How to cite this paper:

Muhammad Firdaus Mustapha, Noor Elaiza Abd Khalid, Mazani Manaf, & Azlan Ismail. (2017). Enhanced parallel SOM based on heterogeneous system platform in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference of Computing & Informatics (pp 243-249). Sintok: School of Computing.

ENHANCED PARALLEL SOM BASED ON HETEROGENEOUS SYSTEM PLATFORM

Muhammad Firdaus Mustapha[1], Noor Elaiza Abd Khalid[2], Mazani Manaf[3] and Azlan Ismail[4]

[1]*Universiti Teknologi MARA, Shah Alam, Malaysia*, firdaus19@gmail.com
[2]*Universiti Teknologi MARA, Shah Alam, Malaysia, elaiza@tmsk.uitm.edu.my*
[3] *Universiti Teknologi MARA, Shah Alam, Malaysia*, mazani@tmsk.uitm.edu.my
[4] *Universiti Teknologi MARA, Shah Alam, Malaysia, azlanismail08@gmail.com*

ABSTRACT. In this paper, we propose an enhanced parallel Self-organizing Map (SOM) framework based on heterogeneous system platform, specifically Central Processing Unit (CPU) and Graphic Processing Unit (GPU) soldered together on a single chip. The framework is to improve speed of parallel SOM using GPU since processing parallel SOM on GPU burden by communication latency due to isolate device architecture with CPU. The parallel SOM has been extended to heterogeneous system platform and double kernel for calculation distance and find Best Matching Unit (BMU) are introduced. The results are tested using benchmark data on two different platforms: GPU and heterogeneous system. The proposed framework shows improvement compared to standard parallel SOM on GPU and heterogeneous system.

Keywords: Parallel Self-organizing map, GPU computing, OpenCL, HSA

INTRODUCTION

Self-organizing Map (SOM) is one of the data analysis techniques that have gained popularity over the past few decades. The main disadvantage of SOM is training process is quite time consuming especially for processing distance calculation and update the weight of neurons on SOM map (Xiao, Feng, Han, & Leung, 2014). In order to increase SOM processing, many researchers parallelized the algorithm. One of the promising solutions is using Graphic Processing Unit (GPU) (De, Zhang, & Guo, 2016; Richardson & Winer, 2015). Lachmair et. al (2013) and Wittek & Darányi (2013) reported that running the parallel SOM on GPU variant achieve the speed up for large data compared to Central Processing Unit (CPU). Meanwhile, Gajdos & Platos (2013) reported that executing parallel SOM on GPU reduces computation time when input dimension and SOM mapping size are increasing compared to CPU version. However, some researchers stated that processing of the parallel SOM on GPU could be burden by imposing larger mapping size and feature dimensions (Gajdos & Platos, 2013; Hasan, Shamsuddin, & Lopes, 2014; McConnell, Sturgeon, Henry, Mayne, & Hurley, 2012). Moreover, memory utilization will increase when processing large mapping size which leads to high rate of memory transfer (Hasan et al., 2014).

A quite recent trend in GPU computing is to simplify the programming model for GPU based program. Heterogeneous Uniform Memory Access (HUMA) specification has released by Heterogeneous System Architecture (HSA) foundation, a consortium of companies and

universities that provides the programmer with shared virtual address space between CPU and GPU (Power, Hestness, Orr, Hill, & Wood, 2015). One of the GPU programming framework companies, Khronos Group has released OpenCL 2.0 which supports HUMA specification (Opencl, 2014). The heterogeneous systems that combine CPU and GPU on a single chip are capable to share the same memory which leads to improve communication between each other. One of the features in OpenCL 2.0 is Shared Virtual Memory (SVM) introduced to reduce overhead by eliminating deep copies during host-to-device and device-to-host data transfers (Mukherjee, Sun, Blinzer, Ziabari, & Kaeli, 2016). There are two ways of implementing SVM: coarse-grained and fine-grained. The coarse-grained SVM provides synchronization during mapping and unmapping of memory objects meanwhile for fine-grained SVM, the synchronization occurs during the implementation of program (Mukherjee et al., 2016).

Temporarily, several steps in SOM algorithm had been decomposed by researchers in order to execute in parallel. There are many different formations of parallelized SOM found in the literature. It could be stated that most researchers attempt to parallelize calculate distance and find Best Matching Unit (BMU) steps. Both steps is identified as most time consuming steps in SOM processing (Kohonen, 2013). There are several researchers who decomposed SOM algorithm into three steps: calculate distance, find BMU, and update neurons' weights (De et al., 2016; Gajdos & Platos, 2013; McConnell et al., 2012; Wang, Zhang, & Créput, 2013). Some of the researchers decomposed initialize neuron weights (De et al., 2016; Lachmair et al., 2013).

Accordingly, this paper proposes an enhanced parallel SOM framework based on heterogeneous system platform. The design of this framework is to improve processing speed which triggered by SVM feature in OpenCL 2.0. Basically, the framework consists of three parallelize steps in SOM algorithm that has been separated into three kernels. Moreover, duplicate kernels have been introduced to calculate distance and find BMU kernels. The framework has been evaluated by measuring total computation time and compared with parallel SOM GPU version and parallel SOM heterogeneous system version.

METHODOLOGY

Previous studies that highlight parallel SOM have been successfully executed on GPU. Almost all the researchers in the literature apply parallelism at calculate distance and find BMU steps. There are many of them apply parallelism at update weight step. Consequent of that we propose to parallelize these three steps into the proposed framework. Meanwhile, heterogeneous system compromises a promising solution for reducing latency in communication between CPU and GPU. In order to gain these advantages, our proposed work is utilizing OpenCL 2.0 platform which specifically SVM feature. The implementation of our work is based on fined-grained SVM buffers. The fined-grained SVM buffers are synchronized during the implementation of SVM buffer which could reduce communication latency between CPU and GPU. The design of proposed framework is extended from our previous work (Mustapha, Abd Khalid, & Ismail, 2017) where the proposed framework introduces the duplicate kernels for distance calculation and find BMU as depicted in Figure 1. The main reason of duplicating the kernels is to increase utilization of work units in GPU. This work is supported by OpenCL where OpenCL allows a programmer to create more than one queue for execution. The queue will process based on out-of-order execution (Opencl, 2014). In out-of-order execution mode there is no guarantee that the enqueued commands will finish in order they were queued. For instance, the execution of the Calculate Distance Kernel_1 and Calculate Distance Kernel_2 might overlapping in the same time which could lead to increase the utilization of work units in GPU side.

In depth of our proposed framework, the framework begins with initializing SOM parameters such as learning factor and weights at the host side. The input data is retrieved and stored into an array. These tasks are performed at host side. Each kernel at the GPU side is provoked by function respectively. The functions also provide setting, initializing parameters, and call the kernels. For example, the calculate distance function is used to call Calculate Distance kernel and it is done the same way with the other two kernels.

The first kernel is Calculate Distance kernel that is used to calculate the distance between neurons and current input vector. The amount of work units are employed to parallelize the calculation distance step is mapped by amount of work-items on GPU where the amount of work-items is equal to the number of neurons in the SOM map. Specifically, each work-item of the kernel is responsible for finding the distance between a single neuron and the current input vector. This research applies Manhattan distance calculation.

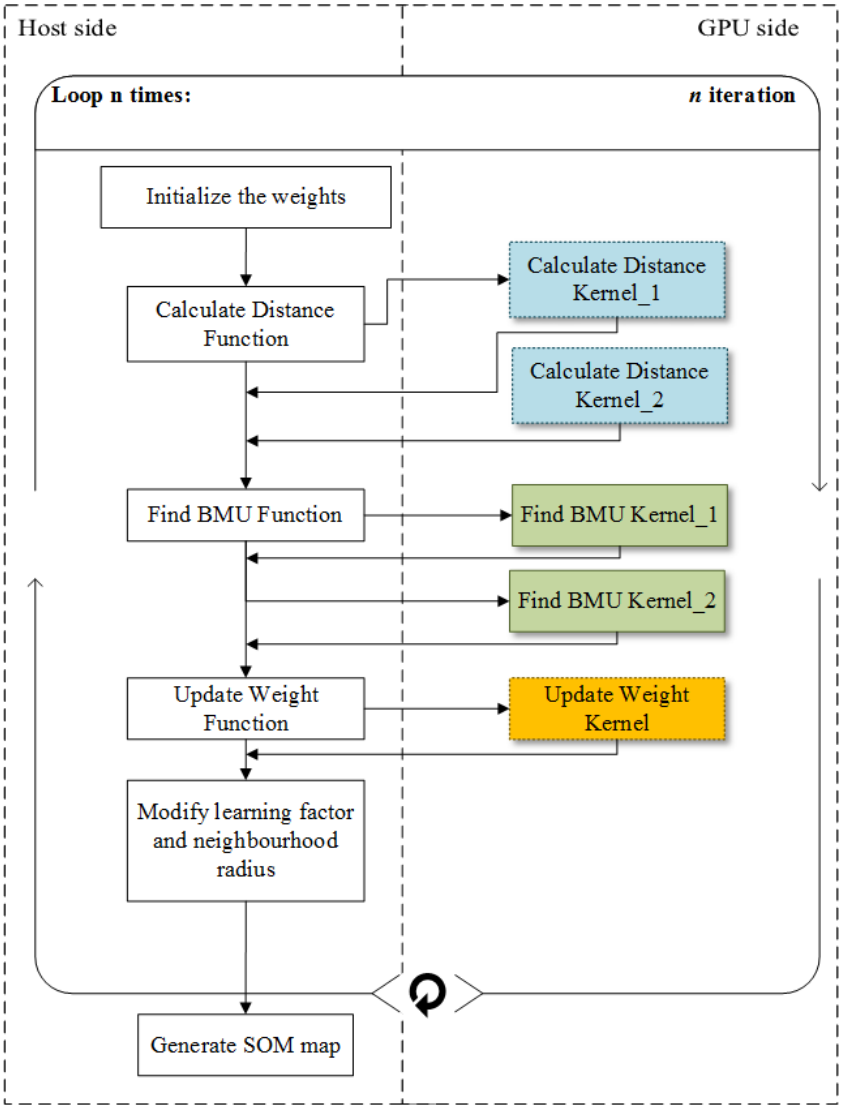

Figure 1. Enhanced parallel SOM framework.

In the meantime, the Find BMU kernel applies two stages reduction method (Bryan Catanzaro, 2010). The kernel utilizes work items the same amount of neurons on SOM map. The first stage of reduction method is to find the minimum distance for each local work

group. The values of minimum distances of each work group will be stored into local array. The second stage is to find the minimum distance for each Compute Unit (CU). The minimum values of each CU are then stored into global array and the host will determine the winning neurons.

The Update Weight kernel is the third kernel in the framework updates the weight of neurons based on learning rate and neighborhood function. The learning rate defines how much a neuron's vector is altered through an update with referring to how far the distance of the neuron from the BMU on the map. The BMU and its close neighbors will be altered the most, while the neurons on the outer edges of the neighborhood are slightly changed. Immediately after executing the three kernels, the learning factor and neighborhood radius are updated with the new values. All of the steps include in the loop block will be repeated until n iteration or epoch before the SOM map is generated.

COMPUTATIONAL RESULTS

Two series of experiments have been conducted to evaluate the proposed framework. The experiments employ Bank Marketing benchmark data set that was taken from UCI Machine Learning Repository. For each experiment, three versions of parallel SOM have been tested: parallel SOM on GPU (PSG), parallel SOM on heterogeneous system (PSH), and enhanced parallel SOM on heterogeneous system (ePSH). The experimental design of the experiments is depicted in Table 1. In the experiments, each dataset is tested using four different map sizes in order to find the best map size (Mustapha et al., 2017). The performance measurement is based on time in seconds. Four kinds of time have calculated from the execution of each kernel and the total time. These experiments have been conducted on a laptop equipped with Intel Skylake i7-6700HQ processor which built in Intel® HD Graphics 530. The processor supports OpenCL 2.0 specifications.

Table 1. The experimental design.

Dataset parameters	SOM parameters		Performance Measurement	Algorithm versions
No. of Samples	Iterations	Map sizes	Time, s	
5000 (3 parameters)	30	10x10 20x20 30x30 40x40	1) Calculate Distance, 2) Update weight, 3) Find BMU, 4) Total time	1) Parallel SOM on GPU (PSG), 2) Parallel SOM on HSA (PSH), 3)Enhanced Parallel SOM HSA (ePSH)
15000 (3 parameters)				

The results for each experiment have been collected are shown in Figure 2 and Figure 3. Figure 2 depicts the results of the first series of experiments which employ 5000 dataset. From the graph, the ePSH that consists of the proposed framework capable to reduce total time compared to PSH and PSG. Based on relative different of total time, ePSH reduces time 4 to 10 percent over PSH and 34 to 39 percent over PSG.

Meanwhile, the second series of experiments also prove ePSH perform faster than PSG and PSH. Figure 3 shows the results of the second series of experiments that apply 15000 dataset. The ePSH shows better results which score 6 to 15 percent over PSH and 39 to 43

percent over PSG. Moreover, both series of experiments illustrate that ePSH perform better when utilizes larger dataset as the relative different values increase.

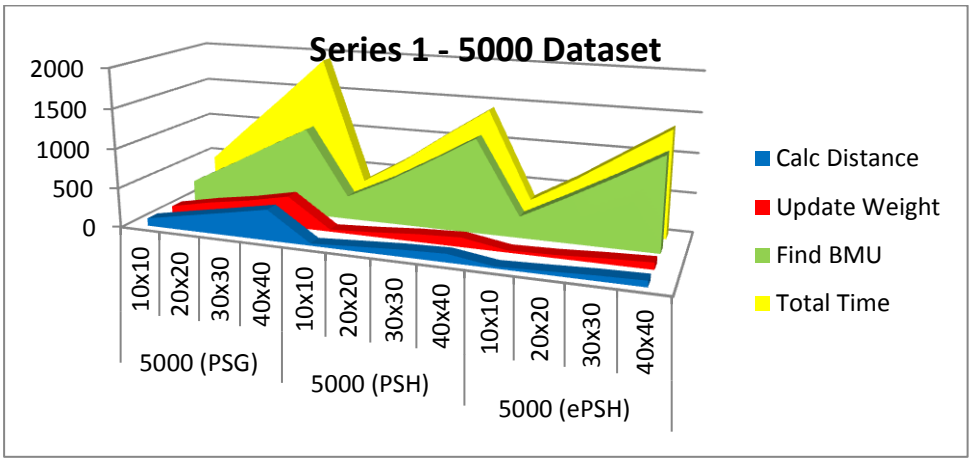

Figure 2. The results of the first series of experiments

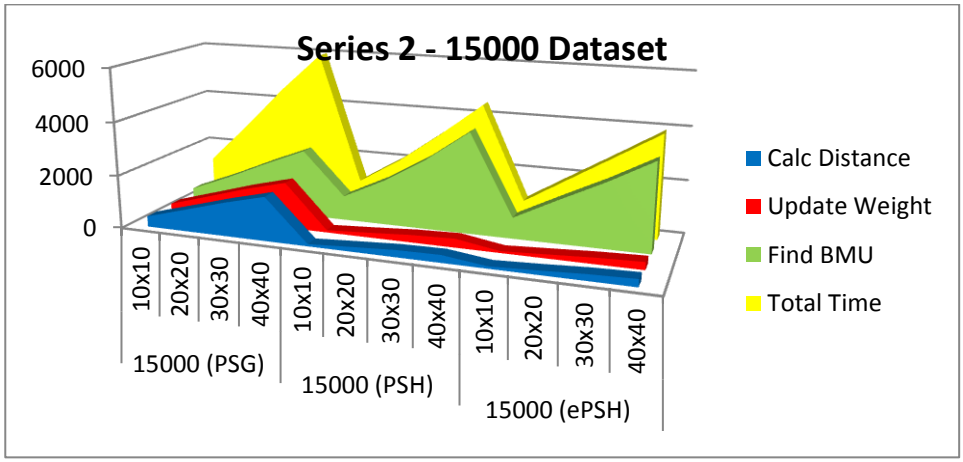

Figure 3. The results of the second series of experiments

CONCLUSION

In this paper, we propose an enhanced parallel SOM framework that based on heterogeneous system. The framework is extended from parallel SOM researches that consist of three kernels: calculate distance kernel, find BMU kernel, and update weight kernel. The proposed framework is included with two calculate distance kernel and two find BMU kernel. The proposed framework is designed with the aim to increase the utilization of processing element on GPU. From the experimental results, the proposed framework achieves better in total time processing compared to PSG and PSH. The proposed framework also can be realized by using heterogeneous platform where it offers efficient communication between CPU and GPU. In the future, we will extend the proposed framework with more duplicating kernels in order to study their performances.

ACKNOWLEDGMENTS

This work was funded by Ministry of Higher Education (MOHE) of Malaysia, under the FRGS, grant no. FRGS/1/2015/ICT02/UITM/02/6 and Academic Staff Bumiputera Training Scheme (SLAB). The authors also would like to thank the Universiti Teknologi MARA for supporting this study.

REFERENCES

Bryan Catanzaro. (2010). OpenCL™ Optimization Case Study: Simple Reductions. Retrieved from http://developer.amd.com/resources/articles-whitepapers/opencl-optimization-case-study-simple-reductions/

De, A., Zhang, Y., & Guo, C. (2016). A parallel image segmentation method based on SOM and GPU with application to MRI image processing. *Neurocomputing*, *198*, 180–189. http://doi.org/10.1007/978-3-319-12436-0_71

Gajdos, P., & Platos, J. (2013). GPU Based Parallelism for Self-Organizing Map. In *Advances in Intelligent Systems and Computing* (Vol. 179, pp. 3–12). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-31603-6

Hasan, S., Shamsuddin, S. M., & Lopes, N. (2014). Machine Learning Big Data Framework and Analytics for Big Data Problems. *International Journal Advance Soft Computing Applications*, *6*(2), 1–17.

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks : The Official Journal of the International Neural Network Society*, *37*, 52–65. http://doi.org/10.1016/j.neunet.2012.09.018

Lachmair, J., Merényi, E., Porrmann, M., & Rückert, U. (2013). A reconfigurable neuroprocessor for self-organizing feature maps. *Neurocomputing*, *112*, 189–199. http://doi.org/10.1016/j.neucom.2012.11.045

McConnell, S., Sturgeon, R., Henry, G., Mayne, A., & Hurley, R. (2012). Scalability of Self-organizing Maps on a GPU cluster using OpenCL and CUDA. *Journal of Physics: Conference Series*, *341*, 12018. http://doi.org/10.1088/1742-6596/341/1/012018

Mukherjee, S., Sun, Y., Blinzer, P., Ziabari, A. K., & Kaeli, D. (2016). A Comprehensive Performance Analysis of HSA and OpenCL 2.0. *2016 IEEE International Symposium on Performance Analysis of Systems and Software*, (April). http://doi.org/10.1109/ISPASS.2016.7482093

Mustapha, M. F., Abd Khalid, N. E., & Ismail, A. (2017). Research Article Evaluation of Parallel Self-organizing Map Using Heterogeneous System Platform. *J. Applied Sci.* http://doi.org/10.3923/jas.2017.Research

Opencl, K. (2014). *OpenCL Specification*. *ReVision*. http://doi.org/10.1016/j.actamat.2006.08.044

Power, J., Hestness, J., Orr, M. S., Hill, M. D., & Wood, D. A. (2015). gem5-gpu: A Heterogeneous CPU-GPU Simulator. *IEEE Computer Architecture Letters*, *14*(1), 34–36. http://doi.org/10.1109/LCA.2014.2299539

Richardson, T., & Winer, E. (2015). Extending parallelization of the self-organizing map by combining data and network partitioned methods. *Advances in Engineering Software*, *88*, 1–7. http://doi.org/10.1016/j.advengsoft.2015.05.003

Wang, H., Zhang, N., & Créput, J.-C. (2013). A Massive Parallel Cellular GPU Implementation of Neural Network to Large Scale Euclidean TSP. In F. Castro, A. Gelbukh, & M. González (Eds.), *Advances in Soft Computing and Its Applications: 12th Mexican International Conference on Artificial Intelligence, MICAI 2013, Mexico City, Mexico, November 24-30, 2013, Proceedings, Part II* (pp. 118–129). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-45111-9_10

Wittek, P., & Darányi, S. (2013). Accelerating text mining workloads in a MapReduce-based distributed GPU environment. *Journal of Parallel and Distributed Computing*, *73*(2), 198–206. http://doi.org/10.1016/j.jpdc.2012.10.001

Xiao, Y., Feng, R.-B., Han, Z.-F., & Leung, C.-S. (2014). GPU Accelerated Self-Organizing Map for High Dimensional Data. *Neural Processing Letters*. http://doi.org/10.1007/s11063-014-9383-4