2014

Heterogeneous Computing Paradigm For Parallel Water Distribution System Analysis

Zheng Yi Wu

GPU-ACCELERATED WATER QUALITY ANALYSIS FOR WATER DISTRIBUTION SYSTEMS

ZHENG YI WU (1), SAAD QUADER (2)
(1): Bentley Systems, Incorporated, Watertown, CT, USA
(2): Ph.D. Candidate in Computer Science and Engineering, University of Connecticut, Storrs, CT, USA

As Graphics Processing Unit (GPU) is emerging as the general purpose computing devices, it is becoming more and more popular and powerful computing paradigm for conducting the numerical computation tasks that are traditionally undertaken by using Central Processing Unit (CPU). In this paper, the accelerated water quality (WQ) analysis solver is developed for efficient water quality simulation on GPU. The GPU-accelerated solver is implemented by using Open Compute Language (OpenCL), an industry standard for heterogeneous computing, to ensure the portability of the parallelized solvers on various hardware vendors' devices. With the parallelized WQ solvers, water quality simulations can be accelerated with the massive computing threads on a GPU. The parallelized solver has been tested with the large water distribution system (WDS) models on different GPUs, including those available from the public cloud computing platforms such as Amazon EC2, which offers very affordable and appealing computing capability that may not be available with some low-end GPU-enabled PCs. The performance analysis and comparison are presented in the paper for both conventional CPU-based and the GPU-accelerated water quality solvers.

INTRODUCTION

Hydraulic and water quality models have been developed and used over decades for water distribution system (WDS) analysis. The models prove to be powerful tools for engineers to gain understanding on the system conditions and conduct technically-sound WDS management. However, it takes significant amount of computational time to perform one simulation run for large systems with tens or hundreds of thousands of pipes over an adequately long period of duration in order to obtain good results for system hydraulic and water quality dynamics. Water quality simulation for a large water system usually requires for constructing an all-pipe model and executing the analysis of a long duration, which usually takes a lot of computation time, especially with chemical analysis. In particular, a real world model with tens or hundreds of thousands of pipes and junctions, the chemical analysis takes much more time than the hydraulics analysis for one simulation run. The time consuming water quality analysis is prohibitive for engineers to conduct many desirable WQ runs to obtain good solutions for various tasks, e.g. WQ model calibration, evaluation of normal and abnormal operation conditions etc. In addition, with the increasing interest in real-time or near real-time modeling for reducing the operation cost, the computational efficiency is essential for performing water distribution analysis. Furthermore, an efficient WDS WQ analysis solver will facilitate various

optimization applications for system design and operation. In the meantime, WDS analysis solvers are not developed to take the advantages of available computing hardware or process units, which are no longer homogeneous, but heterogeneous, including many cores of Central Processing Unit (CPU) and Graphics Processing Unit (GPU). Both CPUs and GPUs commonly co-exist in the personal computers, tablets and smart phones. Nowadays almost every machine has a specialized GPU for processing or displaying graphics. It is designed to execute many computing operations in parallel to meet the demand for high-end graphics applications. While a CPU has usually 2, 4, or 8 cores, a good GPU may have hundreds of cores, capable of running thousands of concurrent threads. It is attractive and desirable to harness this compute-capability for improving computation-intensive applications such as water quality analysis for water distribution system modeling.

In this paper, we report an accelerated computation architecture that parallelizes water quality analysis using GPU. The GPU parallelization is developed using the industry-standard OpenCL framework to ensure the portability of the parallelized solvers on various accelerated computing devices. With the parallelized water quality model, the water quality analysis can be speeded up on GPU with massively parallel computing threads. The parallelized solvers have been tested with large WDS models on different GPUs, including the high-end machines like those offered by the cloud computing platform e.g. Amazon EC2, which offers very affordable and appealing parallel computing capability that may not be available with the low-end GPU-enabled PCs. This paper presents the performance analysis of the parallelized solvers using the heterogeneous and portable parallel computing paradigm with CPU and GPU.

CONVENTIONAL WATER QUALITY ANALYSIS

The water quality analysis is to predict the reaction and transportation of the given constituent propagated with the flow throughout a distribution pipe network. Therefore, the flow/hydraulic characteristics must be the known conditions for performing water quality analysis. Conventionally, hydraulic simulation over an extended period is carried out first and the results are saved in a temporary file for performing water quality analysis. For each hydraulics time step, the hydraulics results are retrieved from the temporary file and the water quality calculation is performed by executing various computation functions [1], namely *segment update* (SU), accumulate, node update, source inputs, release etc. Then WQ solver outputs the quality values for the time step and proceeds to next time step. Our first task was to profile different parts of the existing water quality analysis implementation in order to find the parts that take the most computation time. It is desirable to port these bottlenecks to GPU.

Using a large benchmark model, we have performed various water quality analysis runs with different durations. It was found that chemical and age analysis takes significantly longer time than trace analysis, and the segment update part takes the bulk of this time. During quality analysis, each pipe is virtually divided into many segments and each segment's quality value is calculated for each time step. For a large model about 100,000 pipes with a long simulation duration (e.g. more than 100 hours) [2], a huge number of segments are required and the segment update takes about 80% of WQ computation time. Thus the SU method is identified as the bottleneck of water quality analysis, and selected as the computation task to be parallelized on GPU.

ACCELERATED WATER QUALITY ANALYSIS

The SU method implemented on CPU updates the segment one by one. To accelerate the computation, SU method is implemented on GPU where hundreds of threads can be used for executing SU method in parallel. Figure 1 illustrates the conceptual framework of the

accelerated WQ analysis with CPU and GPU. Using CPU as the host computing device, the computation proceeds on CPU for each time step and write segment data to GPU when SU method needs to be carried out for a given time step. Once SU execution completes on GPU, the results are retrieved back to resume the rest of the computation on CPU. This is a simple but effective parallelization strategy.

In order to ensure the portability, the SU method is implemented by using OpenCL (Open Compute Language) [3], which is the industry standard for heterogeneous computing. This enables the same parallel code to run in either GPU or multi-core CPU, depending on which one is available in the system. Two key challenges regarding memory and data structures had to be addressed for successfully implementing the SU method for GPU computing.

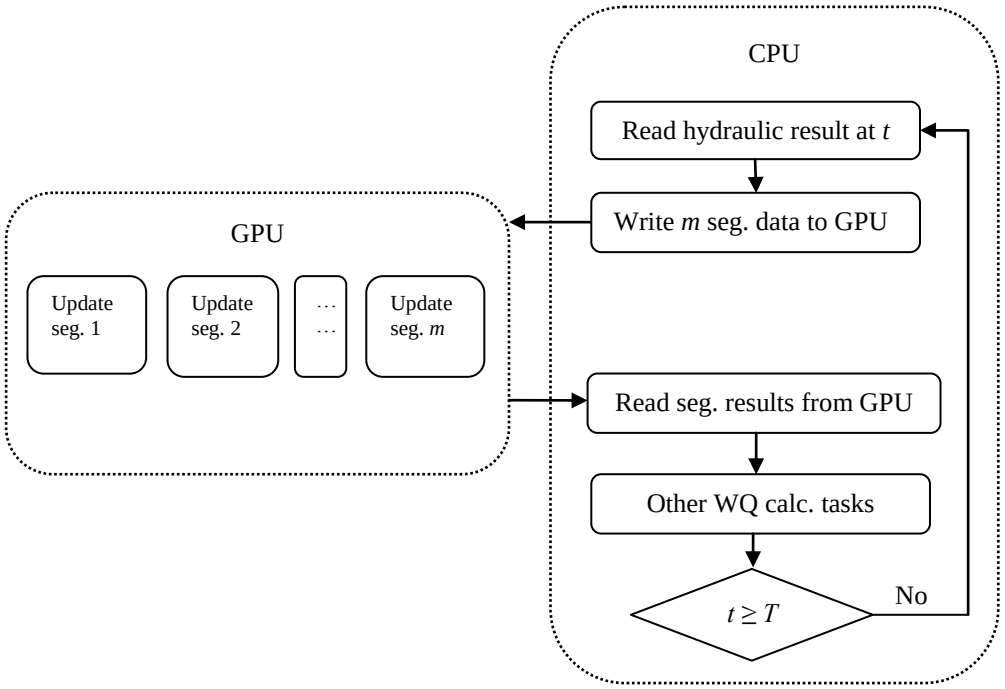

Figure 1. Architecture of parallelized water quality analysis

Data Exchange between CPU and GPU

The memory address spaces of the CPU and the GPU code are different because the CPU and GPU memory are physically located on different chips. Hence we need to manually copy all input data to the GPU memory, then perform segment update computation, and bring the results back to CPU memory to continue the rest of the analysis.

In the original implementation of the water quality analysis, the segments for each pipe are stored in a list which grows or shrinks as needed. For instance, each pipe has a list of segments, and each segment contains the address of the next segment in the chain. Data at each segment can be accessed through a so-called 'pointer'—memory address of that segment. All segments in the system are stored in a list of lists, with one list for one pipe.

Due to the fact that GPU computation does not support pointer operation, it is the challenge if we want to copy these lists from CPU memory to GPU memory because the data represented in a pointer (i.e. memory address) in CPU memory is not meaningful in GPU memory. The only way to exchange the data is to copy the actual values instead of the memory addresses. Copying the values would require us to visit those lists by jumping from segment to segment in CPU

memory in order to copy the data to GPU memory. This overhead would certainly take away any speed-up that could be possibly gained from GPU-accelerated computation.

The solution we adopted was to arrange the segments in an array in CPU memory instead of a linked list. We modified the existing segment data structure so that instead of containing a memory address, it would contain an index in the array.

Data Structure for Arbitrary Number of Segments

Although the above solution fixed the address space issue, another challenge is that an array is a static data structure, and one needs to know the desired size of the array before creating it. However, the number of segments for each pipe at a given time step depends on the quality values across the pipe and various simulation variables, making it impossible to know the final number of segments before the simulation begins. Thus we needed to allow arbitrary number of segments in the system while using an array data structure which can contain only a pre-specified number of segments.

This problem was solved by using a dynamic collection of arrays. Each array, called a *segment bank*, can hold a large (but fixed) number of segments. When the system needed to allocate more segments than the current capacity, one more segment bank was added to the collection. Two similar and synchronized segment management schemes were implemented for WQ analysis using both CPU and GPU. This allowed us to store arbitrary number of segments in a collection of fixed-size arrays which could be copied back and forth between CPU and GPU memory.

TESTING EXAMPLES

Two example models based on real water systems were tested with the parallelized water quality analysis solver. The first example model contains 80,871 pipes, 11 tanks, 8 reservoirs and 22 pumps while the second example the BWSN network II [4] contains 14,822 pipes, 2 tanks and 4 pumps. Each of the models was set up to run water quality analysis with constant constituent input at the sources. Comprehensive comparisons in result accuracy and computation speedup are presented as follows.

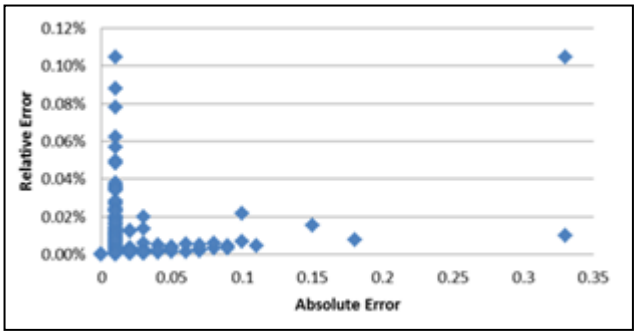

Figure 2. Comparison of water quality results obtained with conventional solver and the parallelized solver

Accuracy

In order to gain confidence on the GPU-based water quality analysis solver, the simulated water quality results by the accelerated solver are compared with those obtained with the original WQ solver executed on CPU. The example 1 was tested for the accuracy comparison. As shown in

Figure 2. The relative errors of less than 0.02% and the absolute error of less than 0.05 mg/L are achieved for most of nodes over 192-hour simulation. Only a few nodes are observed with the relative error of greater than 0.05% and the absolute error of greater than 0.1 mg/L. The maximum relative error of 0.1% was observed between the accelerated and the conventional solvers and the maximum absolute error of 0.35 mg/L at one time step at one node. The subtle difference is most likely caused by the single precision used in the implementation of the accelerated solver because of the double precision is not efficiently handled on GPU computing. However, it is believed that the computation is adequately accurate for WQ analysis.

Table 1. Testing machine configurations

	Machine I	Machine II
CPU cores	2	16
RAM	2 GB	20 GB
GPU Cores	32	448
GPU threads/block	512	1024
CPU/GPU IO bandwidth	Usual	Fast

Computation Performance

The accelerated solver is tested on two example models with two different machines, which have the configurations as in Table 1. Machine I represents a low-end machine while machine II represents a high-end machine. Both are used for testing the GPU-accelerated WQ solver to gauge the computation performance on the large models.

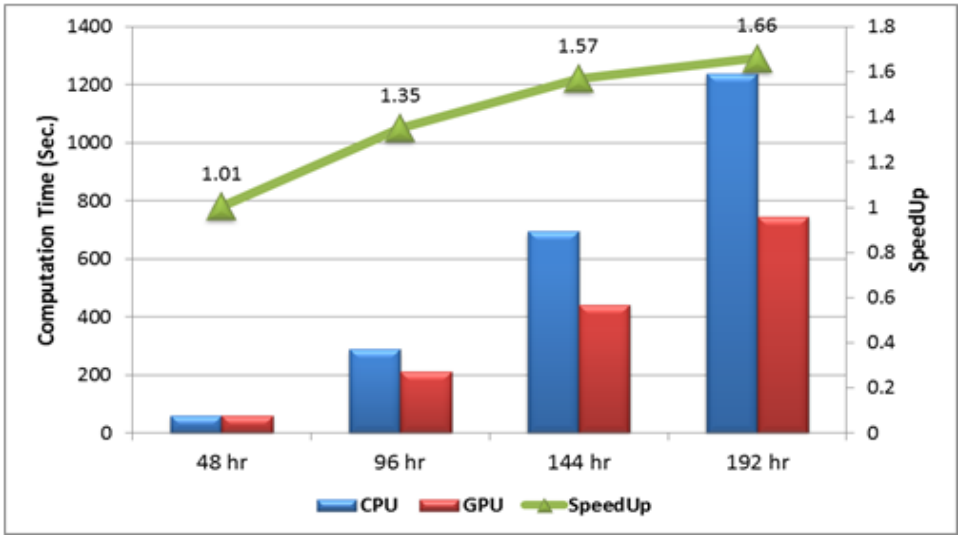

Figure 3. Computation performance of the parallelized water quality solver on machine I

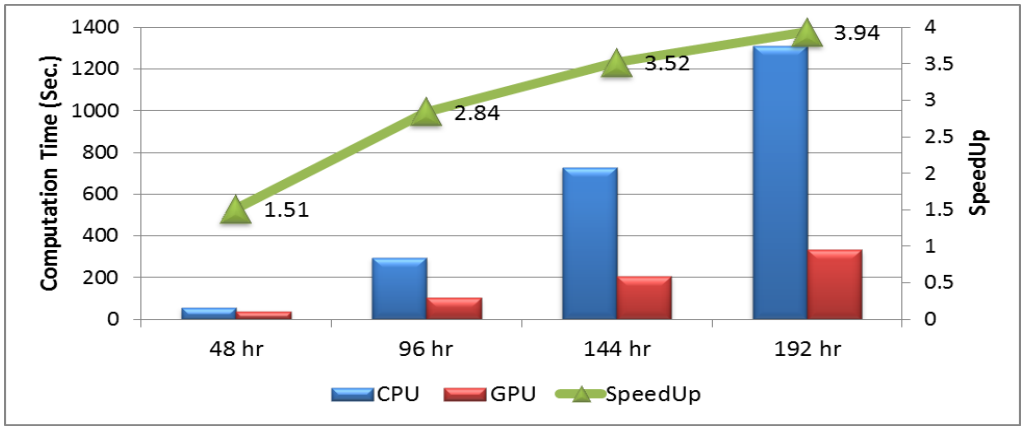

Figure 4. Computation performance of the parallelized water quality solver on machine II

Figure 3 illustrates the computation performance of the accelerated solver tested on example I using machine I. The simulation runs of different periods including 48 hours, 96 hours, 144 hours and 192 hours were undertaken with the GPU-accelerated and CPU-based solvers. It shows that the speedup increases as the simulation period increases. The maximum speed up of 1.66 was achieved for the 192-hour WQ simulation with the low-end machine. Figure 4 shows the computation performance comparison of the WQ solvers tested on example I using the high-end machine. The GPU-accelerated WQ solver is almost 4 times faster than the conventional CPU-based WQ solver on the high-end machine. Greater speed-up was obtained for longer simulations, and also with a more powerful GPU-enabled machine.

The WQ solvers were also tested on both example I and example II for 192-hour simulation using the high-end machine. Figure 5 shows the computation performance of two models. It indicates that the GPU-accelerated solver results in a greater speed up for the larger model.

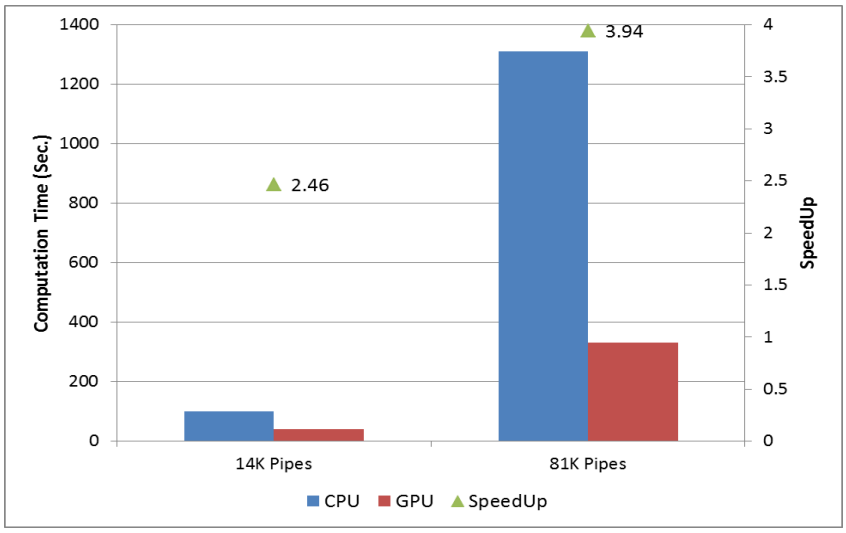

Figure 5. Computation performances of the original (CPU) and GPU-accelerated WQ solvers tested on two models using machine II

CONCLUSIONS

A GPU-accelerated water quality solver has been successfully developed by using the heterogeneous CPU-GPU computing paradigm. The accelerated solver has been tested on two large models using low-end and high-end machines. The testing results show that the accelerated solver is producing adequately accurate results as the original solver based on CPU. The parallelized solver is able to significantly speed up the computation for large models. The larger the model is, the greater the speedup is by the accelerated solver. The longer the simulation is, the greater the speedup is by the GPU-accelerated water quality solver. The more advanced the compute hardware is, the greater the speedup is. The accelerated solver has been implemented with OpenCL that ensures the portability of the solver across various computing hardware. As GPU and general accelerated computing unit is becoming widely available and ubiquitous for heterogeneous computing, the GPU-accelerated water quality solver will be essential at leveraging the computing power to improve the efficiency of the WQ analysis with large models, and enabling engineers to achieve greater productivity and better solution quality by using the accelerated modeling tool than the conventional tool.

REFERENCES

[1] Rossman L.A. "EPANET 2 user's manual", United States Environmental Protection Agency, Cincinnati, USA, (2000).

[2] Wu Z. Y., Walski T., Wang R.H., Boulder D. and Yang S. Y. "Extended Global Gradient Algorithm for Pressure Dependent Demand Analysis of Water Distribution Systems", ASCE *Journal of Water Resources Planning and Management*., Vol.135, No.1, (2009), pp13-22.

[3] Munshi A., Gaster B. R., Mattson T.G., Fung J. and Ginsburg D. "*OpenCL Programming Guide"*, Addison-Wesley, New York, USA, (2011)

[4] Ostfeld A. J, Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, F. di Pierro, S.-T. Khu, D. Savic; D. Eliades, M. Polycarpou, S. R. Ghimire, B. D. Barkdoll, R. Gueli, J. J. Huang, E. A. McBean, W. James, A. Krause, J. Leskovec, S. Isovitsch, J. Xu, C. Guestrin, J. VanBriesen, M. Small, P. Fischbeck, A. Preis1, M. Propato11, O. Piller, G. B. Trachtman, Z. Y. Wu and T. Walsk "Battle of the Water Sensor Networks (BWSN): A Design Challenge for Engineers and Algorithms." ASCE *Journal of Water Resources Planning Management*, Vol. 134, No.6, (2008), pp556-568.