2014

Portable GPU-Based Artificial Neural Networks For Data-Driven Modeling

Zheng Yi Wu

PORTABLE GPU-BASED ARTIFICIAL NEURAL NETWORKS FOR ACCELERATED DATA-DRIVEN MODELING

ZHENG YI WU (1), MAHMOUD ELMAGHRABY (2)
(1): Applied Research, Bentley Systems, Incorporated, Watertown, CT 06795, USA
(2): Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06268, USA

Artificial neural network (ANN) is widely applied as the data-driven modeling tool in hydroinformatics due to its broad applicability of handling implicit and nonlinear relationships between the input and output data. To obtain a reliable ANN model, training ANN using the data is essential, but the training is usually taking many hours for a large data set and/or for large systems with many variants. This may not be a concern when ANN is trained for offline applications, but it is of great importance when ANN is trained or retrained for real-time and near real-time applications, which are becoming an increasingly interested research theme while the hydroinformatics tools will be an integral part of smart city operation systems. Based on author's previous research projects, which proved that the GPU-based ANN is over 10 times more efficient than CPU-based ANN for constructing the meta-model to be applied as the surrogate of a physics-based model, this paper presents the latest development of the GPU-based ANN computing kernels that is implemented with OpenCL an Open Compute Language. The generalized ANN can be used as an efficient machine learning library for data-driven modeling. The performance of the implemented library has been tested with the benchmark example of water distribution system modeling and compared with the previous results.
Keywords: machine learning, artificial neural network, accelerated computing, data-driven model

INTRODUCTION

A data-driven model (DDM) is defined as the mappings (or connections) between a system's inputs and its outputs. The mappings are usually represented by using a set of mathematic equations and learned from a given dataset and validated with another independent dataset. Various DDM techniques have been developed with different mapping representation schemes and learning algorithms. Among them, artificial neural network (ANN) is one of the most popular DDM methods. ANN, along with other computational intelligence methods, is extensively applied to build data-driven models in the areas of hydroinformatics. Solomatine and Ostfel [1] provided a good overview of DDM research works published in hydroinformatics. As indicated by them, the DDM research has passed the first stage of excitement for and experiment of the power of DDM approaches. The new challenge is to build

an effective and robust DDM as useful predictor tool for practicing engineers and managers to solve the real world problems.

Over last decade, ANN has been applied to water distribution analysis. In particular, ANN is used for capturing the hydraulic characteristics of a water distribution network. Broad, Dandy, and Maier [2] applied the ANN to replace the hydraulic and water quality model for prediction of water age in water distribution systems. Rao and Alvarruiz [3] applied ANN to emulate the selected hydraulic characteristics of a water distribution system. Both applications aimed at using the trained ANN as a meta-model for optimizing water distribution operation. Although ANNs enable a tremendously speed-up for the hydraulic prediction in contrast to relatively slow hydraulic models, the times for their training on conventional central processing unit (CPU) can become prohibitively long with large networks required for real systems. The training algorithms such as back propagation (BP) and its variants for training of feed-forward ANNs are desirable for parallel processing, particularly on Single Instruction Multiple Thread (SIMT) architecture of the modern Graphics Processing Units (GPUs) [4], such as the algorithms developed by Lopes et al. [5] and Lopes and Ribeiro [6], using NVIDIA's Compute-Unified Device Architecture (CUDA). Wu and Eftekharian [7] demonstrated that GPU-based ANNs are the potentially powerful tools for capturing water distribution system's hydraulic characteristics. The work was continued by Behandish and Wu [8] on the development of a reliable meta-model to replace the hydraulic model for extended-period simulation, in spite of errors being accumulated as the ANN is called in a succession of time. The developed meta-model was later embedded in a parallel Genetic Algorithm (GA), Darwin Optimization Framework [9], for near-optimal pump scheduling. This optimization technique, which was accelerated from several hours to a few minutes with the use of ANN meta-model, was investigated for real-time control of pump operation by Wu and Behandish [10].

Although the previous research by the authors have proved that GPU-based ANN is more efficient than CPU-based ANN for constructing the meta-model that is applied as a surrogate of the physics-based model, the computing kernels based on CUDA for the ANN training are limited to NVIDINA GPU architecture. To generalize the GPU-based ANN as an accelerated data-driven modeling framework, it is essential to develop a portable GPU-based ANN. Using OpenCL is the choice of the development for this project, as OpenCL provides a GPU computing platform independent from hardware vendors. OpenCL is an industry standard for programming heterogeneous platforms that are composed of a combination of CPUs, GPUs and other accelerated processors.

ACCELERATED ANN ARCHITECTURE

To develop a robust and accelerated ANN using GPU as general purpose computing platforms, it is important to use a framework to program the GPU. OpenCL provides such framework for writing programs that execute across heterogeneous platforms. Figure 1 illustrates the architecture of the OpenCL-based computing model. The platform model includes a single host that is connected to one or more compute devices. The compute device is where the kernels execute (functions that run on the GPUs). A device can be a CPU, a GPU, a DSP or any other processor provided by the hardware and supported by the OpenCL vendor. The devices are further divided into compute units, which are further divided into one or more processing elements. Computation is done within the processing elements. The idea is to transfer heavy computations from the CPU to the GPU by launching the suitable kernel on the GPU. The kernel is consisting of thousands of work-items that are running the same set of instructions, but

with different data. Each work-item is running on a single processing element. The work-items are grouped in what is called a work-group that runs with one compute unit.

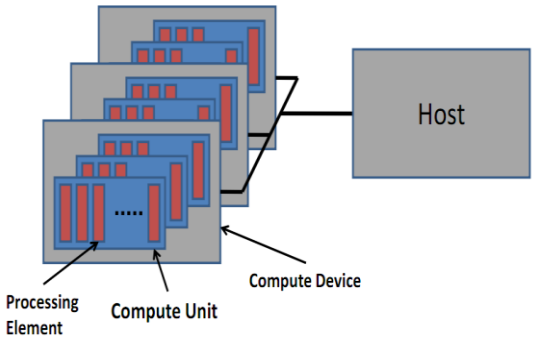
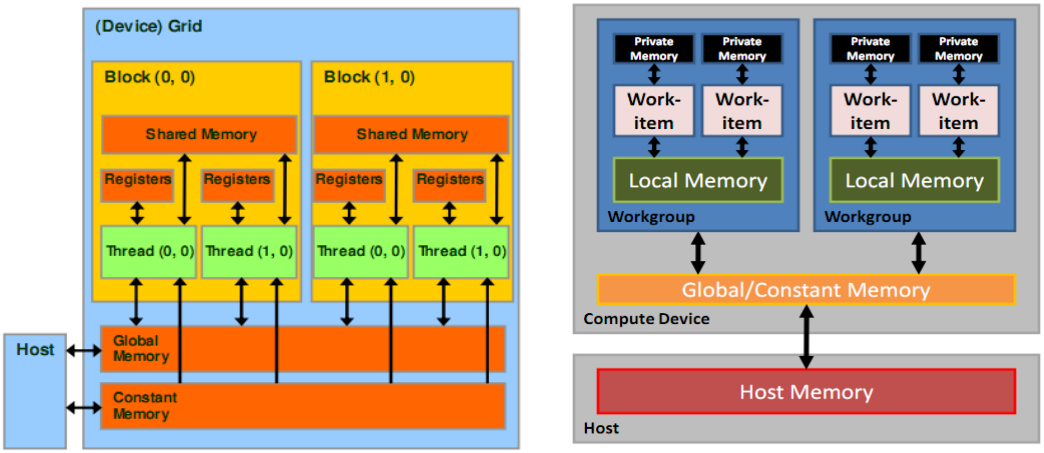


Figure 1. Architecture of OpenCL-based computing platform

Figure 2 shows the memory hierarchy of the GPUs for the accelerated computing by using both CUDA and OpenCL. The memory structures are very much similar to each other with just some differences in terminology. For the OpenCL memory hierarchy, each work-item has its own private memory, and all work-items of the same work-group are sharing what is called a local memory. All work-groups have an access to the global memory. On the other hand, with CUDA, each thread has its own local memory, and the block of threads shares the same local memory. It is the same hierarchy with a different terminology.

The OpenCL ANN Library, or OPENCLANNLib, is the ANN library that is based on OpenCL. This makes the resulting implementations portable and runs on any type of GPUs, not just NVIDIA GPUs. We follow the structure of the public-domain library GPUMLib [5, 6] and introduced extra components plus all necessary modifications to the structure and the implementation.



(a) CUDA-based memory hierarchy (b) OpenCL-based memory hierarchy

Figure 2. Memory hierarchy for GPU-accelerated computing

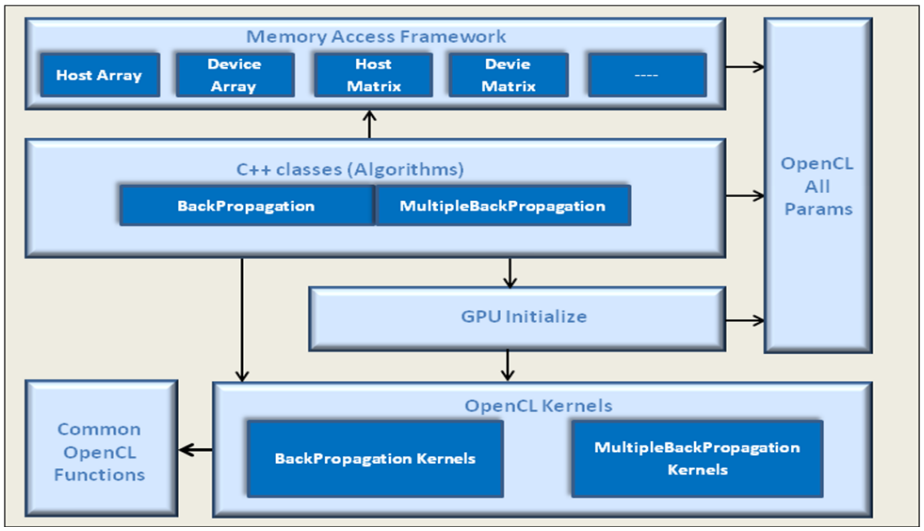


Figure 3. Architecture of OpenCL accelerated ANN Library

Figure 3 shows the architecture of the OpenCL ANN Library. The library consists of the following components:

- **Training Algorithms.** This is the main component that consists of two classes for Back Propagation and Multiple Back Propagation. This component is responsible for implementing the selected algorithm. It uses the Memory Access framework for data allocation and data transfer, uses the GPU Initialize component to initialize the GPU, gets the OpenCL parameters, and calls the kernels when needed.
- **Memory Access Framework.** It contains many classes that allow the allocation and transfer of the data between CPU and GPU. Some OpenCL runtime APIs are used to create buffers in the device memory, and write to or read from the device memory (transfer data between CPU and GPU). Also, the library is used for providing fast matrix multiplication. Some OpenCL parameters are required for memory allocation and data transfer. This takes us to the next component OpenCL All Params.
- **OpenCL All Params.** This is an important component that holds the OpenCL parameters. Those parameters are important for the Memory Access framework as it is required to allow the transfer of data. They are also important for launching the kernels.
- **GPU Initialize.** This component is responsible for initializing the GPU, and getting the OpenCL parameters of the GPU. Those parameters are saved with the OpenCL All Params component.
- **OpenCL Kernels.** This component includes eleven kernels that are called by the Back Propagation and Multiple Back Propagation methods. For instance, those kernels include the kernels to calculate the outputs of all neurons in a given layer, and the kernels to calculate the Root Mean Square (RMS) error of the neural network training.

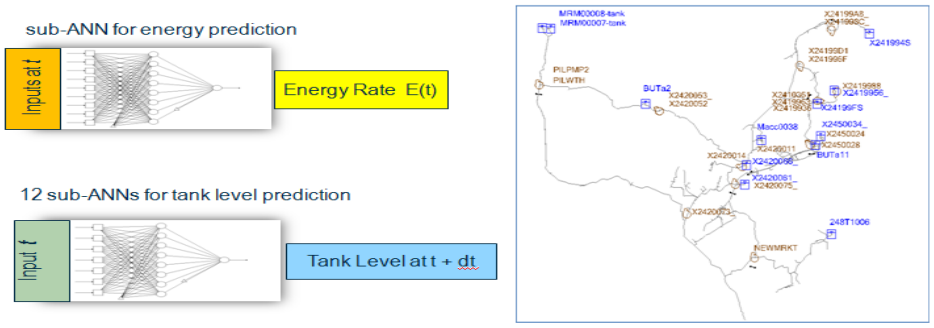


Figure 4. Test example and ANN configurations

APPLICATION

In the previously published works [7, 8], the ANN was trained to surrogate the extended period simulation (EPS) of a water distribution system. The training was done on GPU using the GPUMLib, so the computing kernels are limited to NVIDINA GPU architecture. To compare and test the OpenCL-based ANN library, the same example was used with the same training data and ANN configurations, including:

- 13 different ANNs have been used to model the operation of the water distribution system.
- The trained networks are used to simulate a 24 hours of operation. The aggregate energy consumption of the system as well as the tank levels of 12 different tanks is reported.

ACCURACY AND PERFORMANCE

It was expected that the generated weights by both of the CUDA-based ANN implementation and the OpenCL-based ANN implementation should be identical. As shown in Figure 5, the trained ANN weights are not exactly the same although they are very close in value. The reason is that the GPUMLib is using asynchronous function to update the RMSE value. This value is used later for weight correction. This asynchronous access does not guarantee that the RMSE value is the most recent one. This problem has been solved in the OpenCL version of the ANN implementation.

Figure 5. Comparison of ANN weights obtained with CUDA and OpenCL

As shown in Figure 5, the weights are identical for eight networks, while for five other networks the weights are not identical. The maximum average absolute error is for Tank_10 weights. It is about 0.08 while the maximum average relative error is about 12% for Tank_11 weights. Therefore, we will show the results in more details for one of the best networks, Aggregate energy network, as well as the network that has the largest weight differences, Tank_11.

The 24 hours simulation of the Oldham system operation using the ANN is shown in Figure 6 and Figure 7. Figure 6 shows the Aggregate energy by the hydraulic model, the ANN results using CUDA and the ANN using OpenCL. As we can see, there is no difference at all between the CUDA results and the OpenCL, but only small differences comparing the results of the trained network (CUDA or OpenCL) with the hydraulic model.

Figure 7 shows Tank_11 results. As we can see, there are very slight difference between the CUDA results and the OpenCL, and minor differences comparing the results of the trained network (CUDA or OpenCL) with the hydraulic model.

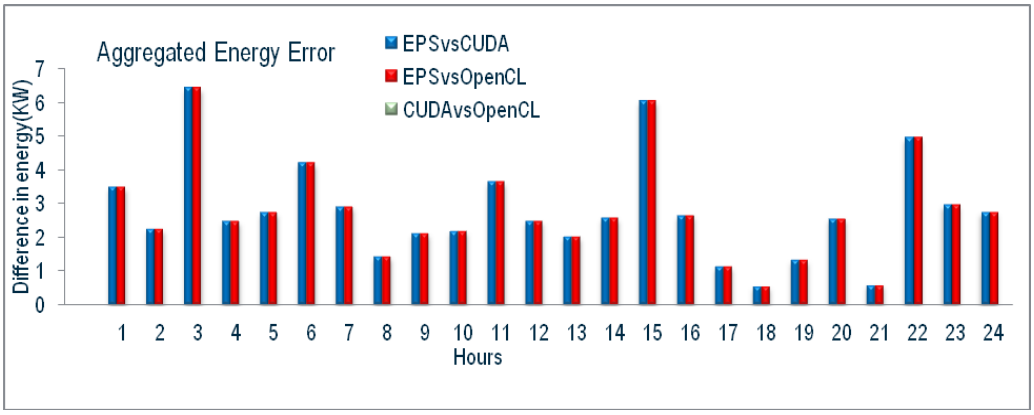


Figure 6. Comparison of energy consumptions simulated by EPS hydraulic model and ANNs

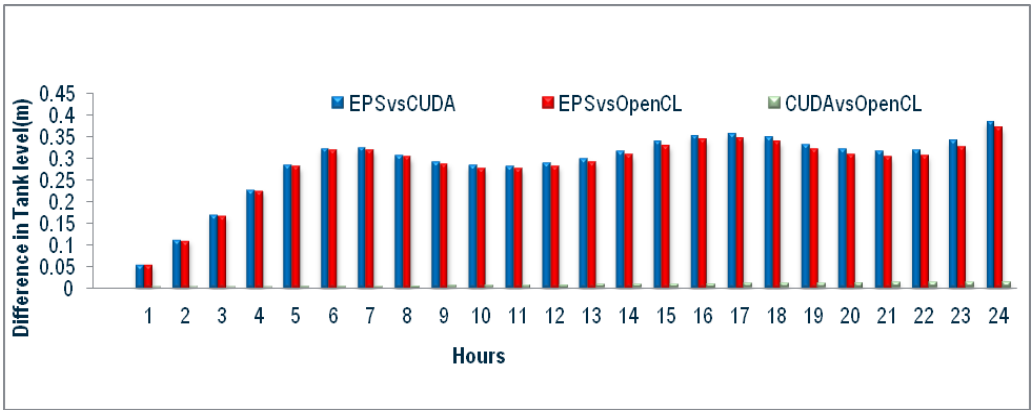


Figure 7. Comparison of tank levels simulated by EPS hydraulic model and ANNs

To test the performance of the library, two different GPUs were used, including:

- **Quadro FX 770M**: 32 GPU cores, with core clock rate of 500MHz, shader clock rate of 1250 MHz.

- **GTX 460**: 336 GPU cores, with core clock rate of 650MHz, shader clock rate of 1300 MHz.

The overall runtime in both GPUs is shown in Figure 8. The runtime of CUDA implementation on the Quadro GPU takes about 31 min. The runtime of OpenCL implementation on the Quadro GPU takes about 46 min. The runtime of CUDA implementation on the GTX460 GPU takes about 7 min. The runtime of OpenCL implementation on the GTX460 GPU takes about 18 min. The results show that the CUDA library is still faster than the OpenCL library. This is because CUDA is developed specifically for NVIDIA GPUs, and it is more mature, and has some features that are not available yet from OpenCL, such as the implementation support for pointers to pointers. Still the main advantage of OpenCL is its portability. The results show too that the OpenCL-based ANN implementation is scalable, and the performance is improved when running on a more powerful GPU.

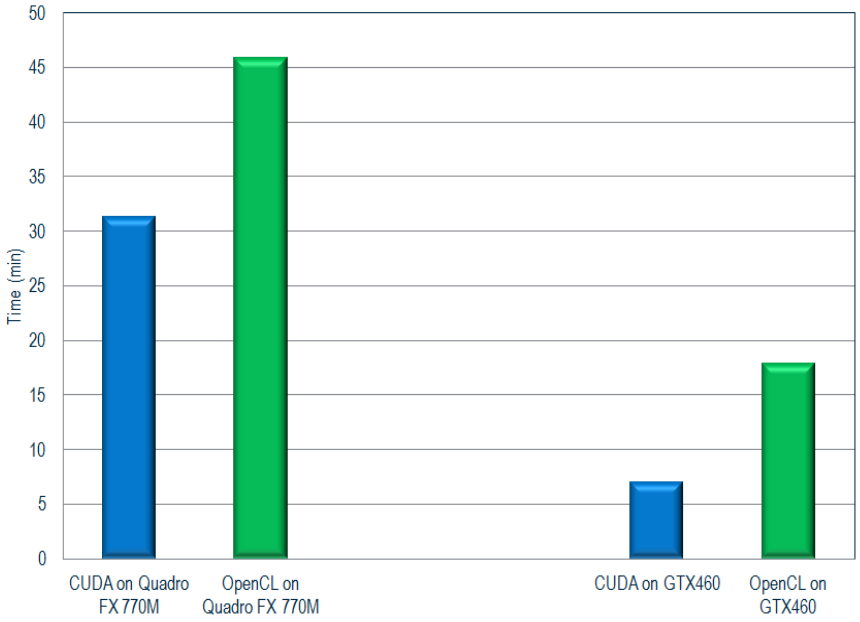


Figure 8. Computing performance of ANN accelerated by CUDA and OpenCL

CONCLUSIONS

This paper presented a generalized and accelerated ANN data-driven model, which has been implemented by using OpenCL. The approach has been tested on the example of constructing a meta-model for a real water distribution system. It has been found that (1) OpenCL-based ANN is as accurate as CUDA-based ANN; (2) OpenCL-based ANN is slightly slower than CUDA-based ANN; but (3) OpenCL-based ANN is still many times faster than CPU-based ANN, and most importantly (4) OpenCL-based ANN ensures the portability across different GPUs or accelerated computing units. In the near future, the work will be extended to take advantages of multiple GPUs, instead of one GPU, and also to be integrated with other machine learning techniques in a heterogeneous computing paradigm. It is believed that such an integrated high performance tool will enable engineers and researchers to effectively and efficiently undertake data-driven modeling in the field of hydroinformatics.

REFERENCES

[1] Solomatine, D. P. and Ostfeld A. "Data-driven modeling: some past experience and new approaches." *IWA Journal of Hydroinformatics,* 10.1, doi: 102166/hydro.2008.015, (2008).

[2] Broad D, Dandy G.C. and Maier H.R. "Water distribution systems optimization using metamodels", *Journal of Water Resources Planning and Management, ASCE* , 131 (3), (2005), pp172 – 180.

[3] Rao, Z., & Alvarruiz, F. "Use of an artificial neural network to capture the domain knowledge of a conventional hydraulic simulation model.", *Journal of Hydroinformatics,* 09(1), (2007), pp15-24.

[4] Jang H., Park A. and Jung K. "Neural Network Implementation using CUDA and OpenMP.", *Technical Report, Department of Digital Media, College of Information Science,* Soongsil University, South Korea, (2009).

[5] Lopes, N., Ribeiro, B., & Quintas, R. "GPUMLib: A new library to combine machine learning algorithms with graphics processing units." *10th International Conference on Hybrid Intelligent Systems.* Atlanta, GA, (2010).

[6] Lopes, N., & Ribeiro, B. "GPUMLib: An efficient open-source GPU machine learning library.", *International Journal of Computer Information Systems and Industrial Management Applications,* 3, (2011), 355-362.

[7] Wu, Z. Y., & Eftekharian, A. A. "Parallel artificial neural network using CUDA-enabled GPU for extracting hydraulic domain knowledge of large water distribution systems.", *ASCE Annual World Environmental and Water Resources Congress.* Palm Springs, CA, USA, (2011).

[8] Behandish, M., & Wu, Z. Y. "GPU-based ANN configuration and training for water distribution system analysis." *ASCE Annual World Environmental and Water Resources Congress.* Albuquerque, NM, USA, (2012).

[9] Wu, Z. Y., Wang, Q., Butala, S., and Mi, T. "Generalized framework for high performance infrastructure system optimization." *International Conference on Computing and Control for Water Industry.* Exeter, UK, (2011).

[10] Wu, Z. Y., & Behandish, M. "Real-time pump scheduling using genetic algorithm and artificial neural network based on graphics processing unit. ", *14th Water Distribution Systems Analysis Conference (WDSA 2012).* Adelaide, South Australia, (2012).