Performance Evaluation of a Two-Dimensional Flood Model on Heterogeneous High-Performance Computing Architectures

Md Bulbul Sharif
Tennessee Technological University
Cookeville, Tennessee, USA
msharif42@students.tntech.edu

Sheikh K. Ghafoor
Tennessee Technological University
Cookeville, Tennessee, USA
sghafoor@tntech.edu

Thomas M. Hines
Tennessee Technological University
Cookeville, Tennessee, USA
tmhines42@students.tntech.edu

Mario Morales-Hernández
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
moraleshernm@ornl.gov

Katherine J. Evans
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
evanskj@ornl.gov

Shih-Chieh Kao
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
kaos@ornl.gov

Alfred J. Kalyanapu
Tennessee Technological University
Cookeville, Tennessee, USA
akalyanapu@tntech.edu

Tigstu T. Dullo
Tennessee Technological University
Cookeville, Tennessee, USA
ttdullo42@students.tntech.edu

Sudershan Gangrade
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
gangrades@ornl.gov

ABSTRACT

This paper describes the implementation of a two-dimensional hydrodynamic flood model with two different numerical schemes on heterogeneous high-performance computing architectures. Both schemes were able to solve the nonlinear hyperbolic shallow water equations using an explicit upwind first-order approach on finite differences and finite volumes, respectively, and were conducted using MPI and CUDA. Four different test cases were simulated on the Summit supercomputer at Oak Ridge National Laboratory. Both numerical schemes scaled up to 128 nodes (768 GPUs) with a maximum 98.2x speedup of over 1 GPU. The lowest run time for the 10 day Hurricane Harvey event simulation at 5 meter resolution (272 million grid cells) was 50 minutes. GPUDirect communication proved to be more convenient than the standard communication strategy. Both strong and weak scaling are shown.

CCS CONCEPTS

• **Computing methodologies** → Massively parallel and high-performance simulations.

KEYWORDS

2D flood model, flood simulation, GPU programming, CUDA, high-performance computing, Multi-GPU

ACM Reference Format:
Md Bulbul Sharif, Sheikh K. Ghafoor, Thomas M. Hines, Mario Morales-Hernández, Katherine J. Evans, Shih-Chieh Kao, Alfred J. Kalyanapu, Tigstu T. Dullo, and Sudershan Gangrade. 2020. Performance Evaluation of a Two-Dimensional Flood Model on Heterogeneous High-Performance Computing Architectures. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '20), June 29-July 1, 2020, Geneva, Switzerland.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394277.3401852

PASC '20, June 29-July 1, 2020, Geneva, Switzerland

ACM ISBN 978-1-4503-7993-9/20/06…$15.00
https://doi.org/10.1145/3394277.3401852

1 INTRODUCTION

More than 21% of the global population live within 30 km of a coastline [13] and are at risk of being affected by flooding. In 2017, the record-breaking Hurricane Harvey caused $125 billion in property damage in the Houston metropolitan area and Southeast Texas [5] [20]. Hydrodynamic flood models have been applied to simulate floods for more than 50 years [6], but with the increase in computational capabilities, more accurate and rapid hydraulic models have emerged that can provide a more reliable basis for decision-making in flood risk management.

Traditionally, 1D flood models, such as the Hydrologic Engineering Center's River Analysis System (HEC-RAS) [3] or MIKE 11 [9], have been used, which are computationally less expensive in terms of total run time. However, 1D models, especially in urban flood-prone areas, cannot properly represent flood wave propagation over floodplains in events that exceed the predefined 1D river channels [35]. On the other hand, 2D models, such as HEC-RAS 2D [4] or SRH-2D [23], eliminate the primary limitations of 1D models by allowing topographical representation of higher-order and preferential flood pathways [35]. The main drawback of 2D models is their computational burden, and the simulation of flooding over a large area at high resolution in a reasonable amount of time requires high-performance computing (HPC) [24] [18]. Current large graphics processing unit (GPU)–based systems such as Summit [32] can simulate flooding at high resolution. Such simulation requires that the code be scalable over multiple GPUs in a distributed, heterogeneous computing environment. Few studies in the literature have reported on using multiple GPUs to solve shallow water equations [8], [39], [41] or to simulate floods or tsunamis [28], [33], [26] and [7]. This paper describes one of the few studies in which multiple GPUs were used to simulate a 2D flood model.

One of the objectives of our current research is to develop a performance portable architecture agnostic software framework for regularly structured grid-based simulation applications. The 2D flood simulation application is the first step toward that goal. We have implemented the 2D flood model using MPI, OpenMP, and CUDA for execution on a single node (using OpenMP), multiple nodes (using MPI + OpenMP), or on multiple nodes with GPU (using MPI + OpenMP + GPU). In this paper, we evaluate our implementation on Summit using multiple nodes with MPI and CUDA.

The contributions of this study include (1) implementation of a 2D flood simulation model using two different numerical schemes in a heterogeneous multi-GPU HPC environment, (2) simulation of a large-scale 10 day Hurricane Harvey flood event at 5 meter resolution (272 million grid cells) on Summit with up to 128 nodes (768 GPUs), (3) demonstrating the possibility of large-scale, high-resolution flood simulation, (4) identifying both strong and weak scaling with multiple GPUs, and (5) performance analysis and comparison of two different numerical schemes and GPUDirect [31] acceleration technology.

2 RELATED WORKS

Although GPU computing has been incorporated into a wide range of computational and modeling applications with success, it has found limited application in computational fluid dynamics and flood modeling. In 1966, the Stanford watershed model IV [6], now known as HSPF, was first introduced and applied computer modeling to simulate floods. In [17], a real-time visual simulation of diverse dynamic phenomena using a GPU was presented and demonstrated a speedup of 25x on an NVIDIA GeForce 4 as compared with the same simulation run on a Pentium CPU. A visual simulation study of shallow water waves using a GPU [16] reported a speedup of 15x to 30x compared with a CPU simulation. A speedup of 112x using a diffusive wave flood modeling approach was compared with a CPU model in [24]. A study of the computational enhancement of a GPU-enabled 2D flood model was described in [19] in which the researchers achieved an 88x speedup on a Tesla C1060 GPU. An efficient implementation of a state-of-the-art high-resolution explicit scheme for shallow water equations on a GPU was reported in [2] and demonstrated computation of the first 4000 s of the Malpasset dam-break case in 27 s. Additionally, the implementation of a finite volume method to solve 2D shallow water equations on a GPU was described in [22]. The strategy was designed to work efficiently with unstructured meshes, which are widely used in many fields of engineering. A shallow water equations parallel numerical scheme suitable for a GPU described in [38] claimed to achieve speedups two orders of magnitude over a single-core CPU. A few years later, the same authors [37] also used a GPU-parallel numerical model to solve 2D shallow water equations based on Block-Uniform Quadtree (BUQ). They claimed that a very large domain can be simulated (38 km river reach) with high resolution (2 m). Other researchers used a GPU for their flood modeling and reported good improvement in performance compared with a CPU [34], [1], [25] and [15].

One of the major drawbacks of the studies mentioned above is that they are limited to a single GPU. A state-of-the-art shallow water simulator running on multiple GPUs was described in [33]. The simulator showed a rate of over 1.2 Gigacells per second using four Fermi-generation GPUs. In [7] the authors proposed a 2D numerical scheme to simulate tsunamis generated by landslides. They claimed good weak and strong scaling using up to 24 GPUs in real and artificial problems. In another study [26], researchers tried to present a practical implementation of a 2D flood simulation model using hybrid distributed-parallel technologies including MPI, OpenMP, and OpenCL. They used a maximum of 8 GPUs for their experiments and reported a maximum performance of 3 Gigacells per second.

3 NUMERICAL SCHEMES

Two different numerical schemes are compared in this study: a finite difference (FD) scheme and a finite volume (FV) scheme. They are described in this section.

3.1 FD Scheme

The first numerical algorithm used in this model is an upwind finite difference scheme that solves nonlinear hyperbolic shallow water equations using a first-order accurate scheme. These equations are derived from the Navier-Stokes equations by integrating the horizontal momentum and continuity equations over a depth that is often referred to as the depth-averaged or depth-integrated shallow water equations. The non-conservative form of the partial differential equations [36] [18] are as follows:

Continuity equation

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} + \frac{\partial vh}{\partial y} = 0 \,, \tag{1}$$

Momentum equation in x-direction

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial H}{\partial x} + gS_{fx} = 0 \,, \tag{2}$$

Momentum equation in y-direction

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial H}{\partial y} + gS_{fy} = 0 \,, \tag{3}$$

where h is the water depth, H is the water surface elevation, u is the velocity in the x-direction, v is the velocity in the y-direction, t is the time, g is the acceleration due to gravity, S_{fx} is the friction slope in the x-direction, and S_{fy} is the friction slope in the y-direction. An upwind finite difference scheme is used to discretize the governing equations (1)–(3) following [19]. Friction terms S_{fx} and S_{fy} are computed as

$$S_{fx} = n^2(u_{ij})\sqrt{\frac{u_{ij}^2 + \bar{v}_{ij}^2}{h_{ij} + h_{(i+1)j}}} \,, \tag{4}$$

$$S_{fy} = n^2(v_{ij})\sqrt{\frac{v_{ij}^2 + \bar{u}_{ij}^2}{h_{ij} + h_{i(j+1)}}} \,, \tag{5}$$

$$\bar{u}_{ij} = \frac{u_{ij} + u_{i(j-1)} + u_{(i-1)(j-1)} + u_{(i-1)j}}{4} \,, \tag{6}$$

$$\bar{v}_{ij} = \frac{v_{ij} + v_{i(j+1)} + v_{(i+1)(j+1)} + v_{(i+1)j}}{4} \,, \tag{7}$$

where n is the Manning's roughness coefficient. The time step size Δt is restricted according to the Courant-Friedrich-Lewy (CFL) condition

$$\Delta x_i = \frac{dx}{|u|_i + \sqrt{g(h_i + \epsilon)}}\,, \tag{8}$$

$$\Delta y_i = \frac{dy}{|v|_i + \sqrt{g(h_i + \epsilon)}}\,, \tag{9}$$

$$\Delta t = \mathrm{CFL} \min_i(\Delta x_i, \Delta y_i) \qquad \mathrm{CFL} = 0.1\,, \tag{10}$$

where CFL represents Courant value and $\epsilon = 0.001$. This scheme has been tested and validated in [19].

3.2 FV Scheme

The second numerical scheme is also based on 2D shallow water equations but in a conservative differential form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S_b} + \mathbf{S_f}, \tag{11}$$

$$\mathbf{U} = \begin{pmatrix} h \\ q_x \\ q_y \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} q_x \\ \frac{q_x^2}{h} + \frac{1}{2}gh^2 \\ \frac{q_x q_y}{h} \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} q_y \\ \frac{q_x q_y}{h} \\ \frac{q_y^2}{h} + \frac{1}{2}gh^2 \end{pmatrix}$$
$$\mathbf{S_b} = \begin{pmatrix} 0 \\ -gh\frac{\partial z}{\partial x} \\ -gh\frac{\partial z}{\partial y} \end{pmatrix} \quad \mathbf{S_f} = \begin{pmatrix} 0 \\ -\frac{gn^2}{h^{7/3}} q_x \sqrt{q_x^2 + q_y^2} \\ -\frac{gn^2}{h^{7/3}} q_y \sqrt{q_x^2 + q_y^2} \end{pmatrix}, \tag{12}$$

where $\mathbf{U}$ is the vector of conserved variables including the water depth, h, and the unit discharges in x and y directions, called q_x and q_y, respectively. Moreover, $\mathbf{F}$ and $\mathbf{G}$ are the vector of fluxes, and $\mathbf{S_b}$ and $\mathbf{S_f}$ contain the bed and roughness source terms, respectively, with z being the bed elevation. Again, roughness is modeled using Manning-Gauckler's law.

In this scheme, a finite upwind volume explicit system is implemented based on the Roe's linearization. A squared mesh is used, denoting Δx as the grid spacing. The derivation of the numerical scheme follows [29], [27] for the fluxes and bed slope source terms, including the correct estimation of bed slope source terms at each edge. Nevertheless, a different discretization for the roughness terms is considered that follows [42], in which an implicit formulation is chosen. Therefore, a two-step algorithm is proposed for the update of a cell i from time t^n to time t^{n+1}:

$$\mathbf{U}_i^\star = \mathbf{U}_i^n - \underbrace{\frac{\Delta t}{\Delta x} \sum_{k=1}^{4} \sum_{m=1}^{3} \left[(\tilde{\lambda}\tilde{\alpha} - \tilde{\beta}_b)\tilde{\mathbf{e}} \right]_{m,k}^n}_{(\#)} \tag{13}$$

$$\mathbf{U}_i^{n+1} = \mathcal{F}(\mathbf{U}_i^n, \mathbf{U}_i^\star), \tag{14}$$

where $\tilde{\alpha}$ and $\tilde{\beta_b}$ are the fluxes and slope source term linearizations and $\tilde{\lambda}$ and $\tilde{\mathbf{e}}$ are the eigenvalues and eigenvectors of the system of equations, respectively. On the other hand, $\mathcal{F}$ is defined as follows:

$$\begin{aligned} \mathcal{F}^1 &= h^\star, \\ \mathcal{F}^2 &= -(q_x{}^\star)\left(\frac{1 - \sqrt{1 + 4S_f}}{2S_f}\right), \\ \mathcal{F}^3 &= -(q_y{}^\star)\left(\frac{1 - \sqrt{1 + 4S_f}}{2S_f}\right), \end{aligned} \tag{15}$$

where

$$S_f = \frac{\Delta t g n^2 \sqrt{(q_x{}^\star)^2 + (q_y{}^\star)^2}}{(h^n)^{7/3}}. \tag{16}$$

Being an explicit scheme, the time step size Δt is restricted again by the CFL condition:

$$\Delta t = \mathrm{CFL} \frac{\Delta x}{\max_i \left\{ \left|\frac{q_x}{h}\right|_i + \sqrt{gh_i}, \left|\frac{q_y}{h}\right|_i + \sqrt{gh_i} \right\}} \qquad \mathrm{CFL} \le 0.5 \tag{17}$$

More details on the numerical scheme can be found in [29], [27], and [42]. The validation of accuracy can also be found in [28].

4 IMPLEMENTATION

The general flowchart of our implementation is shown in Figure 1. Parallelization is done with MPI, with as many ranks as there are GPUs.

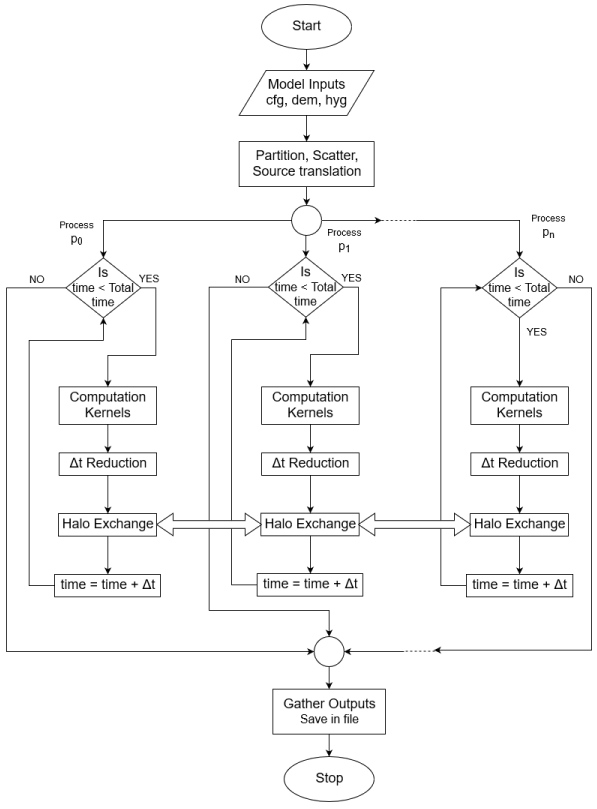

Figure 1: Parallel flowchart

4.1 Input

There are three types of data associated with our model.

- Topographical data (e.g., Digital Elevation Model [DEM]) is an input with a uniform grid structure [12].
- A surface roughness coefficient (Manning's n value) and source boundary information are provided as an input data set [10]. Typically, the roughness of the entire domain is represented using a single Manning's n value in 2D modeling applications [18], but our model can incorporate spatially varied surface roughness coefficients. For the purpose of performance evaluation, we only use a single Manning's n value across all grids in this study.
- The flow hydrographs provide water inflows to the domain that can be developed from a hydrological model, dam break model, or direct observations [10]. There can be single or multiple water source locations. After each time step, the depth values at each source location are calculated from the flow hydrograph input and updated for the next iteration.

4.2 Grids

Data are stored in 2D grid cells. Each cell has two constant values that do not change over the simulation: elevation and Manning's n value. Three values are updated at each cell during each time step: water depth and velocities in both x- and y-directions. The stencil (the neighboring cells needed to evolve the solution in time) for each numerical scheme is displayed in Figure 2.

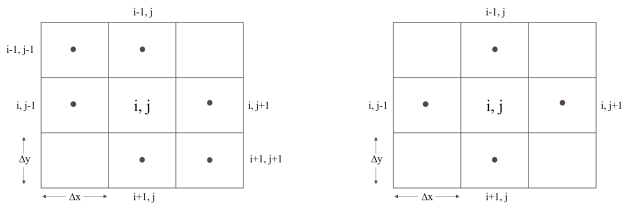

Figure 2: Grid stencil for (a) FD and (b) FV schemes

The grids are partitioned using a row-wise strategy over the MPI ranks, and double-precision floating-point numbers are used to store the values at each grid cell.

4.3 Step Size

Our model offers both variable and fixed time-step size. Although a variable time step based on the CFL condition offers the maximum allowable time-step size that makes the solution stable, the next Δt must be agreed upon by all ranks that require further communication. In our workflow, each rank first calculates the next local minimum Δt from all cells in its partition. The global minimum Δt is then identified and applied across all ranks.

4.4 Halo Exchange

As data is partitioned row-wise, the halo size is equal to the number of columns. Non-blocking communication has been used to perform the halo exchange by using MPI_Isend and MPI_Irecv. By using non-blocking communication, our application creates a request for communication for send and/or receive, gets back a handle, and then terminates. Two communication steps are needed to finish the halo exchange between all the ranks. In the first step, all the ranks $i < N - 1$ send halo data to rank $i + 1$, where N denotes the total number of ranks. In second step, all the ranks $i > 0$ send halo data to rank $i - 1$. MPI_Wait is used to check whether the communication has finished.

4.5 Kernels

Each computation step executed on the GPU is assigned to a CUDA kernel. There is a total of 8 kernels in the FD Scheme and 14 in the FV Scheme. There are separate kernels for computation of cell variables (water velocity, volume, height), copying halo cells, computing minimum time step size, updating boundary conditions, updating flow locations, and few other utility functions. Although it is possible to combine some of the kernels, separate kernels provide better code maintainability.

4.6 Output

The water height and velocity can be written at user-defined time intervals. Because output in ASCII is computationally intensive [28], we simply store our output in binary format and then use a post-processing script to convert it back to ASCII (when needed). As an example, Figure 3 shows the simulated maximum inundation depths for the Hurricane Harvey flood event using the FV model.

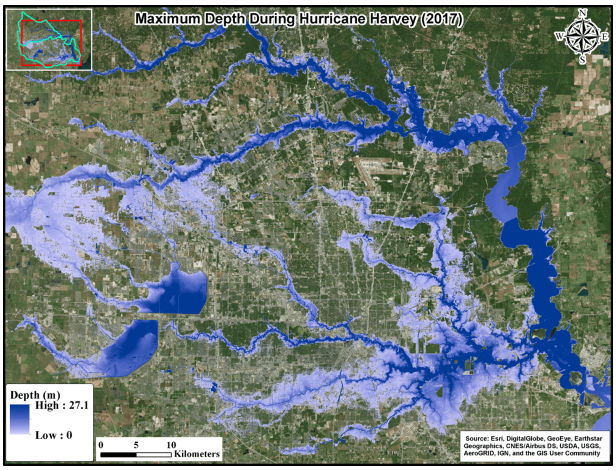

Figure 3: Simulated maximum inundation depths for Hurricane Harvey flood event

5 EXPERIMENTAL SETUP

The experimental setup is designed to measure and evaluate key performance metrics for specific test cases relevant to the community.

5.1 Key Performance Metrics

5.1.1 Billion Lattice Updates per Second. Million Lattice Updates Per Second (MLUPS) is a commonly used measure of simulation

efficiency across model implementations in the flood research community. In this study we use Billion Lattice Updates per Second (BLUPS), which are calculated as follows:

$$BLUPS = \frac{N_{gc} \times N_{ts}}{T \times 10^9}, \tag{18}$$

where N_{gc} is the number of grid cells, N_{ts} is the number of time steps used, and T is the run time of the simulation.

5.1.2 Speedup. We define Speedup as a measurement used to analyze how fast simulation is in comparison to the execution time using 1 GPU:

$$Speedup = \frac{T_1}{T_n}, \tag{19}$$

where T_1 is the execution time using 1 GPU and T_n is the execution time using n GPUs.

5.1.3 Communication Time. We define Communication Time (CT) as the total time required for any data exchange between multiple MPI processes along with data transfer between host and device:

$$CT = T_h + T_r + T_m, \tag{20}$$

where T_h is the total time for halo exchanges over all iterations, similarly T_r is the total for MPI reductions, and T_m is time for data transfer between host and device. These times are measured using individually timer and equation (20) is used to compute the total communication time.

5.2 Benchmark Test Cases

We used four different test cases to evaluate our FD and FV implementations. Table 1 summarizes the characteristics of all test cases. The binary output is generated at the final step of simulation for all test cases to avoid the uncertainty associated with data output.

Table 1: Total cell size and simulation time of all test cases

Test Case	Sources	No of Cells	Time(s)
Conasauga	300	4,852,675	432,000
Harvey 30m	69	7,562,646	864,000
Harvey 10m	69	68,080,474	864,000
Harvey 5m	69	272,321,896	864,000

5.2.1 Conasauga River Basin. The first test case represents the Conasauga River Basin, which is located in southeast Tennessee and northwest Georgia, USA. The Conasauga River Basin has a drainage area of approximately 1883 sq. km. The flood model has 300 inflow locations for a 5 day event, which are used as upstream and internal boundary conditions. The flow hydrographs used in this study were created from the annual maximum discharge events [11]. The topography of the Conasauga River Basin is represented by a 30 m resolution DEM made of 2,659 rows and 1,825 columns.

5.2.2 Hurricane Harvey. The other three test cases simulate a 10 day flood event that took place in August 2017 due to catastrophic rainfall-triggered flooding associated with Hurricane Harvey in the metropolitan area of Houston and Southeast Texas [40]. There are 69 inflow locations, and the elevation input data come from three different DEM resolutions (30 m, 10 m, and 5 m). The number of cells for each configuration is displayed in Table 1.

5.3 Hardware Specifications

All our experiments were run on Summit [32], the most powerful supercomputer at Oak Ridge National Laboratory (ORNL). Summit consists of 4,608 nodes, each with two IBM Power 9 processors, 512 GB of DDR4 RAM, and six NVIDIA Volta V100 GPUs. The processors are connected to the GPUs by NVIDIA's NVLink interconnect. Each link has a peak bandwidth of 25 GB/s (in each direction), and because there are two links between a processor and a GPU, data can be transferred from GPU-to-GPU and CPU-to-GPU at a peak rate of 50 GB/s. We used up to 128 Summit nodes (768 GPUs) for our experiments.

5.3.1 GPUDirect. The standard method of data transfer between GPUs usually follows three steps: (1) a device-to-host transfer to the host's RAM, using cudaMemcpy, (2) data exchange among appropriate MPI processes, and (3) a host-to-device transfer using cudaMemcpy as shown in figure 4a. However, Summit has an inter-node InfiniBand [14] interconnect with GPUDirect-RDMA [30] to speed up GPU-to-GPU transfers. With this technology, the data go directly from the GPU to the network and then to the destination GPU without doing any device-to-host or host-to-device transfers as shown in Figure 4b.

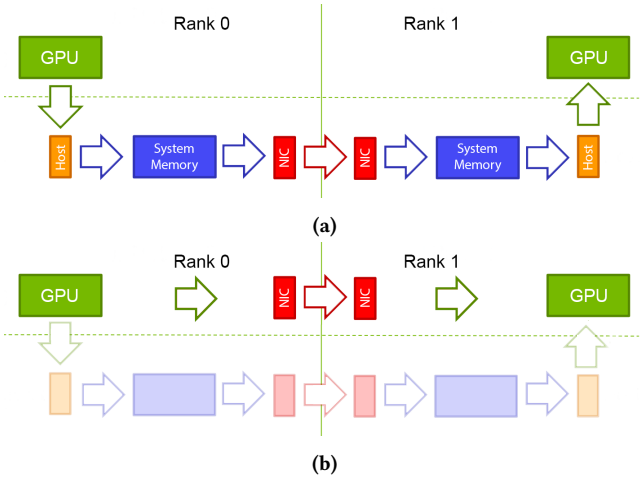

Figure 4: (a) Standard MPI GPU to remote GPU (b) MPI GPU to remote GPU using GPUDirect-RDMA

We used the CUDA-aware MPI [21] library with GPUDirect to send GPU buffers instead of host buffers. CUDA-aware MPI can more efficiently exploit the underlying protocol and can automatically utilize GPUDirect acceleration technologies.

5.4 Configurations

All the experiments were carried out using standard host-device transfers or GPUDirect transfers for exchanging the halo rows between MPI ranks. Combined with the two numerical schemes, this gives four configurations: FDS (finite difference standard host-device transfer), FDG (finite difference GPUDirect), FVS (finite volume standard host-device transfer) and FVG (finite volume GPUDirect).

6 RESULTS AND ANALYSIS

Note that all the simulations used one MPI process per GPU using six MPI processes per node (Summit contains six GPUs per node).

6.1 Scaling

6.1.1 Weak Scaling. Figure 5 shows the performance of the three Hurricane Harvey test cases (i.e., three resolutions) with the number of GPUs for a fixed problem size per GPU with four configurations. We observe that for all configurations, performance increases as the number of GPUs increases. For example, in the case of FVG when we use a 9x input size (10 m resolution), performance increases by a factor of 7.13 (from 17.9 to 127.6 BLUPS). Furthermore, a 36x input size (5 m resolution) shows a factor of 22.28 increase. The other three configurations show similar behaviors.

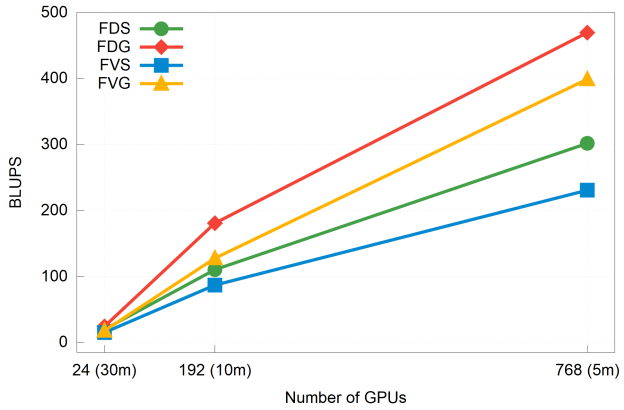

Figure 5: Performance of Hurricane Harvey test cases with the number of GPUs for a fixed problem size per GPU

6.1.2 Strong Scaling. Figure 6 shows the performance of four test cases and four configurations with different numbers of GPUs. Conasauga has the smallest domain and scales the least, stagnating after 24 GPUs for all configurations. The slightly larger Harvey 30 m case scales out to 8–16 nodes (48–96 GPUs), depending on the configuration. The Harvey 10 m domain is 9x larger than the Harvey 30 m domain, thus showing increasing performance out to 64 nodes (384 GPUs) from 17.69–140.65 BLUPS in the case of FVG configuration. We also see our first scaling differences between the configurations. The GPUDirect version of each numerical scheme becomes increasingly better than its standard counterpart. The Harvey 5 m case continues the trend and scales out to 128 nodes (768 GPUs) from 36.49–469.4 BLUPS in the case of FDG configuration. As long as the number of GPUs increases, the GPUDirect configurations become even more dominant, achieving 50–70% more BLUPS than the standard configurations.

6.2 Runtime

The total runtime of the simulation is often more important than the efficiency of the scaling. Figure 7 shows the total runtime for the different test cases and configurations. Note the apparent inconsistency in Figure 6, where the FD scheme achieves higher BLUPS (than the FV scheme) but takes longer to run. The main reason is that the FD scheme needs a lower Courant value (0.1 versus 0.5) to remain stable. Thus, the Δt is smaller and more iterations are needed.

Conasauga and Harvey 30 m are both small enough that it would be reasonable to run them on a single node. Harvey 10 m and Harvey 5 m are more computationally challenging. Here a single node (6 GPUs) Harvey 10 m run takes over 10 hours. Using 64 nodes (384 GPUs), the FVG configuration drops to 18 minutes. For Harvey 5 m, the large differences become 80 hours versus 50 minutes.

6.3 GPUDirect

Figures 6 and 7 suggest that the GPUDirect approach is always equal to or better than the standard configuration. To quantify this, Figure 8 shows just the communication time of the test cases and configurations. We can see that the GPUDirect communication time is always lower than the standard communication time, being as low as one-quarter of the standard time for large test cases. Note that the communication time does not increase considerably with more nodes for a given configuration and test case.

6.4 Speedup

Figure 9 shows the Speedup with respect to the number of GPUs for all four test cases and configurations. As observed, Harvey 5 m shows the maximum increase in speedup (e.g., FVG using 768 GPUs shows a speedup of 98.2). In addition, FV scheme implementation has a higher speedup value than the FD scheme, mainly because of higher computationally expensive kernels for each time step. We can also deduce from Figure 9 that a higher resolution usually implies a higher speedup.

7 CONCLUSION AND FUTURE WORK

We implemented a two-dimensional flood simulation using two numerical schemes and simulated four test cases on Summit, ORNL's most powerful supercomputer. Both standard inter GPU communication and GPUDirect for halo exchange were used. All implementations showed reasonable weak and strong scalability (up to 768 GPUs for the largest test case). In all cases, the GPUDirect showed better computational performance than the standard configuration. The difference in performance was minimal when a lower number of GPUs were used but increased as the number of GPUs increased. Based on our experimental results, we can conclude that 2D stencil-type codes using GPUDirect for halo exchange will perform better on Summit than the standard default communication. Additionally, the computational performance of the finite volume scheme is better than the finite difference scheme.

For work division, we used 1D row-wise partitioning rather than a different strategy that can be used to exchange halo between MPI processes (e.g., a higher-order partitioning method). Furthermore, knowledge of the underlying hardware devices is essential to the development of software intended for large-scale grid simulations. Our future efforts will include overlap of halo exchange with stencil computation, the development of a set of generalized data structures supporting several different partitioning strategies, run-time configuration parameters matching a selected partitioning strategy with a complementary mapping strategy, and a simple,

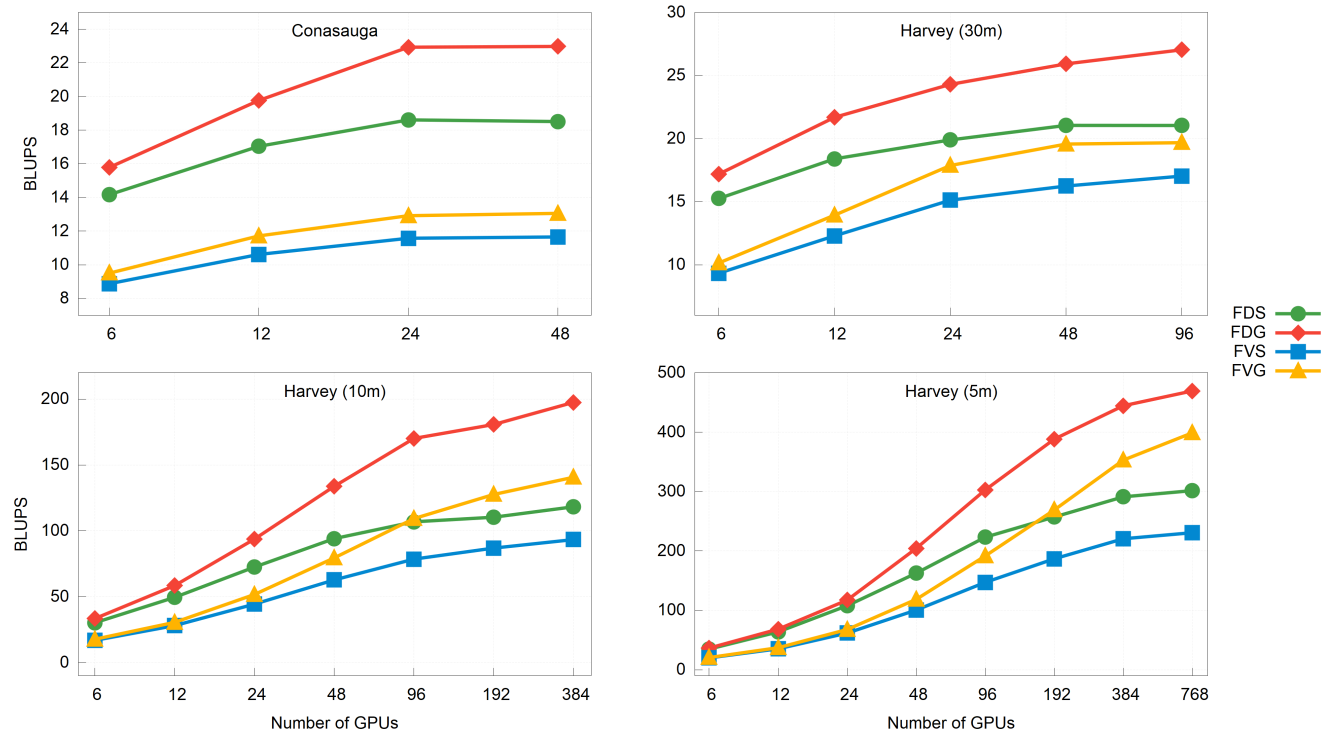

Figure 6: Performance of test cases with an increasing number of GPUs

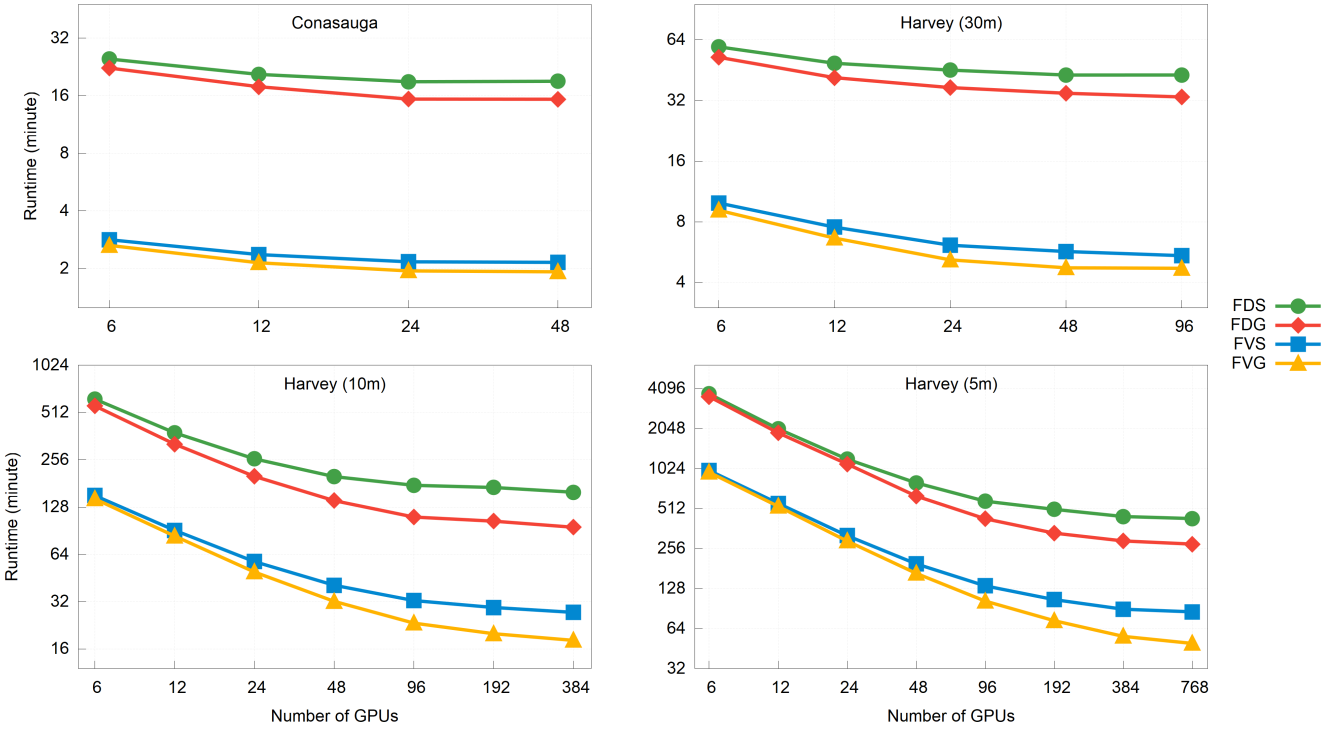

Figure 7: Runtime of test cases with an increasing number of GPUs

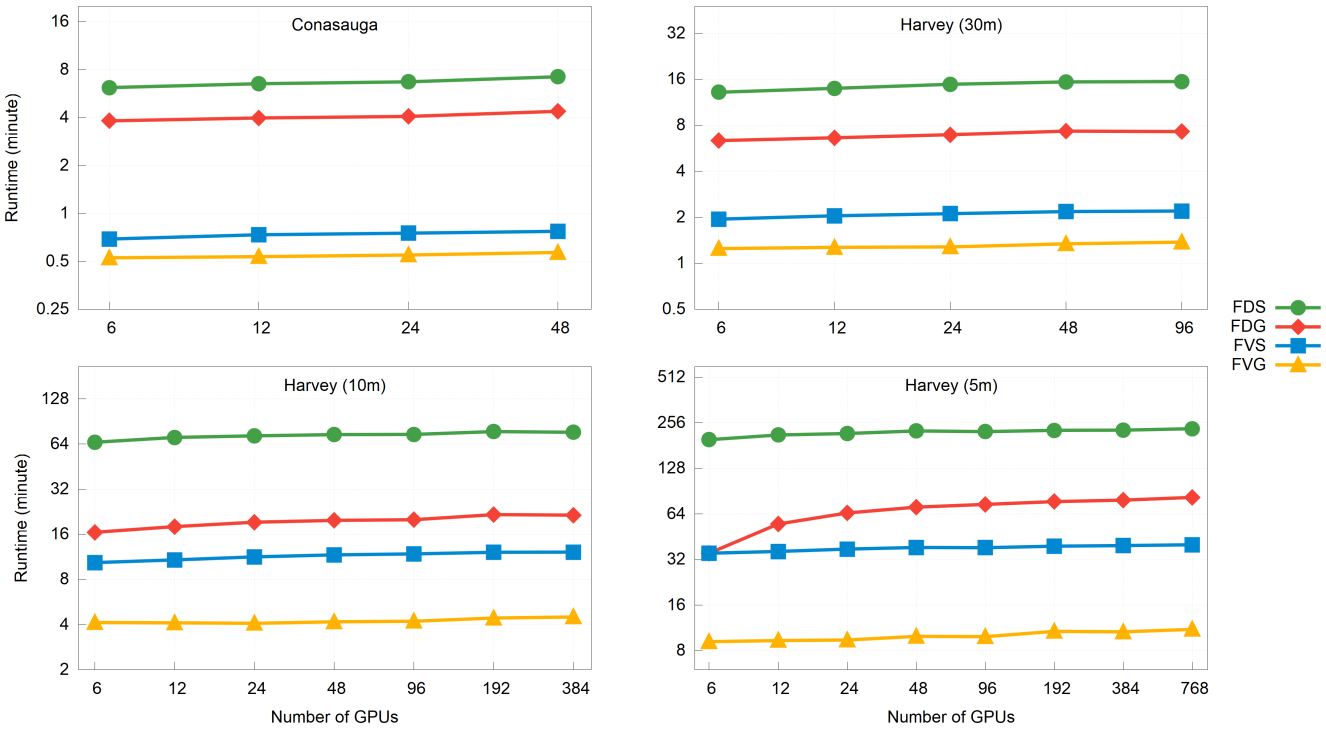

Figure 8: Communication time of test cases with an increasing number of GPUs

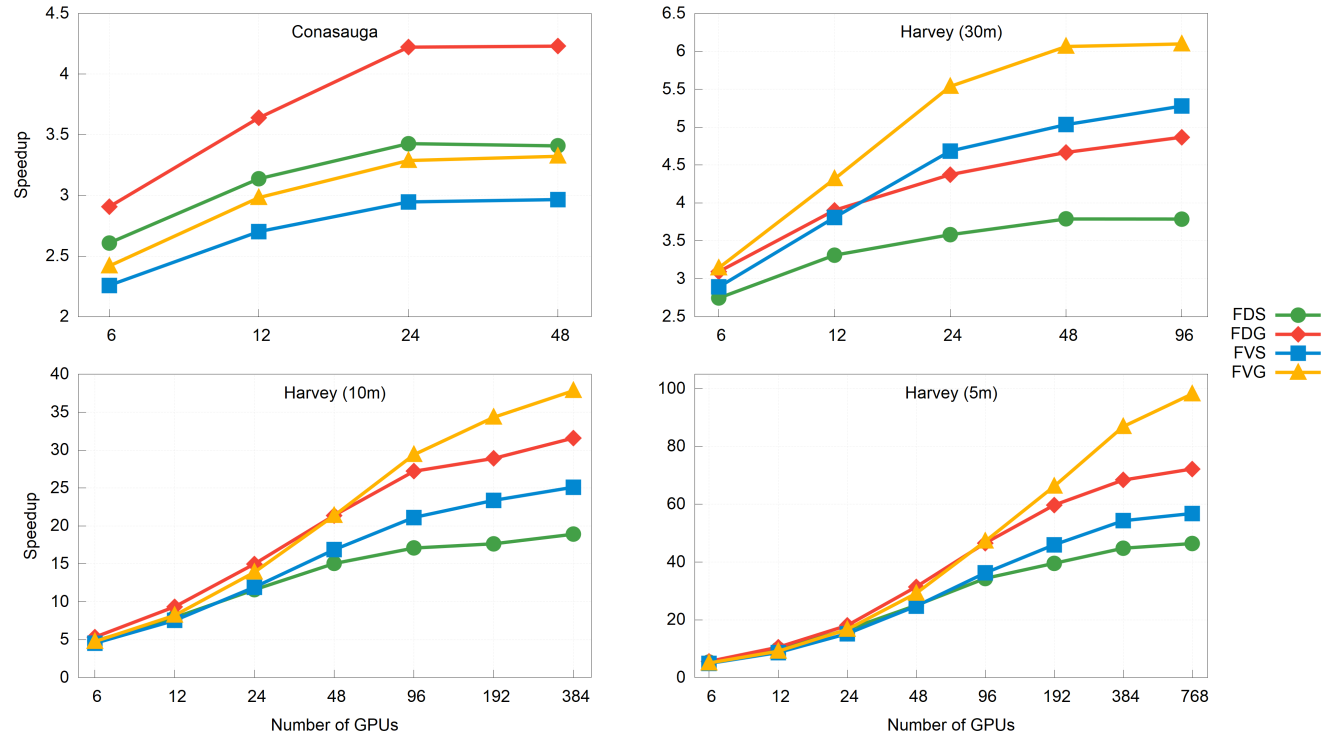

Figure 9: Speedup of test cases with an increasing number of GPUs

user-friendly meta computing API so that domain scientists will be able to access efficient hardware without needing to have the level of expertise that is required today.

8 ACKNOWLEDGMENTS

This research was supported by the US Air Force Numerical Weather Modeling Program and the Center of Management, Utilization, and Protection of Water Resources at Tennessee Technological University. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory. Some of the co-authors are employees of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy. Accordingly, the US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript or allow others to do so, for US Government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] André R Brodtkorb, Trond R Hagen, Knut-Andreas Lie, and Jostein R Natvig. 2010. Simulation and visualization of the Saint-Venant system using GPUs. *Computing and Visualization in Science* 13, 7 (2010), 341–353.

[2] André R Brodtkorb, Martin L Sætra, and Mustafa Altinakar. 2012. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids* 55 (2012), 1–12.

[3] Gary W Brunner. 1995. *HEC-RAS River Analysis System. Hydraulic Reference Manual. Version 1.0.* Technical Report. Hydrologic Engineering Center Davis CA.

[4] Gary W Brunner. 2016. *HEC-RAS River Analysis System. Hydraulic Reference Manual. Version 5.0.* Technical Report. Hydrologic Engineering Center Davis CA.

[5] US Costliest. 2018. Tropical Cyclones Tables Updated.

[6] Norman H Crawford and Ray K Linsley. 1966. Digital Simulation in Hydrology: Stanford Watershed Model 4.

[7] Marc de la Asunción, Manuel J Castro, José Miguel Mantas, and Sergio Ortega. 2016. Numerical simulation of tsunamis generated by landslides on multiple GPUs. *Advances in Engineering Software* 99 (2016), 59–72.

[8] Marc De La AsunciÓN, José M Mantas, Manuel J Castro, and Enrique Domingo Fernández-Nieto. 2012. An MPI-CUDA implementation of an improved Roe method for two-layer shallow water systems. *J. Parallel and Distrib. Comput.* 72, 9 (2012), 1065–1072.

[9] DHI. 2014. MIKE 11 A Modelling System for Rivers and Channels Reference Manual. "https://www.mikepoweredbydhi.com/products/mike-11"

[10] Tigstu TSIGE Dullo, Sudershan Gangrade, Alfred J Kalyanapu, Shih-Chieh Kao, Sheikh K Ghafoor, and Katherine J Evans. 2018. High-resolution modeling of Hurricane Harvey Flooding for Harris County, TX using a calibrated GPU-accelerated 2D Flood Model.

[11] Tigstu TSIGE Dullo, Sudershan Gangrade, Ryan Marshall, Sheikh R Islam, Sheikh K Ghafoor, Shih-Chieh Kao, and Alfred J Kalyanapu. 2017. A large-scale simulation of climate change effects on flood regime-A case study for the Alabama-Coosa-Tallapoosa River Basin.

[12] Dean Gesch, Michael Oimoen, Susan Greenlee, Charles Nelson, Michael Steuck, and Dean Tyler. 2002. The national elevation dataset. *Photogrammetric engineering and remote sensing* 68, 1 (2002), 5–32.

[13] R Gommes, J Du Guerny, F Nachtergaele, and R Brinkman. 1998. Potential impacts of sea-level rise on populations and agriculture.

[14] Paul Grun. 2010. Introduction to infiniband for end users.

[15] Michele Guidolin, Albert S Chen, Bidur Ghimire, Edward C Keedwell, Slobodan Djordjević, and Dragan A Savić. 2016. A weighted cellular automata 2D inundation model for rapid flood analysis. *Environmental Modelling & Software* 84 (2016), 378–394.

[16] Trond Runar Hagen, Jon M Hjelmervik, K-A Lie, Jostein R Natvig, and M Ofstad Henriksen. 2005. Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory* 13, 8 (2005), 716–726.

[17] Mark J Harris, Greg Coombe, Thorsten Scheuermann, and Anselmo Lastra. 2002. Physically-based visual simulation on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. Eurographics Association, Goslar, DEU, 109–118.

[18] David R Judi, Steven J Burian, and Timothy N McPherson. 2010. Two-dimensional fast-response flood modeling: Desktop parallel computing and domain tracking. *Journal of Computing in Civil Engineering* 25, 3 (2010), 184–191.

[19] Alfred J Kalyanapu, Siddharth Shankar, Eric R Pardyjak, David R Judi, and Steven J Burian. 2011. Assessment of GPU computational enhancement to a 2D flood model. *Environmental Modelling & Software* 26, 8 (2011), 1009–1016.

[20] Shih-Chieh Kao, Scott T DeNeale, and David B Watson. 2019. Hurricane Harvey Highlights: Need to Assess the Adequacy of Probable Maximum Precipitation Estimation Methods. *Journal of Hydrologic Engineering* 24, 4 (2019), 05019005.

[21] Jiri Kraus. 2013. An Introduction to CUDA-Aware MPI. https://devblogs.nvidia.com/introduction-cuda-aware-mpi/

[22] Asier Lacasta, Mario Morales-Hernández, Javier Murillo, and Pilar García-Navarro. 2014. An optimized GPU implementation of a 2D free surface simulation model on unstructured meshes. *Advances in engineering software* 78 (2014), 1–15.

[23] YG Lai. 2008. SRH-2D version 2: Theory and User's Manual.

[24] Rob Lamb, Mandy Crossley, and Simon Waller. 2009. A fast two-dimensional floodplain inundation model. In *Proceedings of the Institution of Civil Engineers-Water Management*, Vol. 162. Thomas Telford Ltd, Scotland, 363–370.

[25] Qiuhua Liang and Luke S Smith. 2015. A high-performance integrated hydrodynamic modelling system for urban flood simulations. *Journal of Hydroinformatics* 17, 4 (2015), 518–533.

[26] Ryan Marshall, Sheikh K Ghafoor, Alfred J Kalyanapu, Mike Rogers, and Tigstu T Dullo. 2017. Performance Improvement of a Two-Dimensional Flood Simulation Application in Hybrid Computing Environments. In *2017 Fifth International Symposium on Computing and Networking (CANDAR)*. IEEE, Aomori, Japan, 21–29.

[27] M Morales-Hernández, A Lacasta, J Murillo, P Brufau, and P García-Navarro. 2015. A Riemann coupled edge (RCE) 1D–2D finite volume inundation and solute transport model. *Environmental Earth Sciences* 74, 11 (2015), 7319–7335.

[28] Mario Morales Hernandez, Md Bulbul Sharif, Sudershan Gangrade, Tigstu T Dullo, Shih-Chieh Kao, Alfred J Kalyanapu, Sheikh Ghafoor, and Katherine J Evans. 2019. High Performance Computing in hydraulics: the new era of flood forecasting. In *Proceedings of Modeling Hydrodynamics for Water Resources (MODWATER 2019)*. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Zaragoza, Spain.

[29] Javier Murillo and P García-Navarro. 2010. Weak solutions for partial differential equations with source terms: Application to the shallow water equations. *J. Comput. Phys.* 229, 11 (2010), 4327–4368.

[30] NVIDIA. 2019. Developing a Linux Kernel Module using GPUDirect RDMA. https://docs.nvidia.com/cuda/gpudirect-rdma/index.html

[31] NVIDIA. 2019. GPUDirect. https://developer.nvidia.com/gpudirect

[32] OLCF. 2019. Summit. https://www.olcf.ornl.gov/summit/

[33] Martin Lilleeng Sætra and André Rigland Brodtkorb. 2010. Shallow water simulations on multiple GPUs. In *International Workshop on Applied Parallel Computing*. Springer, Berlin, Heidelberg, 56–66.

[34] Luke S Smith and Qiuhua Liang. 2013. Towards a generalised GPU/CPU shallow-flow modelling tool. *Computers & Fluids* 88 (2013), 334–343.

[35] V Tayefi, SN Lane, RJ Hardy, and D Yu. 2007. A comparison of one-and two-dimensional approaches to modelling flood inundation over complex upland floodplains. *Hydrological Processes: An International Journal* 21, 23 (2007), 3190–3202.

[36] Tawatchai Tingsanchali and Winyu Rattanapitikon. 1999. 2-D modelling of dambreak wave propagation on initially dry bed. *Thammasat Int. J. Sc. Tech* 4, 3 (1999), 28–37.

[37] Renato Vacondio, Alessandro Dal Palù, Alessia Ferrari, Paolo Mignosa, Francesca Aureli, and Susanna Dazzi. 2017. A non-uniform efficient grid type for GPU-parallel Shallow Water Equations models. *Environmental modelling & software* 88 (2017), 119–137.

[38] Renato Vacondio, Alessandro Dal Palù, and Paolo Mignosa. 2014. GPU-enhanced finite volume shallow water solver for fast flood simulations. *Environmental modelling & software* 57 (2014), 60–75.

[39] Moisés Viñas, Jacobo Lobeiras, Basilio B Fraguela, Manuel Arenaz, Margarita Amor, José A García, Manuel J Castro, and Ramon Doallo. 2013. A multi-GPU shallow-water simulation with transport of contaminants. *Concurrency and Computation: Practice and Experience* 25, 8 (2013), 1153–1169.

[40] Wikipedia contributors. 2019. Hurricane Harvey — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Hurricane_Harvey

[41] Xilin Xia and Qiuhua Liang. 2016. A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations. *Environmental modelling & software* 75 (2016), 28–43.

[42] Xilin Xia and Qiuhua Liang. 2018. A new efficient implicit scheme for discretising the stiff friction terms in the shallow water equations. *Advances in water resources* 117 (2018), 87–97.