is for one iteration only. First we see the comparing result in different B, i.e. period of centroid updating, in Figure 4. In most case, KACU is four times faster than SPA. Especially in very small value of B, the number of updating increases explosively but our proposed KACU can conduct in a constant time level. This result shows that KACU is more scalable than SPA.

We use five datasets with different D, i.e. dimension, to estimate the influence of dimension. The result is presented in Figure 5. Because KACU can update centroids in parallel form, the number of dimensions only affects the performance slightly. Oppositely, SPA with software centroid updating will be significantly impacted.

V. Conclusions

In this paper, we have proposed a novel paradigm to enhance algorithm in the field of data mining research and

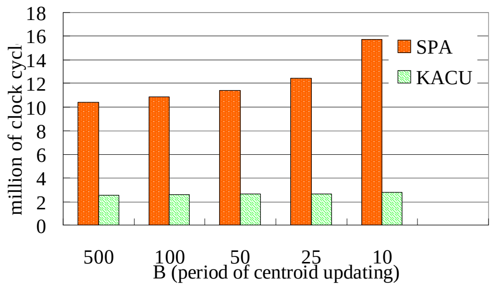

Fig. 4. The number of clock cycles needed for different B. (quantity of data=10000, K=4, dimension=2)

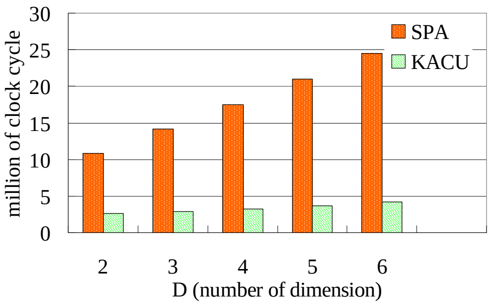

Fig. 5. The number of clock cycles needed for different D. (quantity of data=10000, K=4, B=100)

implemented successfully in commercial FPGA. The idea is to integrate an updating mechanism into a number of stages in pipeline to reduce a considerable processing time. From numerous experiments, our proposed framework can reach outstanding performance in different parameters. The KACU framework can accelerate with maximum speedup achieved of 5.6 speedup over the SPA with software centroid updating.

References

[1] Altera Corporation. http://www.altera.com.

[2] M. Estlick, M. Leeser, J. Theiler, and J. J. Szymanski. Algorithmic Transformations in the Implementation of K-means Clustering on Reconfigurable Hardware. In *Proceedings of the 2001 ACM/SIGDA 9th International Symposium on Field Programmable Gate Arrays*, pages 103–110, 2001.

[3] V. Faber. Clustering and the Continuous k-Means Algorithm. *Los Alamos Science*, 22:138–144, 1994.

[4] M. Gokhale, J. Frigo, K. McCabe, J. Theiler, C. Wolinski, and D. Lavenier. Experience with a Hybrid Processor: K-means Clustering. *The Journal of Supercomputing*, 26(2):131–148, 2003.

[5] M. Leeser, J. Theiler, M. Estlick, and J. J. Szymanski. Design Tradeoffs in a Hardware Implementation of the K-means Clustering Algorithm. In *Proceedings of the 1st IEEE Sensor Array and Multichannel Signal Processing Workshop*, pages 520–524, 2000.

[6] B. Maliatski and O. Y. Pecht. Hardware-Driven Adaptive K-means Clustering for Real-Time Video Imaging. *IEEE Transactions on Circuits System and Video Technology*, 15(1):164–166, 2005.

[7] C. Ordonez. Clustering Binary Data Streams with K-means. In *Proceeding of the 8th ACM Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12–19, 2003.

[8] J. Theiler, J. Frigo, M. Gokhale, and J. J. Szymanski. Co-design of Software and Hardware to Implement Remote Sensing Algorithms. In *Proceeding of SPIE*, pages 86–99, 2001.

[9] Xilinx Corporation. http://www.xilinx.com.