

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
moralesherm@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 are able to solve the nonlinear hyperbolic shallow water equations using an explicit upwind first-order approach on finite differences and finite volumes respectively. They have been implemented using MPI and CUDA. Four different test cases have been simulated on the Summit supercomputer at Oak Ridge National Laboratory. Both numerical schemes scale up to 128 nodes (768 GPUs) with a maximum 98.2x speedup over 1 GPU. The lowest run time for the 10 days Hurricane Harvey event simulation at 5-meter resolution (272 million grid cells) is 50 minutes. GPUDirect communication is proved to be more convenient than the standard communication strategy. Both strong and weak scaling are shown.

KEYWORDS

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

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). 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>).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

PASC '20, June 29–July 01, 2020, Geneva, Switzerland

© 2020 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

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 *PASC '20: Proceedings of the Platform for Advanced Scientific Computing Conference, June 29–July 01, 2020, Geneva, Switzerland*. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

More than 21% of the global population live within 30 km of a coastline [14] 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 [7] [20]. Hydrodynamic flood models have been applied to simulate floods for more than 50 years [8]. With the increase of computational capabilities, more accurate and rapid hydraulic models have emerged with the purpose of providing a reliable basis for decision-making in flood risk management.

Traditionally, flood models used a 1D approach such as the Hydrologic Engineering Center's River Analysis System (HEC-RAS) [5] or MIKE 11 [2], which are computationally less expensive in terms of total run time. Nevertheless, 1D models, especially in urban flood-prone areas, cannot properly represent flood wave propagation over the floodplains in events that exceed the predefined 1D river channels [32]. On the other hand, 2D models such as HEC-RAS 2D [6] or SRH-2D [22] eliminate the primary limitations of 1D models by allowing the topographical representation of higher-order and preferential flood pathways [32]. 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) [23] [18]. The recent graphical processing unit (GPU)-based large systems such as Summit [1] provides an opportunity to simulate flood at high resolution. Such simulation requires the code to be scalable over multiple GPUs in a distributed, heterogeneous computing environment. Limited efforts have been reported in the literature on using

multiple GPUs for solving the shallow water equations [10], [36], [38], flood simulation [27], [30], [25] or tsunami [9]. The current work reported in this paper is one of the few examples of using multiple GPUs to simulate a 2D flood model.

One of the objectives of our current research effort 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 present the evaluation of our implementation on Summit using multiple nodes with MPI and CUDA.

The contributions of this paper 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 communication [29].

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 [8], which is 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 GPU was presented and demonstrated a speedup of 25x on an NVIDIA GeForce 4 compared with the same simulation run on a Pentium CPU. [16] presented a visual simulation study of shallow-water waves using GPUs and reported a speedup of 15x to 30x compared with a CPU simulation. [23] presented a speedup of 112x using a diffusive wave flood modeling approach compared with a CPU model. In [19], they described a study of the computational enhancement of a GPU-enabled 2D flood model. The researchers achieved an 88x speedup on a Tesla C1060 GPU. [4] presented an efficient implementation of a state-of-the-art high-resolution explicit scheme for the shallow water equations on GPUs. They demonstrated computation of the first 4000 s of the Malpasset dam-break case in 27 s. Additionally, [21] described the implementation of a finite volume method to solve the 2D shallow water equations on the GPU. The strategy is fully oriented to work efficiently with unstructured meshes which are widely used in many fields of engineering. [35] described a shallow water equations parallel numerical scheme suitable for GPU. They claimed to get speedups of two orders of magnitude compared to a single-core CPU. The same authors [34] also implemented a few years later a GPU-parallel numerical model for the solution of the 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). [31], [3], [24] and [15]

also tried to use GPU for their flood modeling and described good performance improvement compare to the CPU version.

One of the major limitations of the researches mentioned above is that they are limited to a single GPU. [30] presented a state-of-the-art shallow water simulator running on multiple GPUs. Their implementation showed a rate of over 1.2 Gigacells per second using four Fermi-generation GPUs. In [9] they 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 [25], 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 presented a maximum performance of 3 Gigacells per second.

3 NUMERICAL SCHEMES

Two different numerical schemes are compared in this work: 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 the 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 [33] [18] are as follows:

Continuity equation

$$\frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} + v \frac{\partial h}{\partial y} = 0 \quad (1)$$

Momentum equation in x-direction

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + g \frac{\partial H}{\partial x} + g S_{fx} = 0 \quad (2)$$

Momentum equation in y-direction

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + g \frac{\partial H}{\partial y} + g S_{fy} = 0 \quad (3)$$

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:

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

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

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

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

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

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

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

$$\Delta t = \text{CFL} \min_i (\Delta x_i, \Delta y_i) \quad \text{CFL} = 0.1 \quad (10)$$

where CFL represents Courant value and $\epsilon = 0.001$.

3.2 FV Scheme

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

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S}_b + \mathbf{S}_f \quad (11)$$

$$\mathbf{U} = \begin{pmatrix} h \\ q_x \\ q_y \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} q_x \\ \frac{q_x^2}{h} + \frac{1}{2}gh^2 \\ \frac{q_x q_y}{h} \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} q_y \\ \frac{q_x q_y}{h} \\ \frac{q_y^2}{h} + \frac{1}{2}gh^2 \end{pmatrix} \quad (12)$$

$$\mathbf{S}_b = \begin{pmatrix} 0 \\ -gh \frac{\partial z}{\partial x} \\ -gh \frac{\partial z}{\partial y} \end{pmatrix} \quad \mathbf{S}_f = \begin{pmatrix} 0 \\ -\frac{gn^2}{h^{7/3}} q_x \sqrt{q_x^2 + q_y^2} \\ -\frac{gn^2}{h^{7/3}} q_y \sqrt{q_x^2 + q_y^2} \end{pmatrix}$$

where \mathbf{U} is the vector of conserved variables including the water depth, h , and the unit discharges in x and y directions, named 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 contains the bed and roughness source terms respectively, being z the bed elevation. Again, roughness is modeled using Manning-Gauckler's law.

In this scheme, a finite upwind volume explicit system is introduced based on the Roe's linearization. A squared mesh is used, denoting Δx as the grid spacing. The derivation of the numerical scheme follows [28], [26] 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 following [39], 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} :

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

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

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:

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

where

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

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

$$\Delta t = \text{CFL} \frac{\Delta x}{\max_i \left\{ \left| \frac{q_x}{h} \right|_i + \sqrt{gh_i}, \left| \frac{q_y}{h} \right|_i + \sqrt{gh_i} \right\}} \quad \text{CFL} \leq 0.5 \quad (17)$$

More details about the numerical scheme can be found in [28], [26], and [39].

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.

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 [13].
- A surface roughness coefficient (Manning's n value) and source boundary information are provided as an input data set [11]. 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 directly from observations [11]. 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 is 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.

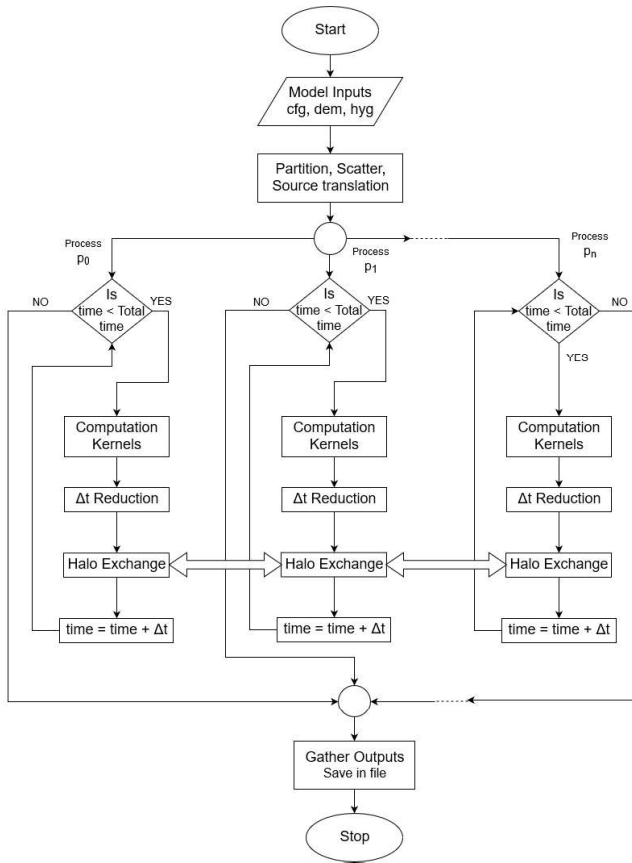


Figure 1: Parallel flowchart

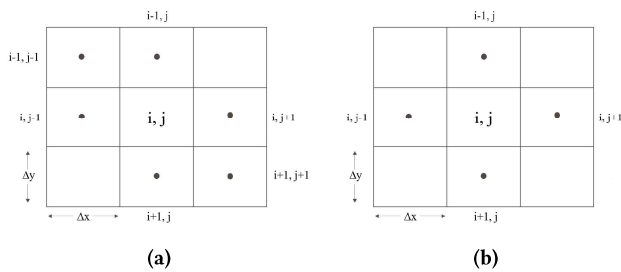


Figure 2: Grid stencil for (a) FD and (b) FV schemes

The grids are partitioned using a row-wise strategy over the MPI ranks. Besides, 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 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 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 Output

The water height and velocity can be written at user-defined time intervals. Since output in ASCII is computationally intensive [27], we simply store our output in binary format and then use a post-processing script to convert them 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.

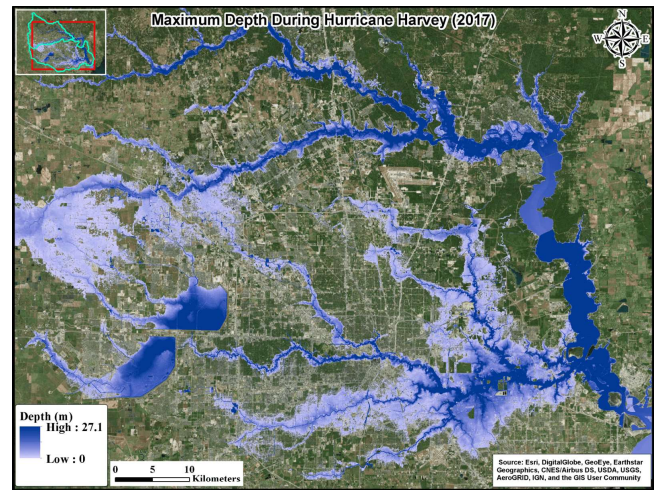


Figure 3: Simulated Max. Inundation depths for Hurricane Harvey flood event

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 (BLUPS). 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 BLUPS that are calculated as follows:

$$BLUPS = \frac{N_{gc} \times N_{ts}}{T \times 10^9} \quad (18)$$

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 the Speedup as a measurement to analyze how fast simulation is in comparison to the execution time using 1 GPU:

$$Speedup = \frac{T_1}{T_n} \quad (19)$$

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 (CT). 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:

$$CT = T_h + T_r + T_m \quad (20)$$

where T_h is the time needed for all halo exchanges, T_r is the time for all MPI reductions, and T_m is time for all data copy between host and device.

5.2 Benchmark Test Cases

We use 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 that is located in southeast Tennessee and northwest Georgia, USA. The Conasauga River Basin has an approximate drainage area of 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 [12]. 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 [37]. There are 69 inflow locations and the elevation input data come from three different DEM resolutions (30 m, 10 m, and 5 m, respectively). The number of cells for each configuration is displayed in Table 1.

5.3 Hardware Specifications

We ran all experiments using Summit [1], the latest 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 since there are 2 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 way of data transfer between GPUs usually follows three steps: 1) a device-to-host transfer, 2) data exchange among MPI ranks, and 3) a host-to-device transfer. However, Summit has the capability to use GPUDirect RDMA to speed up intra-node and inter-node GPU-to-GPU transfers. With this technology, the data goes directly from the GPU to the network and then to the destination GPU without doing any device-to-host or host-to-device transfers.

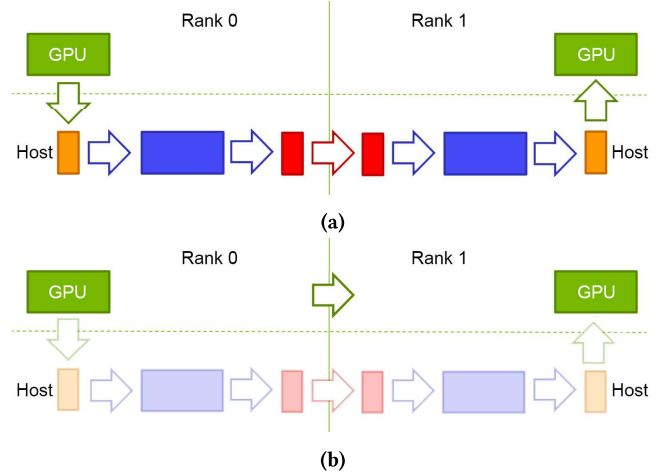


Figure 4: (a) Regular MPI GPU to remote GPU (b) MPI GPU to remote GPU using GPUDirect

5.4 Configurations

All the experiments are carried out using standard host-device transfers and 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 1 MPI process per GPU using 6 MPI processes per node (Summit contains 6 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 four configurations using 16 Summit nodes (96 GPUs). We observe that for

all configurations, performance increases as the domain size grows. As an example, in the case of FVG when we use 9x input size (10m resolution) performance increases with a factor of 5.55 (from 19.67 to 109.27 BLUPS). Furthermore, a 36x input size (5m resolution) shows a factor of 9.74. The other three configurations also show similar behaviors.

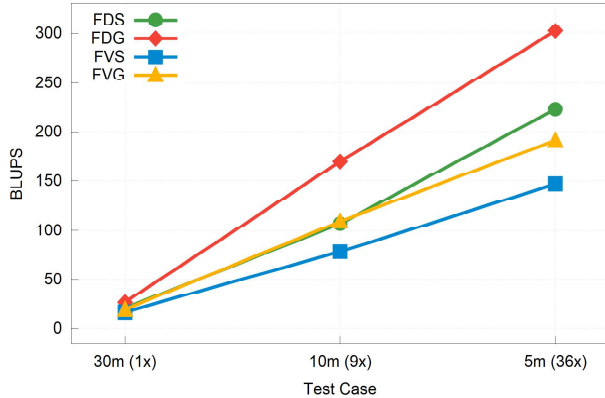


Figure 5: Performance of Hurricane Harvey test cases with different domain sizes on 96 GPUs

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 30m case scales out to 8-16 nodes (48-96 GPUs), depending on the configuration. The Harvey 10m domain is 9x larger than the Harvey 30m domain, and thus gives increasing performance out to 64 nodes (384 GPUs) from 17.69 to 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 5m case continues the trend and scales out to 128 nodes (768 GPUs) from 36.49 to 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% to 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 30m are both small enough that it would be reasonable to run them on a single node. Harvey 10m and Harvey 5m are more computationally challenging. Here a single node (6 GPUs) Harvey 10m run takes over 10 hours. Using 64 nodes (384

GPUs), the FVG configuration drops to 18 minutes. For Harvey 5m, the large differences become 80 hours versus 50 minutes.

6.3 GPUDirect

Figures 6 and 7 suggest that 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 Speed Up

Figure 9 shows the Speedup with respect to the number of GPUs for all four test cases and configurations. As observed, Harvey 5m shows the maximum increase of speedup (e.g., FVG using 768 GPUs shows a Speedup of 98.2). Besides, 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 have implemented a two-dimensional flood simulation using two numerical schemes and simulated four test cases on Summit supercomputer at ORNL. We have used both the standard inter GPU communication and GPUDirect for halo exchange. All implementations show reasonable weak and strong scalability (up to 768 GPUs for the largest test case). In all cases, the GPUDirect has better computational performance than the standard configuration. The performance gap is not much at the lower number of GPUs but it grows along with the number of GPUs. From our experimental results, we can conclude that 2D stencil-type codes using GPUDirect for halo exchange will achieve better performance on Summit compared to the standard default communication. Additionally, the computational performance of the finite volume scheme is higher than the finite difference scheme.

For work division, we have used 1D row-wise partitioning despite 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 exploration includes 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, user-friendly meta computing API so that domain scientists can access efficient hardware without the expertise that is required today.

8 ACKNOWLEDGEMENTS

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

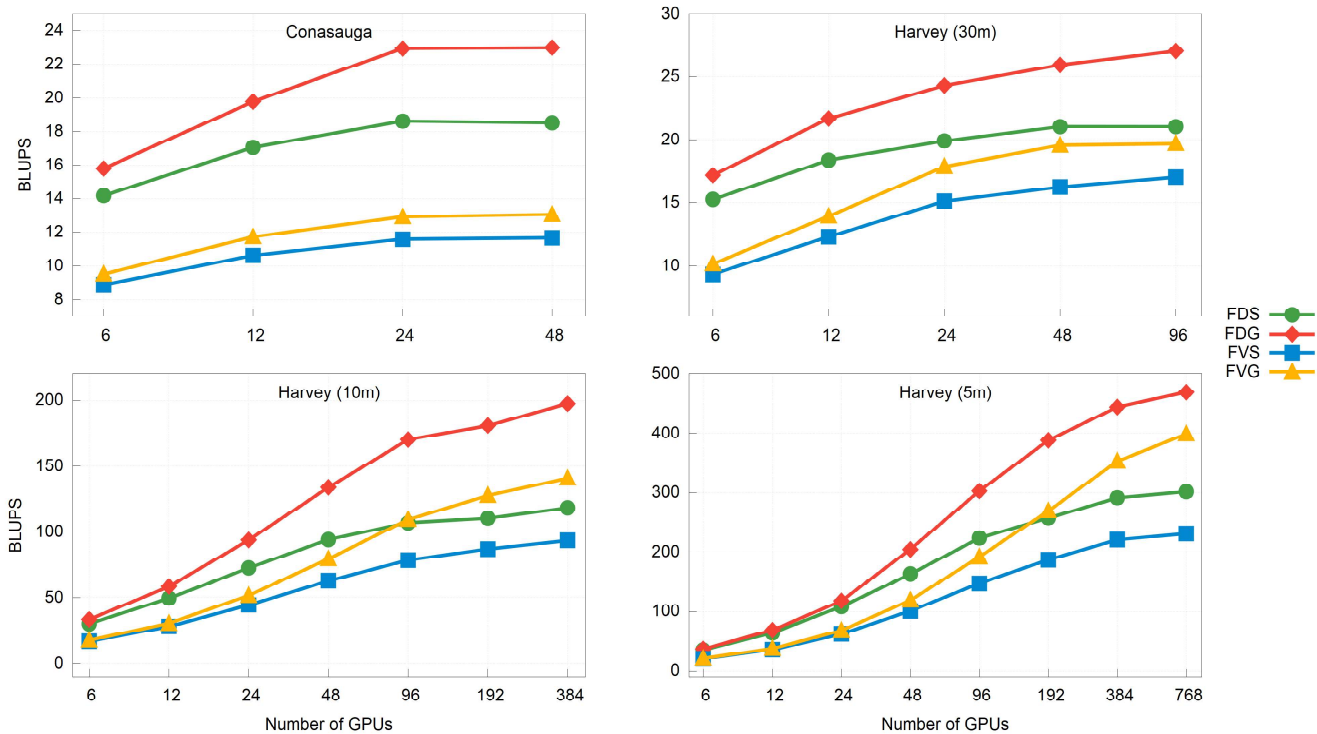


Figure 6: Performance of test cases with an increasing number of GPUs

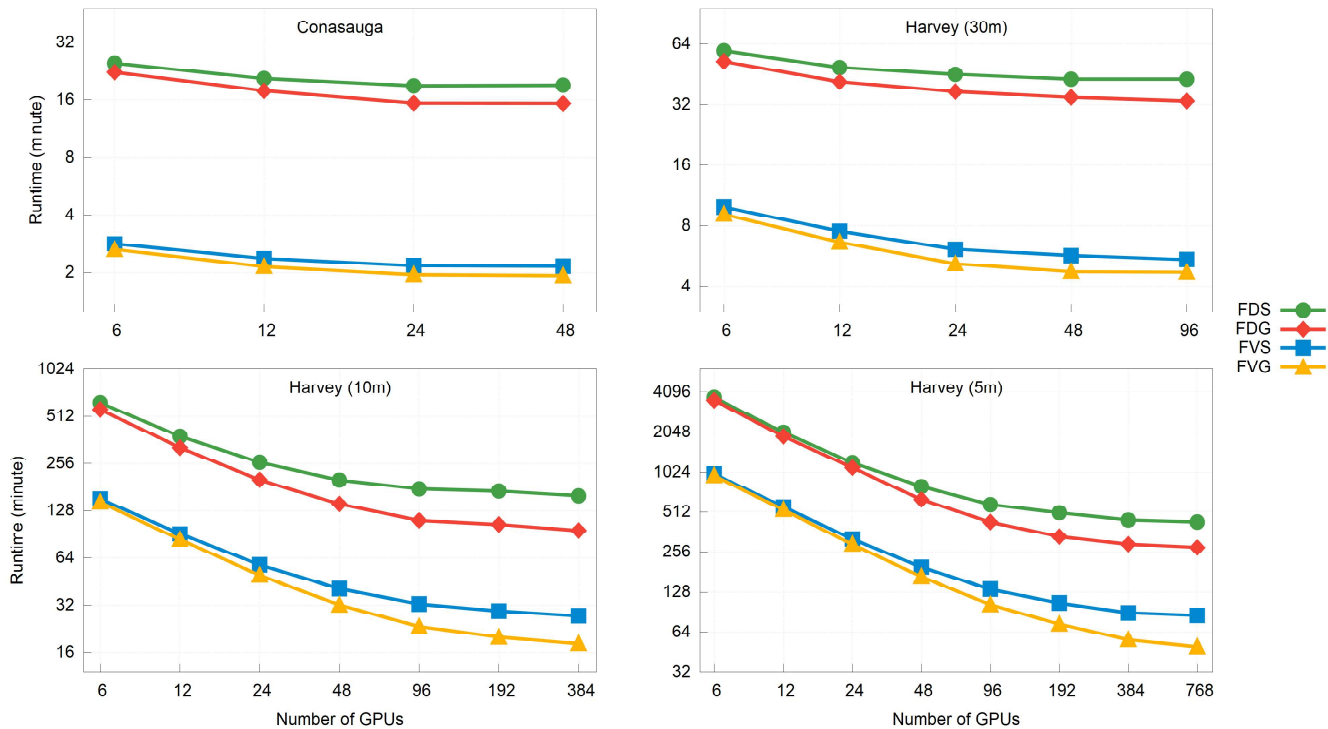


Figure 7: Runtime of test cases with an increasing number of GPUs

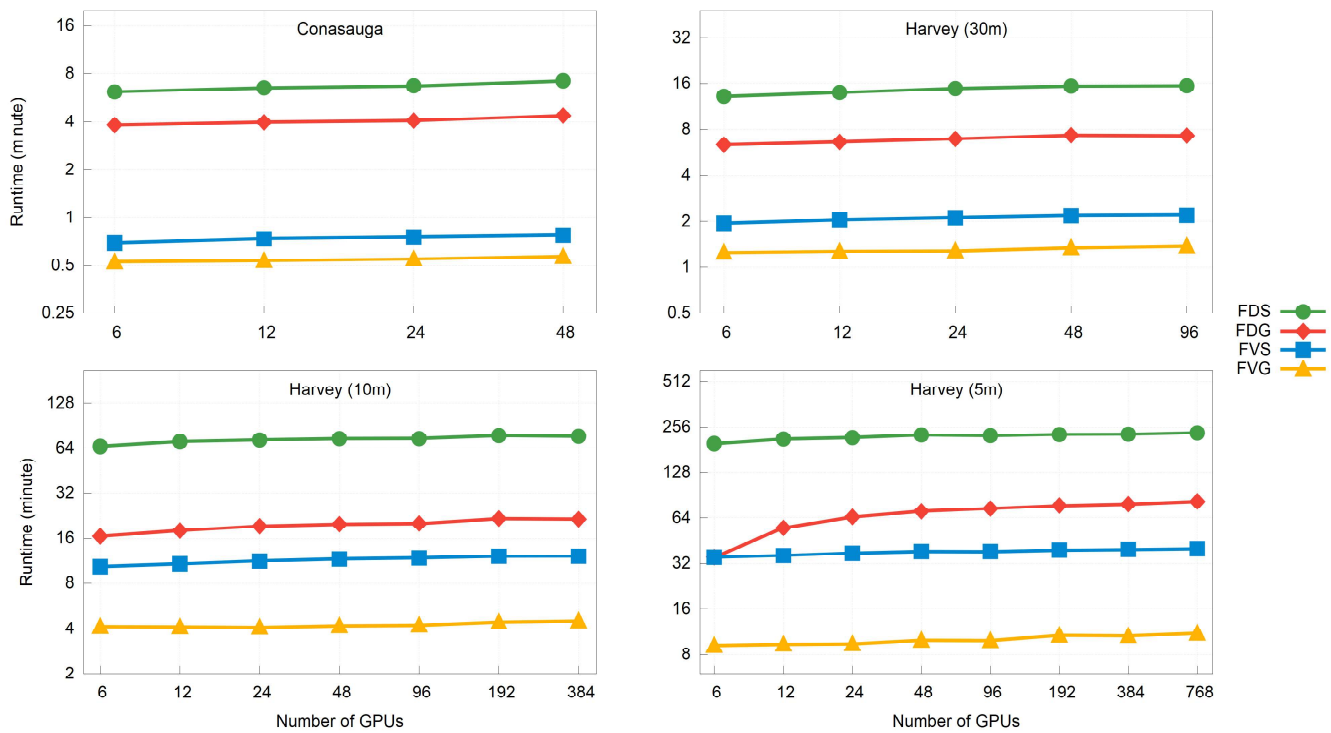


Figure 8: Communication time of test cases with an increasing number of GPUs

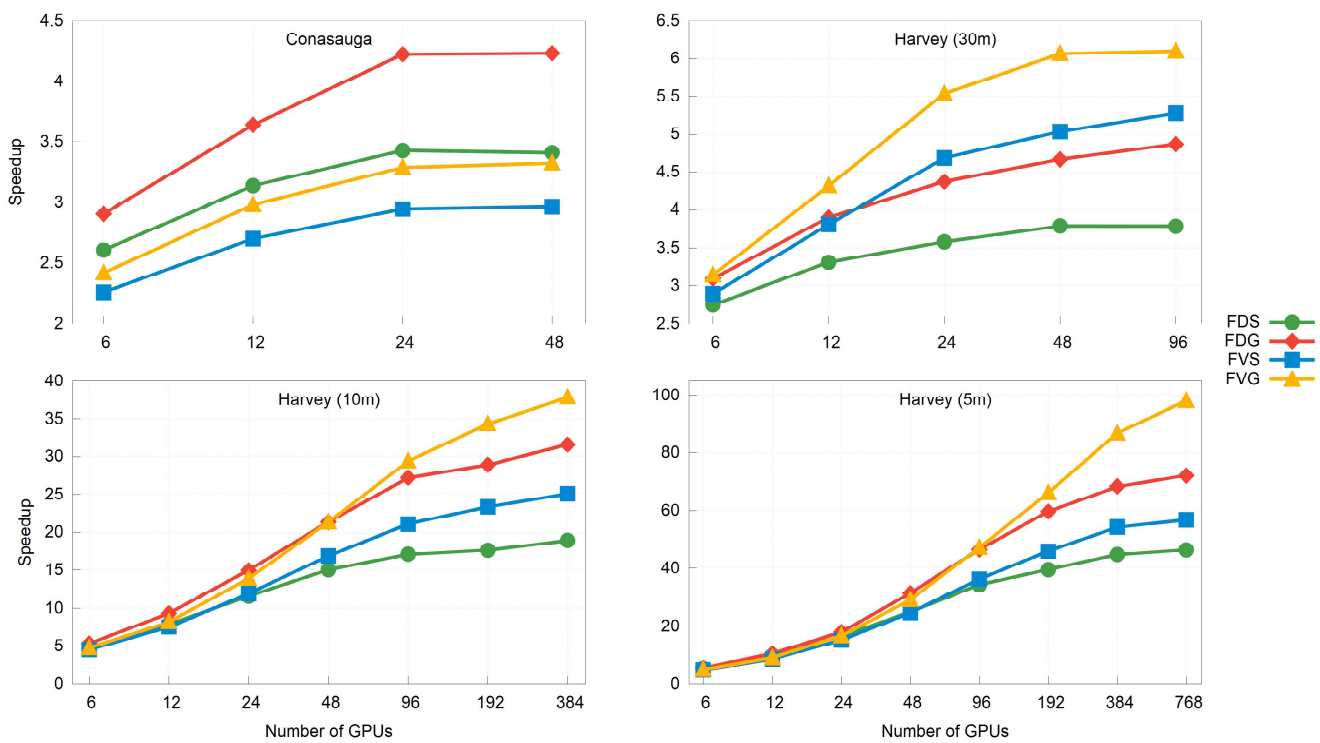


Figure 9: Speedup of test cases with an increasing number of GPUs

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] [n.d.]. www.olcf.ornl.gov/olcf-resources/compute-systems/summit/
- [2] 2014. MIKE 11 A Modelling System for Rivers and Channels Reference Manual. <https://www.mikepoweredbydhi.com/products/mike-11>
- [3] 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.
- [4] André R Brodtkorb, Martin L Sætra, and Mustafa Altınakar. 2012. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids* 55 (2012), 1–12.
- [5] Gary W Brunner. 1995. *HEC-RAS River Analysis System. Hydraulic Reference Manual. Version 1.0*. Technical Report. Hydrologic Engineering Center Davis CA.
- [6] Gary W Brunner. 2016. *HEC-RAS River Analysis System. Hydraulic Reference Manual. Version 5.0*. Technical Report. Hydrologic Engineering Center Davis CA.
- [7] US Costliest. 2018. Tropical Cyclones Tables Updated. *United States National Hurricane Center* (2018).
- [8] Norman H Crawford and Ray K Linsley. 1966. Digital Simulation in Hydrology: Stanford Watershed Model 4.
- [9] 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.
- [10] 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.
- [11] TT Dullo, S Gangrade, AJ Kalyanapu, SC Kao, SK Ghafoor, and KJ Evans. 2018. High-resolution modeling of Hurricane Harvey Flooding for Harris County, TX using a calibrated GPU-accelerated 2D Flood Model. In *AGU Fall Meeting Abstracts*.
- [12] Tigstul 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-Tallahapoosa River Basin. In *AGU Fall Meeting Abstracts*.
- [13] 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.
- [14] R Gommès, J Du Guerny, F Nachtergaele, and R Brinkman. 1998. Potential impacts of sea-level rise on populations and agriculture.
- [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, 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] 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.
- [22] YG Lai. 2008. SRH-2D version 2: Theory and User’s Manual. *Sedimentation and River Hydraulics—Two-Dimensional River Flow Modeling, US Department of Interior, Bureau of Reclamation, November* (2008).
- [23] 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, 363–370.
- [24] 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.
- [25] Ryan Marshall, Sheikh K Ghafoor, Alfred J Kalyanapu, Mike Rogers, and Tigstul 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, 21–29.
- [26] 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.
- [27] Mario Morales Hernandez, Md Bulbul Sharif, Sudershan Gangrade, Tigstul 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*. Technical Report. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).
- [28] 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.
- [29] Sreeram Potluri, Khaled Hamidouche, Akshay Venkatesh, Devendar Bureddy, and Dhableswar K Panda. 2013. Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with NVIDIA GPUs. In *2013 42nd International Conference on Parallel Processing*. IEEE, 80–89.
- [30] Martin Lilleeng Sætra and André Rigland Brodtkorb. 2010. Shallow water simulations on multiple GPUs. In *International Workshop on Applied Parallel Computing*. Springer, 56–66.
- [31] Luke S Smith and Qiuhua Liang. 2013. Towards a generalised GPU/CPU shallow-flow modelling tool. *Computers & Fluids* 88 (2013), 334–343.
- [32] 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.
- [33] 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).
- [34] 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.
- [35] 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.
- [36] Moisés Viñas, Jacobo Lobeiras, Basilio B Fragueta, 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.
- [37] Wikipedia contributors. 2019. Hurricane Harvey — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Hurricane_Harvey
- [38] 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.
- [39] 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.