Profiling and Evaluation of Implicit and Explicit Storm Surge Models

Abdullah Alghamdi and Muhammad Akbar

***Abstract*—Storm surge models are vital nowadays because the recent destruction caused by hurricanes. In this study, we are profiling and evaluating explicit and implicit solvers, two popular methods used in storm surge predictions. Hurricane Katrina storm surge hindcast has been used in this simulation for both methods. We use analysis tools to examine both methods from various perspectives including: execution times, computational intensity, characterization of instructions, cache memory misses and others. Many experiments have been performed with various number of processors. We did not only present the differences between methods, but also we addressed the challenges associated with these methods.**

***Index Terms*—Storm surge, implicit and explicit solvers, parallelization, and profiling.**

I. Introduction

Hurricanes are one of the most sophisticated phenomena, and it is evident that the disasters caused by hurricanes are affecting millions of people and cost a lot of money. Therefore, the need for precise, fast, and reliable models that are capable of predicating storm surges, floods, and levee overtopping is inevitable. An enormous amount of previous work is being done in this regard, resulting in computational models which solves the Shallow Water Equations (SWEs) for predication of water elevation and velocity. There are many variables that have been taken in consideration to formulate the phenomena mathematically, resulting in accurate simulation and prediction. These variables include: acceleration, wind stress, pressure, the Coriolis force due to planetary rotation, waves that ride the circulation, and many other potential parameters [1]. The algorithms used to solve the SWEs equations depends on explicit [1], implicit [2], or semi-implicit [1] methods.

In this study, we are profiling and evaluating the performance of explicit and implicit solvers from various perspectives including: execution time, computational intensity, MIPS and MFLOPS, cache memory miss rates, TLB misses, IO and others. We are considering Advanced Circulation Model (ADCIRC) [3] and the Computation and Modeling Engineering Laboratory (CaMEL) [2] as models for the evaluation of explicit and implicit solvers, respectively.

The rest of this paper is organized as the following: Section II discusses the related work regarding ADCIRC and CaMEL, section III discuss the methodology used in this study, Section IV presents the found results followed by the challenges in Section V, and conclusion in Section VI.

Manuscript received June 1, 2017; revised November 24, 2017.

A. Alghamdi is with College of Engineering at Tennessee State University, Nashville, TN 37209 USA (e-mail: aalghamd9@ my.tnstate.edu).

M. Akbar is with Mechanical and Manufacturing Engineering Department at Tennessee State University, Nashville, TN 37209 USA (e-mail: makbar@tnstate.edu).

II. Related Work

At the beginning, ADCIRC was developed by Luettich *et al.* [4] and Luettich and Westerink [1]; however, it has been improved throughout the years by many researchers [5]-[9]. In ADCIRC, the water elevation and velocity are obtained by solving the depth-integrated continuity equation in Generalized Wave-Continuity Equation (GWCE), and the momentum equations in its 2DDI or 3D forms, respectively. On the other hand, CaMEL is developed recently by Akbar and Alibadi [2] and Aliabadi *et al.* [10]. CaMEL uses hybrid finite element and finite volume techniques to solve the conservative equations implicitly with the focus of being numerically stable. Both models are written in Fortran, and both are using METIS [11] partitioning library; however, versions are different.

III. Methodology

In this study, we are measuring the performance of both models from various aspects including: execution time, cache misses, millions of instructions per second (MIPS) and mega floating-point operations per second (MFLOPS) and others with various number of processors. To achieve the consistency, we are using the same storm surge hindcast, Hurricane Katrina with a mesh that consists of 254,565 nodes and 492,179 elements [12]. This mesh covers half of the Atlantic Ocean and the entire Gulf of Mexico. HPCtoolkit [13] with Performance Application Programming Interface (PAPI) [14] has been linked dynamically to both models to collect data during the execution for profiling and evaluation. Our experiments ranged from one processor up to 256 processors. We ran our experiments on a cluster formed of Dell blade servers that have two 8-core Intel Xeon E5-2400 processors and 96GB of RAM, and it is backed by an InfiniBand FDR/Ethernet 10/40GB interconnect [15]. For the results below, we ran the full simulation of Hurricane Katrina hindcast for each model, unless otherwise stated. Finally, it is worth noting that HPCtoolkit overhead is 1-5% [13].

IV. The Results

Even though the two models are using completely different

DOI: 10.7763/IJCTE.2017.V9.1178

techniques, we showed that very few mismatches, if any, are visible between the two results. Fig. 1 shows the comparison between ADCIRC and CaMEL solvers for Hurricane Katrina near its peak point. One of the most important differences is the time step. ADCIRC explicit solver uses 4 seconds as its maximum time step to run successfully, while CaMEL can go far beyond that with its capability to run successfully with 100 seconds. Although CaMEL implicit solver is far stable than ADCIRC explicit solver, CaMEL is slower than ADCIRC.

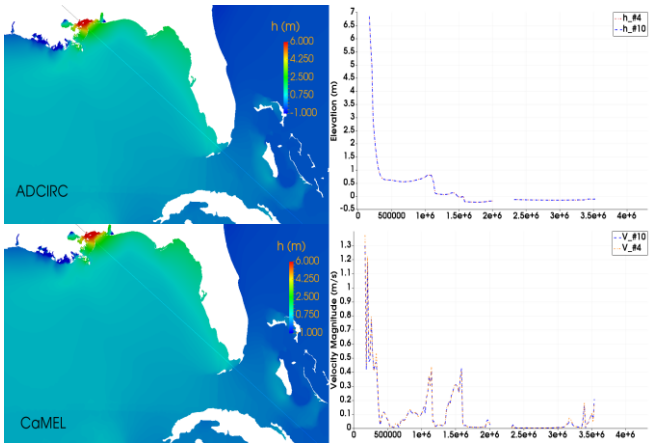

Fig. 1. Comparison between ADCIRC and CaMEL.

In the following subsections, we will start by presenting our findings at the execution time followed by the computational intensity. Then, characterization of instructions, and cache memory misses and memory and IO operations will come after that.

A. Execution Time

In our work, we classified the models' subroutines and statements into three categories. First, MPI category, which includes all the statements invoked for the sake of parallelization using Message-Passing Interface (MPI) library. Second, APP category, which includes all the core statements and subroutines that are concerned with the calculation of water elevation and velocity and IO statements of the model. Third, OTHERS category, which includes memory allocation/deallocation, resource reservation, preparatory steps, and any other statement that do not fit the previous categories. Fig. 2 shows our findings of ADCIRC explicit and CaMEL implicit solvers plotted against the number of processors used in every experiment. The presented findings are in seconds, and the total number of time steps is 70,000 time-steps for both models to assure that both models undergo the same conditions.

It is obvious that the CaMEL implicit solver is much slower than ADCIRC explicit solver. As expected, the APP time gets decreased significantly whenever we double the number of processors. Also, we notice that the time being spent on the OTHERS category is negligible, especially when we increase the number of processors. Observing the time of MPI increasing whenever we increase the number of processors was anticipated. We notice that MPI time of implicit solver is much higher than when we use explicit solver. This result could be correlated to the fact that the implicit solver communicates the results during every time step with all the processors; however, that is not the only reason. ADCIRC and CaMEL programming and parallelization are different. Noticing the performances of ADCIRC and CaMEL serial codes, where there are no processor-to-processor communications, indicates the huge difference between the two models.

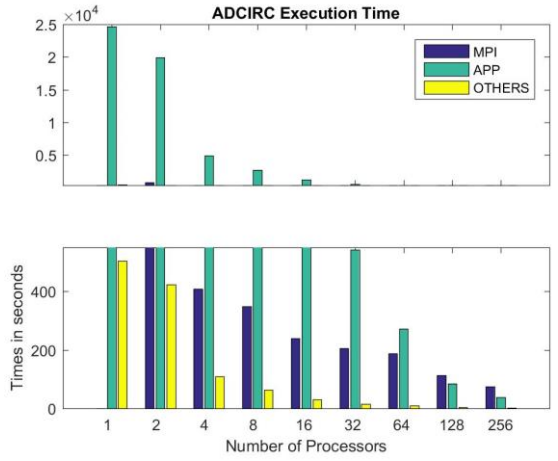

Fig. 2a. Execution time for ADCIRC model.

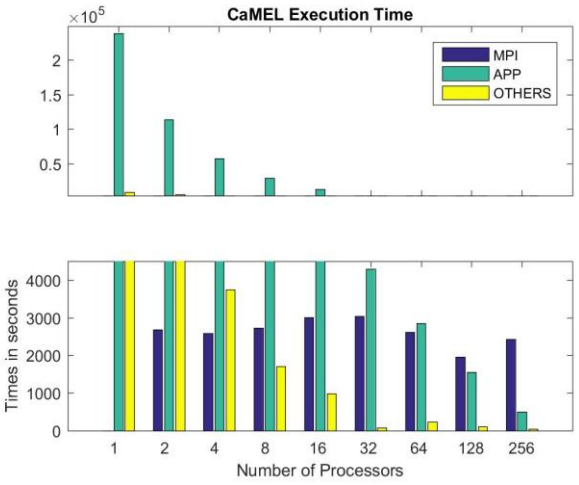

Fig. 2b. Execution time for CaMEL model.

B. Computational Intensity

After finding that the serial program in CaMEL takes much more time than ADCIRC serial program, we decided to calculate the computational intensity for both programs. Fig. 3 shows the results of our experiments. It is indisputable that ADCIRC executes twice more instructions as CaMEL executes per cycle, given the number of processors is 128 and 256. Also, whenever we increase the number of processors, ADCIRC computational intensity is increasing, especially for 128 and 256 processors, while CaMEL is not gaining much of its parallelization on this aspect.

Also, we investigated the number of CPU cycles that has no instruction issued, and the difference was interesting. The difference in ADCIRC for each experiment with a different number of processors was imperceptible as presented in Fig. 4. On the other hand, CaMEL has significantly increasing the number of CPU cycles with no instruction issued which

inhibited the speedup of the program. One reason for such result is the communication intensity between processors where the processors will spend so much time waiting for others to communicate; however, that could not be the only reason.

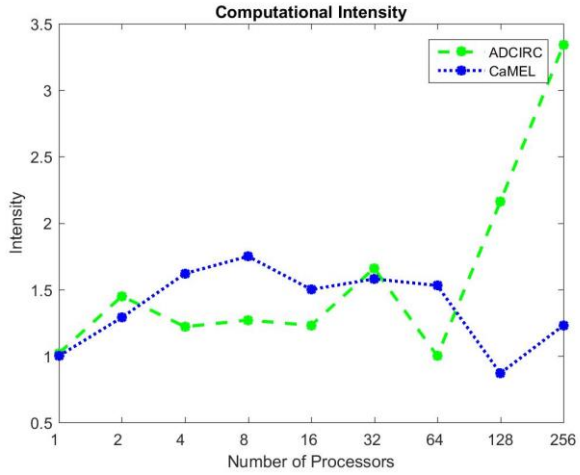

Fig. 3. Computational Intensity for both models.

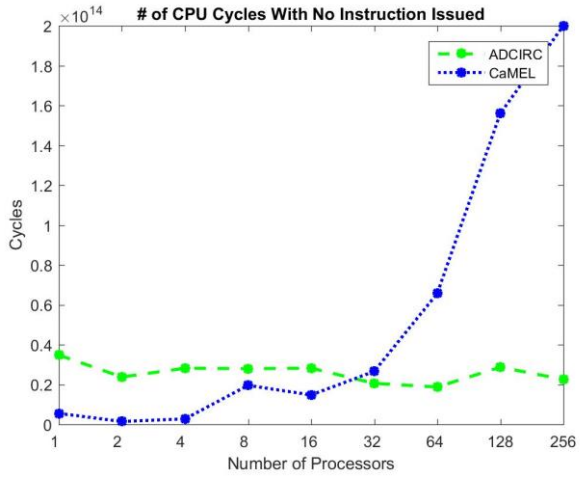

Fig. 4. The number of CPU cycles with no instruction issued.

C. Characterization of Instructions

In order to study the instructions being executed by both models, the instructions have been categorized into four classes: floating-point instructions (FL) which represents all the arithmetic operations over floating-points numbers, loading data instructions (LD) which represents instructions issued to fetch a piece of data from the memory, storing data instructions (SR) which represents the instructions to store a piece of data at the memory, branch instruction (BR) which represents all the jump instructions to execute a different sequence of instructions, and OTHERS which represents all the instructions that do not belong to the previous categories.

Because of the similarity between the experiments and to avoid the repetition, Fig. 5 shows only the results of instructions classification based on experiments for 2, 32, 256 processors. The LD instructions, which consume great deal of time, are decreasing whenever we increase the number of processors. Nonetheless, CaMEL's LD instructions are much higher comparing to ADCIRC for 256 processors. Because the floating-point division is a time-consuming instruction [16], it has been studied and we notice that around 9% of the FL instructions are division instructions for ADCIRC. On the other hand, CaMEL has only around 5% of the FL instruction as division instructions; however, when we compare the actual number of ADCIRC's division instructions and CaMEL's division instructions, CaMEL has 4.41E+13 instructions, which is significantly higher number comparing to ADCIRC that has only 4.94E+12 instructions.

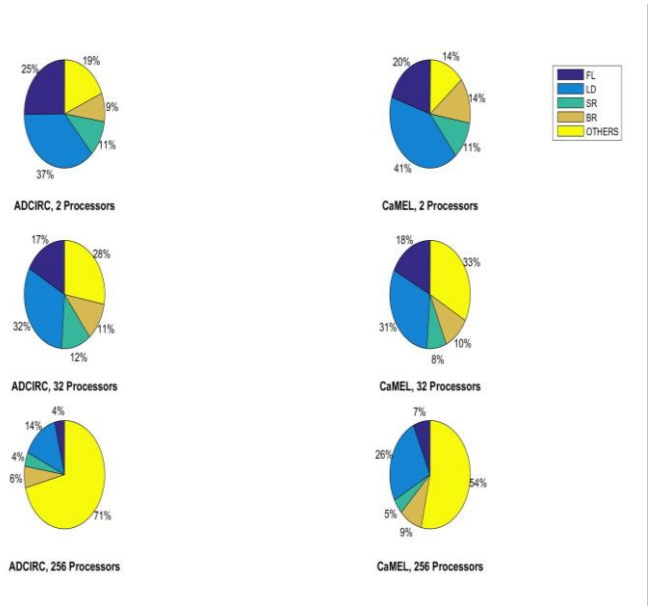

Fig. 5. Instruction characterization for both models.

D. MIPS and MFLOPS

MIPS and MFLOPS are commonly used metrics in programs and computer architecture evaluations. In our study, we show the aggregate MIPS and MFLOPS for both models in Fig. 6. Both models have a very similar starting point, and it increases when the number of processors get increased. For MIPS, ADCIRC seems to gain much benefit of its parallelization by having sharper growth compared to CaMEL, especially with 128 and 256 processors. On the other hand, both models have a similar growth on MFLOPS, with the exception that CaMEL is slightly better with 64 and 128 processors.

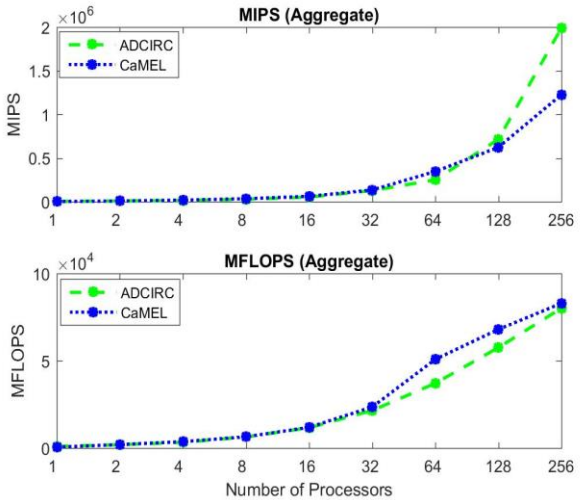

Fig. 6. The aggregate MIPS and MFLOPS for both models.

E. Cache Memory Misses

Cache misses are a time-consuming process due to the long latency associated with accessing the main memory after failing to read or write to the cache. In our study, we investigated the miss rate for both cache memories: level two (L2) and level three (L3), and misses includes both data misses and instructions misses. In Fig. 7, we show that CaMEL has an enormous amount of L2 and L3 misses compared to ADCIRC. This result explains another reason for CaMEL's slowness due to its low capability of reusing data and instructions brought the cache. On the other hand, ADCIRC has a steady cache miss rate. Moreover, examining the table lookaside buffer (TLB) miss rates for both models shows that both models has a similar starting point; however, the result varies extremely when the number of processors increase. As shown in Fig. 8, CaMEL has a significant TLB misses for 64, 128, and 256 processors which indicates that CaMEL is not optimized properly.

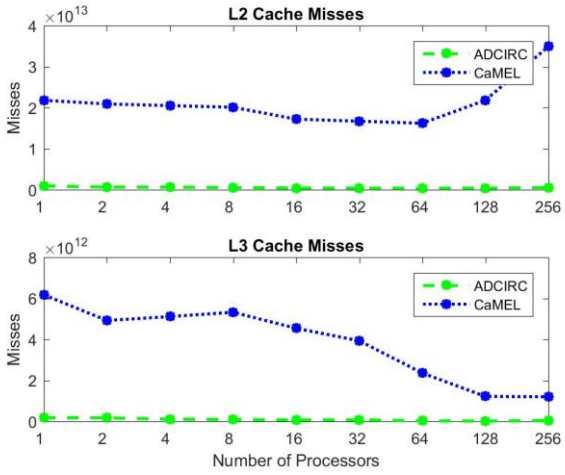

Fig. 7. The number of L2 and L3 cache misses for both models.

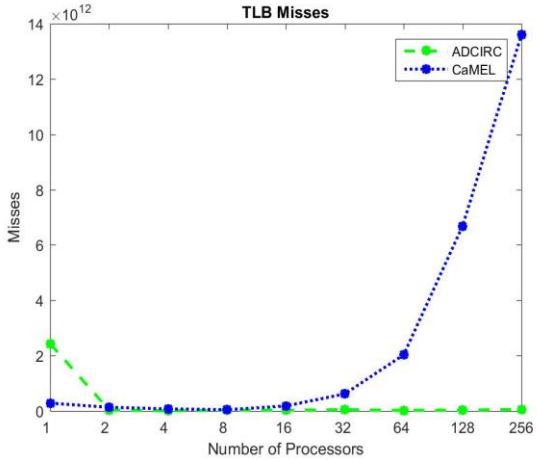

Fig. 8. The number of TLB misses for both models.

F. Memory and IO

Although the effect of memory allocations and deallocation and reading input and writing output on models performances is not prominent in our experiments, we examined these factors too. Our results show that ADCIRC reads slightly less input files than CaMEL because there are some specific parameters required by CaMEL [2]. For writing output and memory allocation and deallocation, ADCIRC seems to be doing much more than what CaMEL is doing as shown in Fig. 9, and yet the memory allocation and deallocation and IO operations are not significant compared to other operations.

V. Challenges

This research endured some challenges along the way. One is the limitation that we had regarding hardware counters availability. Many hardware counters are not available in our experimental environment and it couldn't be derived such as: cache memory misses for level one (L1) counter, floating-point square root, multiplication, and inverse operations counters, synchronized instructions counters and others.

Because ADCIRC and CaMEL are programmed and parallelized differently, the comparison between the explicit and implicit storm surge models are not completely precise. The less optimization that CaMEL has made the comparison discouraged. Developing and evaluating implicit and explicit solvers that have been developed for the same model would have a significant impact on storm surge modeling.

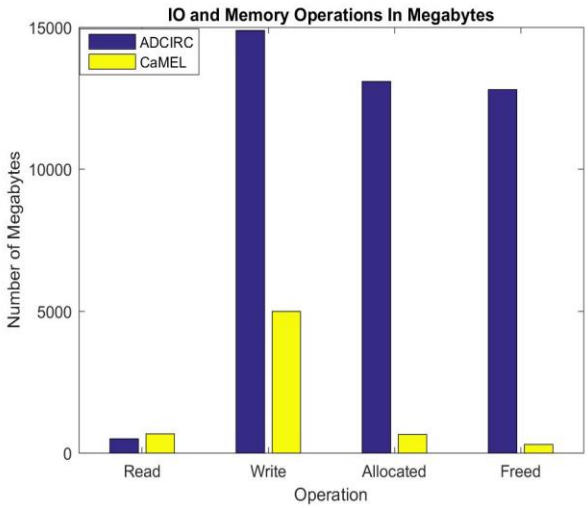

Fig. 9. The number of IO and memory operations in megabytes for both models.

VI. Conclusion

In this study, we examine the storm surge explicit and implicit solvers from various perspective in a comparative study. ADCIRC and CaMEL models are chosen to represent the explicit and implicit solvers, respectively. Hurricane Katrina storm surge hindcast is used in all the experiments that we did. The models are dynamically linked to HPCtoolkit for measurement. We found that there are many factors that contribute to the slowness of the implicit solver including: intensive MPI communication, time-consuming instructions, cache memory misses and TLB misses. The optimization of CaMEL is an obstacle to conduct a precise comparative study between the explicit and implicit solvers. Placing the explicit and implicit solvers on the same programming and parallelization architecture would have a significant impact on storm surge modeling.

ACKNOWLEDGMENT

The authors would like to thank Renaissance Computing Institute (RENCI) for providing the access to high performance computing platform. Special thanks go to scholarship program at Najran University, Saudi Arabia.

REFERENCES

[1] R. Albert Luettich and J. J. Westerink, *Formulation and Numerical Implementation of the 2D/3D ADCIRC Finite Element model R. Luettich*, 2004.
[2] M. Akbar and S. Aliabadi, "Hybrid numerical methods to solve shallow water equations for hurricane induced storm surge modeling," *Environmental Modelling & Software*, vol. 46, pp. 118-128, 2013.
[3] J. J. Westerink, R. A. Luettich Jr, C. A. Blain, and N. W. Scheffner, "ADCIRC: An advanced three-dimensional circulation model for shelves, coasts, and estuaries," Report 2. User's Manual for ADCIRC-2DDI. No. WES/TR/DRP-92-6-2. Army Engineer Waterways Experiment Station Vicksburg MS, 1994.
[4] R. A. Luettich, J. J. Westerink, and N. W. Scheffner, "ADCIRC: An advanced three-dimensional circulation model for shelves, coasts, and estuaries," Technical Report 1, Department of the Army, US Army Corps of Engineers, Washington, DC 20314-1000, 1991.
[5] J. H. Atkinson, J. J. Westerink, and J. M. Hervouet, "Similarities between the quasi‐bubble and the generalized wave continuity equation solutions to the shallow water equations," *International Journal for Numerical Methods in Fluids*, vol. 45, no. 7, pp. 689-714, 2004.
[6] C. Dawson, J. J. Westerink, J. C. Feyen, and D. Pothina, "Continuous, discontinuous and coupled discontinuous–Continuous Galerkin finite element methods for the shallow water equations," *International Journal for Numerical Methods in Fluids*, vol. 52, no. 1, 63-88, 2006.
[7] J. J. Westerink, R. A. Luettich, J. C. Feyen, J. H. Atkinson, C. Dawson, H. J. Roberts, M. D. Powell, J. P. Dunion, E. J. Kubatko, and H. Pourtaheri, "A basin-to channel-scale unstructured grid hurricane storm surge model applied to southern Louisiana," *Monthly Weather Review*, vol. 136, no. 3, 833-864, 2008.
[8] J. C. Dietrich, M. Zijlema, J. J. Westerink, L. H. Holthuijsen, C. Dawson, R. A. Luettich, R. E. Jensen, J. M. Smith, G. S. Stelling, and G. W. Stone. "Modeling hurricane waves and storm surge using integrally-coupled, scalable computations," *Coastal Engineering*, vol. 58, no. 1, pp. 45-65, 2011.
[9] S. Bunya, J. C. Dietrich, J. J. Westerink, B. A. Ebersole, J. M. Smith, J. H. Atkinson, R. Jensen *et al.*, "A high-resolution coupled riverine flow, tide, wind, wind wave, and storm surge model for southern Louisiana and Mississippi. Part I: Model development and validation," *Monthly Weather Review*, vol. 138, no. 2, pp. 345-377, 2010.
[10] S. Aliabadi, M. Akbar, and R. Patel, "Hybrid finite element/volume method for shallow water equations," *International Journal for Numerical Methods in Engineering*, vol. 83, no. 13, 1719-1738, 2010.
[11] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*,vol. 20, no. 1, pp. 359-392, 1998.
[12] A. Y. Mukai, J. J. Westerink, and R. A. Luettich, "Guidelines for using Eastcoast 2001 database of tidal constituents within Western north Atlantic Ocean, Gulf of Mexico and Caribbean sea," No. ERDC/CHL-CHETN-IV-40. Engineer Research and Development Center Vicksburg MS Coastal and Hydraulics LAB, 2002.
[13] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor‐Crummey, and N. R. Tallent, "HPCToolkit: Tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, 685-701, 2010.
[14] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "PAPI: A portable interface to hardware performance counters," in *Proc. the Department of Defense HPCMP Users Group Conference*,1999, vol. 710.
[15] Predicting hurricane storm surge and waves precisely. [Online]. Available: http://renci.org/wp-content/uploads/2014/08/2014_RENCI_Dell-CaseStudy.pdf
[16] M. Abd-El-Barr and H. El-Rewini, *Fundamentals of computer Organization and Architecture*, vol. 38, John Wiley & Sons, 2005.

Abdullah A. Alghamdi currently is a Ph.D. student at Tennessee State University (TSU), College of Engineering, and team member of High Performance Computing for Global Challenges (HPCGC) laboratory at TSU. He got his masters from Rochester Institute of Technology, Rochester, NY, 2013 in networking and systems administration. Previously, he worked as teaching assistant, and lecturer at Najran University, Najran, Saudi Arabia. Currently, he is working on developing a fully implicit solver for ADCIRC storm surge model, where we try to increase the stability, and keep up the parallelization of model. Also, he is interested in any multidisciplinary applications where his skills and knowledge can be applied to address global challenges.

Muhammad K. Akbar is an assistant professor at Mechanical and Manufacturing Engineering Department at Tennessee State University (TSU) and the leader of High Performance Computing for Global Challenges (HPCGC) laboratory. Previously, he served as senior research associate at Jackson State University, visiting assistant professor at Georgia Institute of Technology. Also, he was graduate research assistant at Georgia Institute of Technology and University of Alabama.

He has received many funds as PI and CO-PI from NSF, DHS and Air Force Research Laboratory. His research interests are computational fluid Dynamics; thermal fluids, hurricane storm surge model development; porous media; gas turbine film cooling; catalyticl converter simulation; multiphase flow; particulate flow; microchannel / minichannel flow; MPI based parallel model development; delayed detached eddy simulation turbulence model development; Ansys fluent based modeling; fluid structure interaction; solar energy.