This is a manuscript of an article that was accepted in 21 May 2018 for publication in the Journal of Environmental Modelling & Software. The final publication is available at https://doi.org/10.1016/j.envsoft.2018.05.007.

A Cloud-Based Flood Warning System for Forecasting Impacts to Transportation Infrastructure Systems

Mohamed M. Morsy[a,b], Jonathan L. Goodall[a*], Gina L. O'Neil[a], Jeffrey M. Sadler[a], Daniel Voce[a], Gamal Hassan[c], Chris Huxley [d]

[a] Department of Civil and Environmental Engineering, University of Virginia, 351 McCormick Road, PO Box 400742, Charlottesville, VA, 22908, USA

[b] Irrigation and Hydraulics Department, Faculty of Engineering, Cairo University, P.O. Box 12211, Giza 12613, Egypt

[c] Hassan Water Resources, PLC, 2255 Parkers Hill Drive, Maidens, VA, 23102, USA

[d] BMT WBM Pty Ltd, Level 8, 200 Creek Street, Brisbane, Queensland, 4000, Australia

Highlights:

- Design of a cloud-based flood warning decision support system
- Prototype of a cloud-based end-to-end system using Amazon Web Services (AWS)
- Produces near real-time results while strategically consuming resources
- Automates access to forecasted rainfall, 2D model run, and output visualization
- Uses GPUs to speedup 2D model run time by up to 80x compared to using a CPU

Abstract

The ability to quickly and accurately forecast flooding is increasingly important as extreme weather events become more common. This work focuses on designing a cloud-based real-time modeling system for supporting decision makers in assessing flood risk. The system, built using Amazon Web Services (AWS), automates access and pre-processing of forecast data, execution of a computationally expensive and high-resolution 2D hydrodynamic model, Two-dimensional Unsteady Flow (TUFLOW), and map-based visualization of model outputs. A graphical processing unit (GPU) version of TUFLOW was used resulting in an 80x execution time speed-up compared to the central processing unit (CPU) version. The system is designed to run automatically to produce near real-time results and consume minimal computational resources until triggered by an extreme weather event. The system is demonstrated for a case study in the coastal plain of Virginia to forecast flooding vulnerability of transportation infrastructure during extreme weather events.

Keywords:

Flood warning applications; cloud computing; 2D hydrologic model; GPUs; Amazon Web Services; reproducibility

Software Availability:

The software created in this research is free and open source. The software information and availability are as follows:

Developers: Mohamed Morsy, Daniel Voce, Gina O'Neil, and Jeffrey Sadler

Programming language: Python, Bash, HTML, JavaScript, Powershell, CSS

GitHub link: https://github.com/uva-hydroinformatics/FloodWarningModelProject

1. Introduction

Floods were the number one natural disaster in the US in terms of lives lost and property damage incurred during the 20th century (Perry, 2000), with statistics showing that total flood insurance claims averaged more than $1.9 billion per year from 2006 to 2015 (NFIP Statistics, 2016). Rainfall events are predicted to become more frequent and intense due to climate change, which is expected to cause increased flooding (Melillo et al., 2014). As society faces flooding events with increasing frequency and intensity, flood modeling will become an even more important tool for decision makers. Such models can be used to warn municipalities and communities of forecasted flooding impacts, and to test alternative flood mitigation strategies for addressing flood problems.

The National Research Council (NRC) has recommended increased use of two dimensional (2D) hydrodynamic models for flood risk management purposes (NRC, 2009). There are several advantages to using 2D models rather than one dimensional (1D) models, such as better resolution of velocity, localized depth and surface water elevation, and determination of floodplain extent directly. 2D hydrodynamic models are especially important for cases with complex flows such as in low-relief terrains with flat or mild slopes. For these low-relief terrains, 1D models are often not sufficient due to the limitations of assumed uniform water velocity and constant water surface elevation modeled on each cross section (Garcia et al., 2015).

Executing 2D hydrodynamic models at the regional scale (~$10x10^3$ - $100x10^3$ km^2) requires parallel computation in order to run in a timeframe reasonable for flood warning applications. Graphical processing units (GPUs) have recently been shown to effectively execute parallelized 2D hydrodynamic models, with observed speed-ups of 20x to 100x (Huxley and Syme, 2016; Garcia et al., 2015; Vacondio et al., 2014; Kalyanapu et al., 2011). Vacondio et al. (2014) expect

that GPUs will continue to be attractive for 2D numerical models compared to clusters of central processing units (CPUs) for several reasons: (i) fast-developing GPU hardware, (ii) quickly decreasing costs, and (iii) lower maintenance compared to large CPU clusters. With the speed-ups provided by GPUs, regional flood warning systems can now be implemented with 2D hydrodynamic models and the spatial resolution needed to provide targeted and detailed information to decision makers.

There are currently several related efforts aimed at improving flood warnings. The National Weather Service (NWS) and the United States Geological Survey (USGS) have a joint project to generate flood inundation maps at locations where a NWS forecast point and a USGS stream gauge exist (Fowler, 2016). At these locations, a flood inundation map is created for multiple possible water surface elevations and, by using a rating curve and forecasted discharge, the data is converted into the corresponding water surface elevation. Then the corresponding flood inundation map is selected from a precomputed library of flood inundation maps. The Flood Locations and Simulated Hydrographs (FLASH) is a system designed by researchers and developers to improve the ability of the NWS to forecast flooding at the Weather Forecast Offices (WFOs) (Gourley et al., 2017). The FLASH system uses the Multi-Radar Multi-Sensor (MRMS) rainfall data with a spatial resolution of 1 km and temporal resolution of up to 2 min, along with a highly efficient distributed hydrologic modeling framework, to generate flood forecasts for over 10.8 million grid points across the conterminous United States (CONUS). The National Flood Interoperability Experiment (NFIE) is a multiagency effort in collaboration with the academic community to improve river and flood forecasts (Maidment, 2017). A key component of NFIE is a model called Routing Application for Parallel computing of Discharge (RAPID) (http://rapid-hub.org/), which was developed to operate on the 2.67 million NHDPlus catchments and use parallel computing to

solve the 1D Muskingum flow equations on this large river network (Maidment, 2017; David et al., 2013, 2011). NFIE showed it was possible to increase the spatial density of flooding forecast locations by more than 700x compared to the present NWS river forecast system (Maidment, 2017). The RAPID model is expected to be replaced by the Simulation Program for River Networks (SPRINT) (Liu and Hodges, 2014), which has the capability of solving the full nonlinear Saint-Venant equations for 1D unsteady flow and depth for channel networks, and promises speed-up of the computation time. However, in some instances a 2D flood model will be necessary to accurately model water transport over large flat areas. Delft-FEWS is a hydrological forecasting and warning framework that provides a platform through which operational forecasting systems can be constructed, allowing for flexibility in the integration of models and data (Werner et al., 2013). Delft-FEWS does not contain any inherent hydrologic modeling capabilities within its code base, instead it relies on the integration of external hydrologic model components.

The objective of this research is to design and prototype a cloudbased system to support decision makers as they assess flood risk to transportation infrastructure during extreme weather events. The system automates access and pre-processing of forecast data, execution of a high-resolution 2D hydrodynamic model, and map-based visualization of model outputs. This work advances on prior approaches described earlier by presenting a cloud-based framework for modeling regions with complex flows using a 2D hydrodynamic model. Rather than relying on precomputed flood maps, flood depths and extents, this approach allows for water flows to be modeled in real-time based on current and forecasted conditions. This approach could be adopted in existing decision support systems to leverage cloud and GPU resources within this general framework.

The study advances on previous work funded by the Virginia Department of Transportation (VDOT) for the Hampton Roads District of Virginia, which produced the Regional River Severe Storm Model (R^2S^2) (Hassan Water Resources PLC., 2012). The purpose of R^2S^2 is to help Residency Administrators to efficiently allocate scarce resources to road closures and to assist first responders with entering and exiting flood prone areas. This research advances R^2S^2 by automating what was previously a manual process of converting forecast rainfall data into model inputs, running the model, and visualizing the results. Furthermore, this research addresses computational challenges with using R^2S^2 for real-time flood warning and emergency management applications. This research also moves R^2S^2 to the cloud and is one of the first cloudbased flood warning applications with (i) an automated workflow for obtaining the real-time forecast rainfall data, (ii) execution of a model to identify flooded bridge and culvert locations in a time duration sufficient for warning and emergency management purposes, and (iii) generation of an online map with locations of the flooded roadways and bridges, and the ability to send automated warning messages via email. This system can provide VDOT with information needed when determining road closures, disseminating warning messages for area residents, and making other emergency management decisions that affect human safety and property damage. Although the current case study application of the system is focused on VDOT as the primary user, the approach could be used as a more general flooding decision support system for other stakeholders.

Cloud computing is gaining attention in environmental applications to satisfy the peak performance needs of applications that use large amounts of processing power (Granell et al., 2016). Sun (2013) used Google Drive, a cloud computing service, to host an environmental decision support systems (EDSS) module that is migrated from the traditional client-server-based architecture. Google Drive has the capability of providing a number of basic visual analytics

features, and the collaboration between the decision makers can be increased while decreasing the cost of small scale EDSS. Ercan et al. (2014) used the Windows Azure Cloud environment to run a created calibration tool built with the modified calibration method, a parallel version of the Dynamically Dimensioned Search (DDS), for calibrating the Soil and Water Assessment Tool (SWAT). Using this tool, results showed a significant speed-up of the model calibration for six different model scenarios. Wan et al. (2014) introduced a public cloud-based flood cyberinfrastructure, CyberFlood. CyberFlood collects, organizes, manages, and visualizes several global flood databases for decision makers and public users in real-time. This database is expanded by applying a methodology for the data collection in which the public reports new flood events using smartphones or web browsers. Hu et al. (2015) implemented a web-based application in the Hadoop-based cloud computing environment to make enhanced coupled human and natural models publicly available. This allows users to access and execute the model without an increase in response time. Kurtz et al. (2017) presented a stochastic cloud-based fully-operational architecture for a real-time prediction and management system related to groundwater management. This proposed system allows for data assimilation and is coupled with a physically based hydrologic model, HydroGeoSphere, in a cloud environment to use the generated prediction for groundwater management. The work presented here advances on prior studies by demonstrating the ability of using resources in a public cloud, including instances with powerful GPUs like those provided by AWS, to build an end-to-end automated cloud-based system for regional-scale flood forecasting. This system is able to run a computationally expensive 2D hydrodynamic model and is activated automatically during extreme weather events by software that is continuously monitoring forecasted rainfall conditions. It is designed to run in a time frame

relevant to realtime emergency management applications and automatically deliver model outputs to decision makers through online maps and email notifications.

The remainder of the paper is organized as follows. First, a Study Area section is presented to introduce the region where the model is applied. Second, the Data and Methods section is presented to outline the available data sources, the pre-processing steps used to translate this data for use in the model, steps taken to speed-up the model, and the post-processing steps used to automate the model output dissemination. Next, the Results and Discussion section presents a prototype of the software and the results of applying the system for an extreme weather event. Finally, the Conclusion section provides a summary of the key research outcomes and steps that could be taken to further advance this work.

2. Study Area

The study area is in a portion of the Chowan River basin that is within VDOT's Hampton Roads District in Virginia, USA and is approximately 5,780 km^2 (2,230 mi^2) (Figure 1). The study area includes the Meherrin, Nottoway, and the Blackwater Rivers. The longest flowpath along NHD flowline features is approximately 175 km (109 mi) with a slope that varies from nearly 0% to 21%. The study area includes 493 georeferenced VDOT bridges and culverts. Due to a high portion of the study area consisting of low-relief terrain, in the coastal plain, especially in the eastern part of the study area (Figure 2), the system utilizes a 2D hydrodynamic model called Two-dimensional Unsteady Flow (TUFLOW) (https://www.tuflow.com/) (Syme, 2001). The upstream portion of the project domain consists of relatively higher-relief terrain and, therefore, the Hydrologic Engineering Center-Hydrologic Modeling System (HEC-HMS), a lumped hydrology model that is less computationally intensive, was deemed appropriate to model flows in these

areas. By using HEC-HMS to generate inflow boundary conditions from the high-relief portion to the low-relief portion of the study area, the overall system runtime is kept smaller. Including these high-relief upstream watersheds, the project domain is approximately 11,000 km^2 (4,240 mi^2).

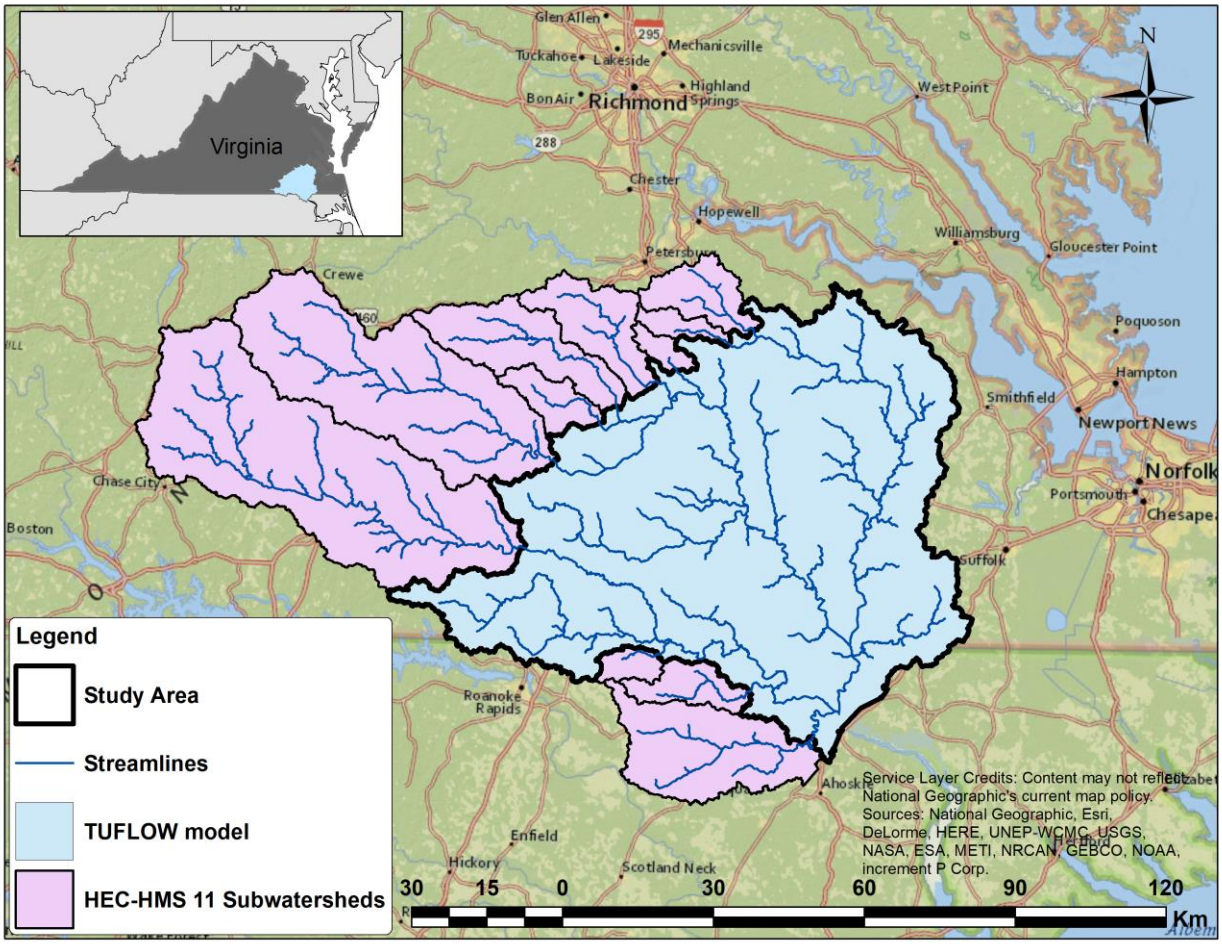


Figure 1 Model domain composed of the study area where the TUFLOW model is run and the 11 subwatersheds where HEC-HMS is used to model contributing inflow to the study area.

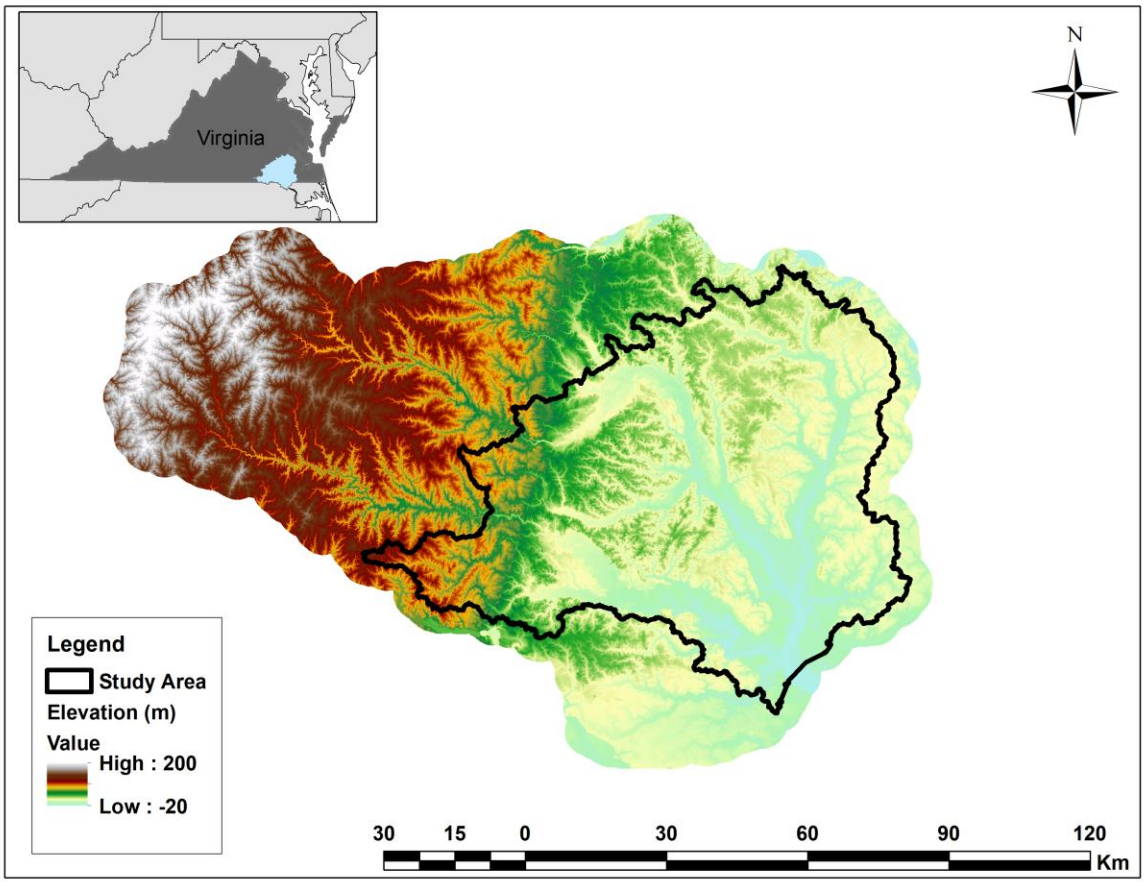


Figure 2 The digital elevation model (DEM) with 10 m resolution for the study area, including the 11 subwatersheds that contribute inflow to the study area.

3. Data and Methods

3.1. R^2S^2 System

The R^2S^2 system was first developed by Hassan Water Resources, PLC to integrate multiple datasets with sophisticated hydrodynamic models to provide flood risk prediction during severe storm events to the Hampton Roads District of VDOT (Hassan Water Resources PLC., 2012). The R^2S^2 system consists of software that processes the many input files required for the TUFLOW model, runs HEC-HMS to establish boundary conditions for TUFLOW, and processes output files from TUFLOW to determine inundated bridges and culverts (Figure 3). Constant input data for the R^2S^2 include a DEM with 10 m resolution, soil data (SSURGO, 2012), and land use

data with 30 m resolution. The observational data must be accessed and processed in real-time from federal data providers. R^2S^2 uses real-time products for rainfall including National Oceanic and Atmospheric Administration (NOAA) gauges (see Figure 8) and Next Generation Weather Radar (NEXRAD) radar data, where available for specific storms. The rainfall data is used as inputs for both R^2S^2 hydrologic models. First, the HEC-HMS model uses the rainfall data to generate the hydrograph for each of the 11 subwatersheds that border the study area. Then the 11 outlet hydrographs generated are applied to the TUFLOW model as boundary conditions along with the rainfall data to generate water levels throughout the study area. The model is calibrated and evaluated using historic stream gauge data. Eventually, real-time stream data will be used to set initial conditions.

3.2. Rainfall Forecast Data Preparation

A preliminary step for building a flood warning system is to identify and automate the pre-processing of the forecast rainfall product. In this study, the procedure to collect and process the forecast rainfall data for model input was automated to reduce human translation errors and decrease the time between when new rainfall forecasts are available and new water level forecasts can be generated. Both the TUFLOW and HEC-HMS models in R^2S^2 require input rainfall data, but in different formats. TUFLOW has three approaches for applying the rainfall directly to the computational cells: (i) polygons covering multiple cells assigned as rainfall time series, (ii) gridded rainfall created as ASCII files for each time step or as one NetCDF file, and (iii) a rainfall control file that allows a user to specify point time series over the model domain and specify how the rainfall is interpolated to the model cells. HEC-HMS has two approaches for applying the rainfall data: (i) a rainfall time series for each basin stored in a data storage system (DSS) file that is prepared by HEC-DSSVue, a program for viewing, editing, and manipulating DSS files

(CEIWR-HEC, 2009), and (ii) gridded rainfall that is prepared by HEC-GridUtil, a utility program for managing gridded data with HEC-DSS (Steissberg and McPherson, 2011).

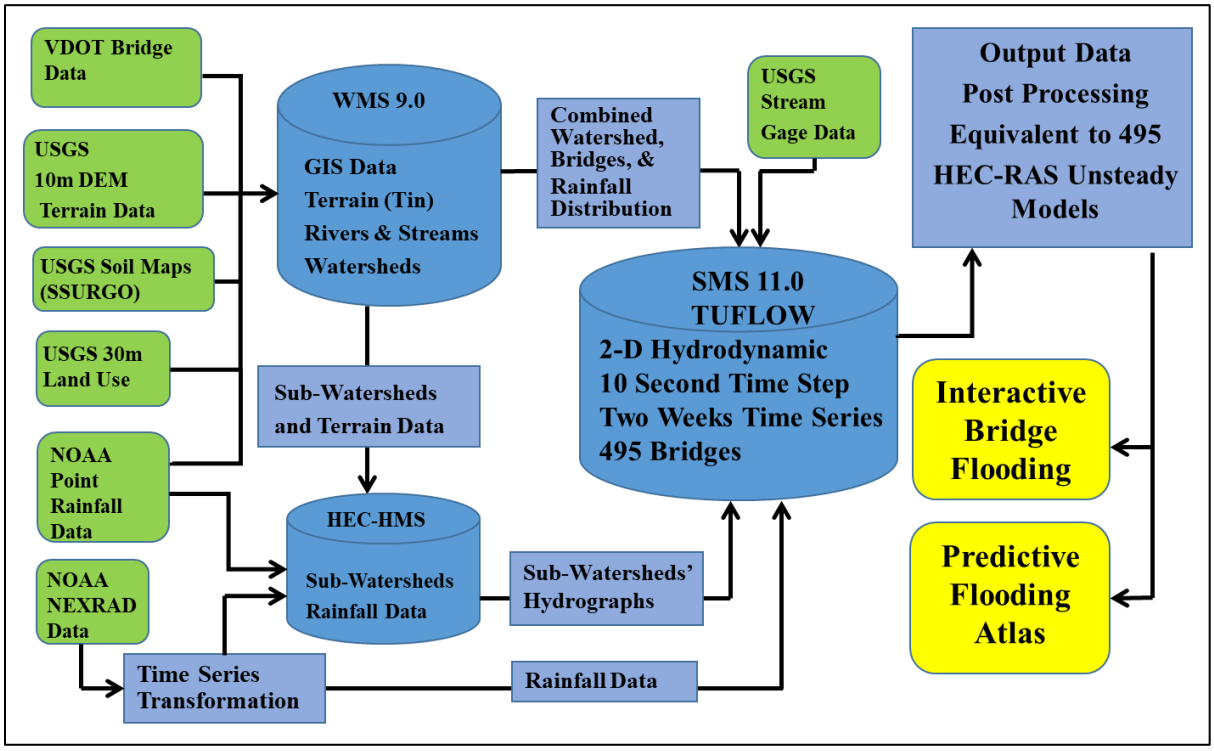


Figure 3 R^2S^2 workflow.

The identification of appropriate forecast datasets focused on NOAA rainfall products that are in a grid format and can be quickly accessed for real-time flood warning applications. Several potential forecast datasets were identified for the study region, including products from the Rapid Refresh (RAP), the High-Resolution Rapid Refresh (HRRR), the North American Mesoscale Forecast System (NAM), and the National Digital Forecast Database (NDFD). The RAP, HRRR, and NAM products are all provided by the National Center for Environmental Prediction (NCEP) and the NDFD is provided by the National Weather Service (NWS). These forecast products were compared in terms of their spatial resolution, temporal resolution, and frequency of model initiation (i.e., model cycle). Results of this comparison and the code written to automate the retrieval and reformatting of the rainfall data to meet the requirements of the TUFLOW and HEC-

HMS models are presented in Section 4.1. This code is tested to retrieve the required rainfall data with the desired formats for the two models and with the correct spatial and temporal resolution.

3.3. Speeding-up R^2S^2 execution

TUFLOW is the computational bottleneck within the overall R^2S^2 workflow. Using a CPU for computation takes more than three days to run a 15-day simulation period, the duration over which Hurricane Sandy caused high flows in the Study region. The use of multiple CPUs and GPUs has been investigated as a means of speeding-up 2D hydrodynamic models (Kalyanapu et al., 2011; Brodtkorb et al., 2012; Rostrup and Sterck, 2010; Castro et al., 2011; Lacasta et al., 2013; Sanders et al., 2010; Garcia et al. 2015). As stated in the introduction, GPU use offers the performance of smaller clusters at a much lower cost (Jacobsen et al., 2010). Therefore, GPUs were investigated for speeding-up the TUFLOW model rather than using CPU clusters.

TUFLOW comes with a GPU Module capable of operating on multiple GPUs in parallel. We explored the use of both local and Amazon Web Services (AWS) resources for GPU computations. The TUFLOW GPU Module uses an explicit scheme only, while the TUFLOW CPU solver (TUFLOW Classic) uses an implicit scheme. Explicit schemes could be less numerically stable compared to implicit schemes if using the same time step and grid cell size. Also, explicit schemes require a small time step and high resolution grid cell size to compete with well-developed implicit schemes (Boris, 1989; Anderson and Wendt, 1995; Tóth et al., 1998; Pau and Sanders, 2006; Zhao, 2008). The differences between these two schemes could be large and need to be checked for consistency.

Two local GPU resources with different capabilities were explored (Table 1). M1 is a machine with a modest GPU and other resources typical of most desktop computers. M2 is a high-

end workstation with 64 GB of RAM and two NVIDIA GeForce Titan Graphics cards. There are several types of AWS Elastic Compute Cloud (EC2) instances designed for GPU-based computations. There are two sizes of G2 instances, which have lower-end GPUs, and three sizes of P2 instances, which have higher-end GPUs (Table 2). The properties and hourly fees for these instances vary, as shown in Table 2.

Table 1 Local machines with GPUs used to investigate TUFLOW model execution times.

ID	Type	CPU	RAM (GB)	GPU GPU RAMInfo
M1	Desktop Dell OptiPlex 990	3.40 GHz, 4 Core(s)	16	NVIDIA Quadro K2000, 2.00 GB of GPU memory, 384 SMX CUDA parallel processing cores, clock speed of 954 MHz
M2	Desktop Viz Lab ESCHER	3.20GHz, 3201 Mhz, 6 Core(s)	64	Two units of NVIDIA GeForce GTX TITAN, each with 6.00 GB, 2688 CUDA parallel processing cores, clock speed of 837 MHz for each

Several tests were performed to measure the TUFLOW model execution times using the AWS EC2 g2.8xlarge and p2.8xlarge instances. The TUFLOW model with a 50 m grid cell size was used for these tests. The g2.8xlarge instance, which has 4 GPUs, was used to execute the model with 1, 2, 3, and 4 GPUs. Likewise, the p2.8xlarge instance, which has 8 GPUs, was used to execute the model with 1 through 8 GPUs. Each of these model runs was performed twice to ensure that model runtimes were consistent. Following this, comparisons of results generated from CPU and GPU solvers were performed. Lastly, using the GPU solver preliminary calibration steps were performed by varying grid cell size and input parameters.

Table 2 Comparison between G2 and P2 EC2 instances performance and costs as of 06/06/2017.

EC2 Instance	Model	GPUs	vCPU	Memory (GiB)	GPU Info	Storage (GB)	Hourly Fee
G2	g2.2xlarge	1	8	15	NVIDIA GRID K520 GPUs, each with 4.00 GB of GPU memory, 1536 CUDA parallel processing cores, clock speed of 800 MHz	SSD 1 x 60	$0.767
	g2.8xlarge	4	32	60		SSD 2 x 120	$2.878
P2	p2.xlarge	1	4	61	NVIDIA K80 GPUs, each with 12.00 GiB of GPU memory, 2496 CUDA parallel processing cores, clock speed of 875 MHz	EBS	$1.084
	p2.8xlarge	8	32	488		EBS	$8.672
	p2.16xlarge	16	64	732		EBS	$17.344

3.4. Post-processing and Automating Model Output Dissemination

The last main step for the flood warning system is to post-process and automatically disseminate the system output. One of the most important outputs from the TUFLOW model for this study is the maximum water level at each computational cell within the study area throughout the simulation duration. Using these maximum water levels and the VDOT bridge locations and deck elevations, a post-processing workflow was created and tested to automate email notifications providing bridges expected to be overtopped based on model projections.

In addition to sending email alerts, map-based visualizations can be created to display flooded bridge locations with flooded depths over the model domain. We explored and tested three options for creating such maps (Table 3). A first, and more basic, option involved a user manually uploading a keyhole markup language zipped file (KMZ) containing post-processed output from the model to the Google Maps website for visualization, providing a quick and simple method to visualize the flooded bridge locations. A second, more complex, option was to use Geosheets (https://www.geosheets.com/), an add-on to the GoogleSheets app that reads and displays post-processed tabular data stored in a Google Drive account as a Google Sheet. Unlike the manual upload of a KMZ file to Google Maps, this second method can be automated to dynamically update

the flooded bridge locations map. This option allows for some customization of map display without needing to configure and deploy a web server. The third and most complex option was developing a custom web interface using the Google Maps API to visualize the output KMZ files along with workflow run information (see Section 4.4). This alternative required the deployment of a web server and was therefore more complex; however, it provided the highest potential for customization and supported the dissemination of other workflow run information in addition to the flooded bridge locations. The post-processing workflow and three visualization options were tested to ensure the correct dissemination of the model output and the proper visualization for the decision-maker.

Table 3 Post-processing visualization options.

Visualization Options	Real-time	Input File	Need Web Server?	Customization
1. Google Maps Website	No	KMZ	No	Low
2. Geo Sheets	Yes	Google Sheet	No	Medium
3. Google Maps API	Yes	KMZ	Yes	High

3.5. Design of an automated flood warning system through AWS

After automating the retrieval of the forecast rainfall data, speeding-up the 2D model, and providing methods for post-processing and automating the model output dissemination, the final step was to create a seamless workflow using cloud services to link these individual components together without requiring intermediate user action. The goal of this automated workflow was to identify the flooded bridges and/or culverts in a time duration sufficient for warning and emergency management purposes based on the highest resolution, reliable rainfall forecast data and by using a publicly available cloud computing resources. Given that a single cloud instance capable of all of these tasks would be too expensive to continuously run, the design of this workflow had to meet several requirements: (i) a smaller, low-cost instance to monitor the rainfall

data for upcoming extreme events, and visualize model outputs (ii) a larger instance with NVIDIA GPU capabilities to accommodate and execute the hydrologic models and other processing scripts and (iii) a storage resource to archive model inputs such as the processed rainfall data and model outputs for later analysis. The smaller instance would trigger the larger instance when a flood event is forecasted. This smaller instance would also assume the role of maintaining the website to display and disseminate the model output so that it can be continuously available. To automate these steps of the workflow, the GPU instance would need to execute a batch file that (i) runs the pre-processing scripts to prepare the rainfall data, (ii) runs the hydrologic models, (iii) runs the post-processing script for preparing the model output for dissemination, (iv) sends outputs to other cloud resources for archiving and visualization, and (v) removes the model output files from the GPU instance.

Each individual step in the workflow (i.e. preprocessing the rainfall data, speed-up the model, and post-processing and disseminating the model output) was tested separately as mentioned in the previous subsections. Then the entire workflow was tested together locally and remotely using AWS resources. To test locally, the batch file that initiates the workflow was run along with the workflow scripts and hydrologic models that were placed on a local machine with NVIDIA GPU capabilities. Then, the batch file was test to run the whole workflow seamlessly. Then a low-cost smaller instance was created to monitor the forecast rainfall data. The smaller machine was then set up to monitor forecasted rainfall and to trigger the local machine to run the batch file and receive post-processed output for visualization purposes. After performing several tests of the designed system locally, the batch file and the hydrologic models were placed in a larger AWS instance with NVIDIA GPU capabilities. Then the smaller machine was linked to this

larger machine with an adjustment to the rainfall threshold to start the larger machine. The AWS-based system performance was then monitored and analyzed to ensure it was working as expected.

4. Results and Discussion

4.1. Rainfall Forecast Data Preparation

Comparison of the spatial resolution, temporal resolution, and model cycle of each dataset (Table 4) shows that HRRR was the best forecast rainfall product for our purposes. HRRR is a weather prediction system composed of a numerical forecast model and an analysis/assimilation system to initialize the model. HRRR is a higher-resolution model nested inside the hourly updated RAP. Although RAP can provide upper-level analyses and short-range forecasts, HRRR is best used to examine surface and near-surface parameters, such as surface precipitation. The HRRR model is run every hour of the day and forecasts out to 18 hours on a 1 hour time-step for each cycle. It provides a surface total precipitation product in units of mm of precipitation depth at a horizontal resolution of 3 km (NOAA, 2012). Surface total precipitation can be accessed as gridded data with dimensions of longitude, latitude, and time. Longitude and latitude are provided in the World Geodetic System (WGS) 1984 coordinate system, and time is in units of decimal days since 1-1-1 00:00:0.0 (NOAA, 2017a). HRRR data are distributed as part of the NOAA Operational Model Archive and Distribution System (NOMADS) project, a network of data servers that use the Open Source Project for a Network Data Access Protocol (OPeNDAP) (NOAA, 2017a). Although the HRRR data was selected as the primary input to the model, the system could alternatively use the coarser Quantitative Precipitation Forecast (QPF) from the NDFD dataset, which forecasts rainfall for the upcoming 72 hours, to monitor for large rainfall events beyond the 18 hours horizon captured by HRRR, and thus allowing for a longer lead time for preparing for

severe storms. The use of higher resolution rainfall forecast data with a longer lead time will reduce the uncertainty of the model, making it a more useful decision support tool, and the system is developed in a flexible way that easily enables the application of this better forecast data that may be available in the future.

Table 4 Comparison of available forecast datasets.

Dataset	Data Provider	Relevant Data Product	Resolution		Forecast (hrs)	Model Cycle
			Spatial (km)	Temporal (hrs)		
HRRR	NCEP	Surface total precipitation	3	1	18	24/day
RAP	NCEP	Surface total precipitation	13	1	18	24/day
NDFD	NWS	Quantitative precipitation forecast	5	6	72	8/day
NAM	NCEP	Surface total precipitation	12	1	36	4/day

Figure 4 shows the workflow for downloading and reformatting the forecast rainfall data. Pydap, a pure Python library client for OPeNDAP servers, is used to retrieve the desired forecast data for the study area. The automated workflow consists of three main parts: (i) access the latest available forecast data from the HRRR database, (ii) retrieve the forecast surface total precipitation with a horizontal resolution of 3 km in WGS 1984 coordinate system, and (iii) reformat the forecast data for model input in the NAD83 UTM 18N projected coordinate system. These rainfall data are reformatted as gridded rainfall data for TUFLOW using the Geospatial Data Abstract (GDAL/OGR) Python library, and as subwatershed time series for HEC-HMS using HEC-DSSVue, Python, and Java libraries. To include these direct rainfall data in TUFLOW, a TUFLOW Event File (TEF) was created to define the storm event properties. Using the new TEF file, the user can run the model for a given storm event using either historic or forecast data.

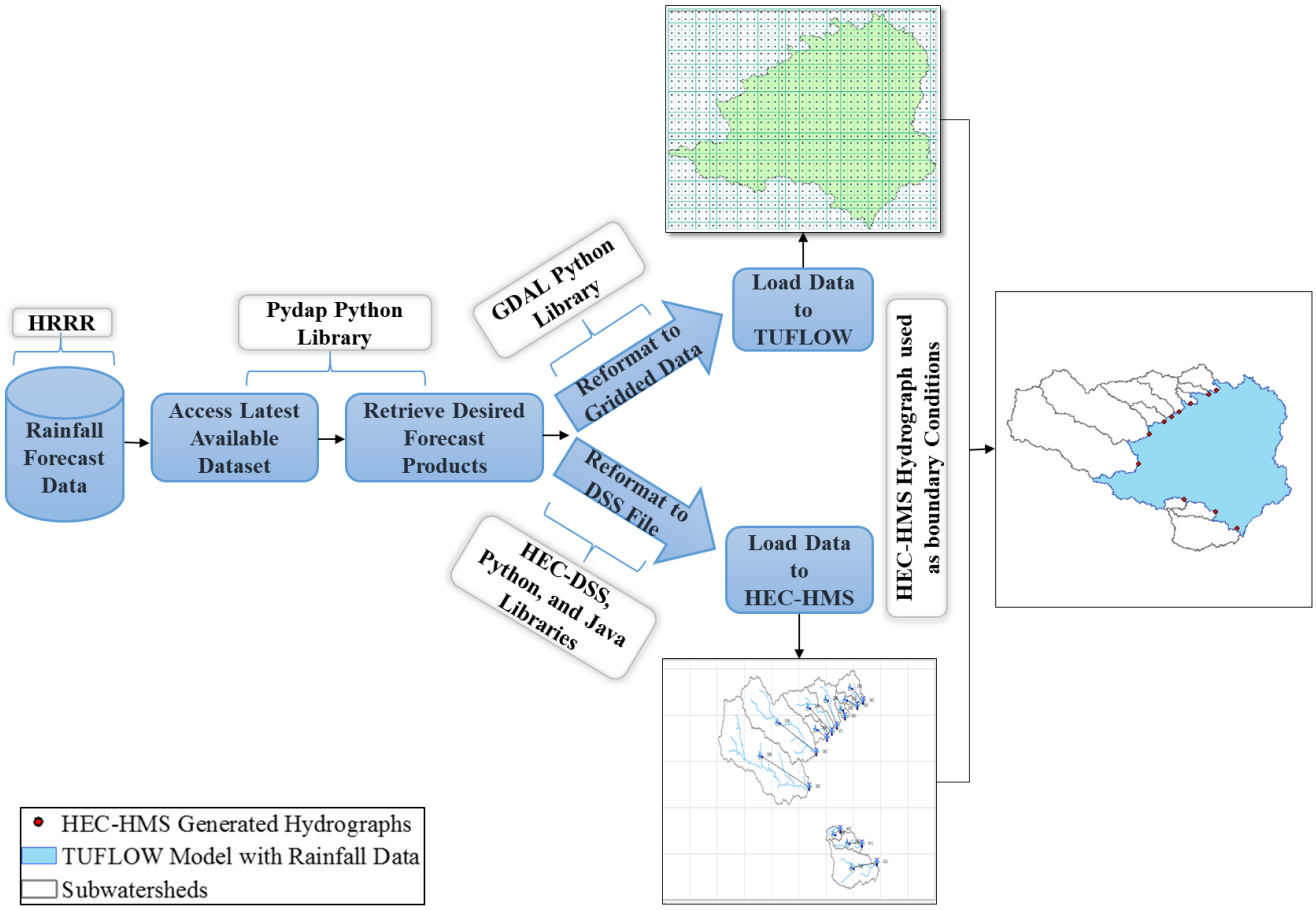


Figure 4 Forecast data workflow from HRRR to R^2S^2 sub-models, TUFLOW and HEC-HMS.

4.2. Speeding-up R^2S^2 execution

The model speed-up was evaluated using rainfall from Hurricane Sandy as input. The rainfall lasted for four days and the total modeled time span was 15 days (October 28 - November 11, 2012). Table 5 summarizes the results of the three TUFLOW model scenarios using the M1 and M2 machines (see Table 1). Using the CPU, the model took 120 hours to execute, and using the modest GPU in the M1 machine, the model took 11.5 hours to execute (10x speed-up compared to the CPU). Using the two more powerful GPUs in the M2 machine, the model took only 2.4 hours to execute (50x speed-up compared to the CPU and 5x speed-up compared to the M1

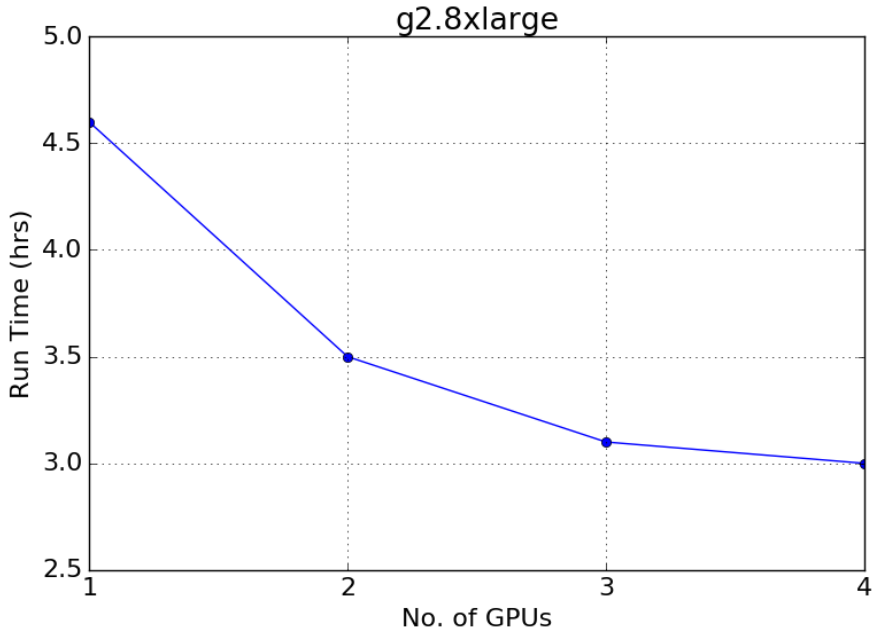
machine using the single GPU). The input timestep did not have a significant effect on the execution time when using GPUs, which is due to the explicit scheme within the TUFLOW GPU module that takes the input time-step value as an initial value and then optimizes the time-step to meet the convergence condition (i.e., courant number ≤1) (BMT WBM, 2016) .

Table 5 Comparison of CPU versus GPU speed-up using local GPU resources (differences bolded in each scenario).

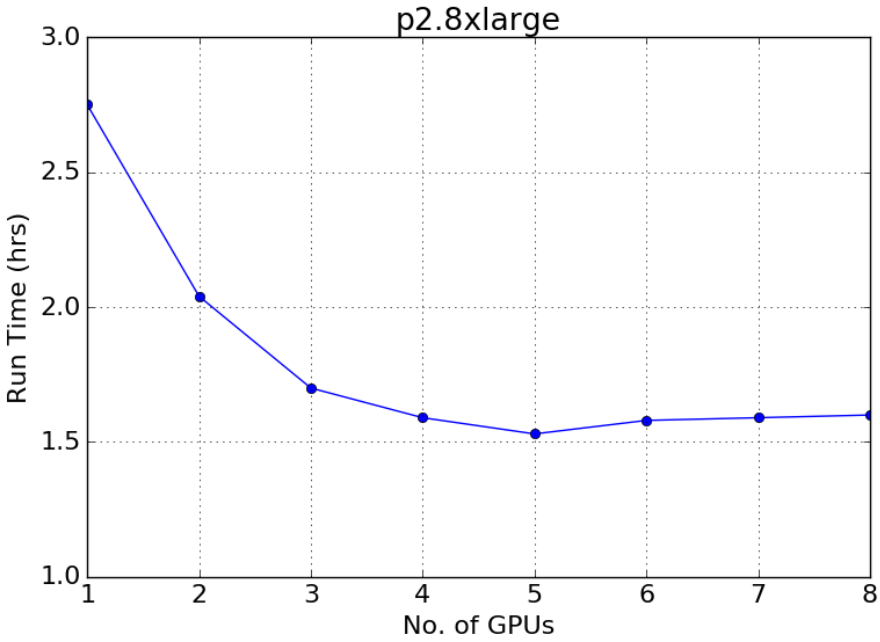
Model Specifications	**Run Scenarios**		
Machine	M1	M1	**M2**
Processing Unit	CPU	**GPU**	**GPU**
No. of GPUs	-	**1**	**2**
Time-step (sec)	10	10	10
Output Cell Size (m)	25	25	25
Running Time (hrs)	**120**	**11.5**	**2.4**

A test was also conducted to determine how increasing the number of GPUs influenced model execution time (Figure 5). As expected, running the model by using different numbers of GPUs produced the same output results (i.e., no differences in the maximum water levels). Figure 5-a provides the results of this test using the GPU model and the AWS g2.8xlarge instance with different numbers of GPUs. Using the g2.8xlarge instance with one GPU, the model takes about 4.6 hours to run. Using the g2.8xlarge instance and increasing the number of GPUs, the minimum execution time is 3 hours when all four GPUs are used, which costs about $9 per run. Because only four GPUs were available on this instance, we were not able to test whether additional GPUs would continue to reduce the running time. Figure 5-b provides the results of this test using the GPU model and the AWS p2.8xlarge instance with different numbers of GPUs. Using the p2.8xlarge instance with one GPU, the GPU model takes 2.75 hours to run, which is less than

using the g2.8xlarge instance with 4 GPUs. This shows the benefit of the more modern GPUs in the P2 versus G2 EC2 instances. Using the p2.8xlarge instance and increasing the number of GPUs, the minimum execution time was found to be 1.5 hours, which is achieved when five GPUs are used. This run, with the minimum execution time of 1.5 hours, costs about $13 per run, which is about 1.5x more expensive than the g2.8xlarge instance run; however, it was 2x faster than the g2.8xlarge instance. Comparing this 1.5 hours execution time to the CPU execution time of 120 hours shows an 80x speed-up for the model. Using six or more GPUs on this instance increases the execution time compared to using five due to known tradeoffs caused by data transfers between parallel GPU units (Huxley and Syme, 2016).



(a)



(b)

Figure 5 Running TUFLOW model through AWS EC2 a) g2.8xlarge instance, and b) p2.8xlarge instance with different numbers of GPUs.

Because the CPU and GPU TUFLOW solvers use different numerical schemes, it is important to understand differences in their outputs (Figure 6). Figure 6 provides the differences in maximum water level (Max. WL) generated from executing the model using the CPU and the GPU solvers. The maximum difference in Max. WL across the study area was around 2.5 m (8 ft), with 87% of the computational cells having differences in the Max. WL less than 0.5 m (1.6 ft). Figure 7 shows the Max. WLs at each bridge location generated by executing the model using the CPU solver versus the GPU solver. The mean absolute error (MAE) of 0.48 m (1.6 ft) and the root mean square error (RMSE) of 0.78 m (2.6 ft) demonstrate a fairly significant difference. In this study, we did a preliminary sensitivity analysis by changing the model grid cell size and Manning coefficient values, but future research should investigate this difference more fully.

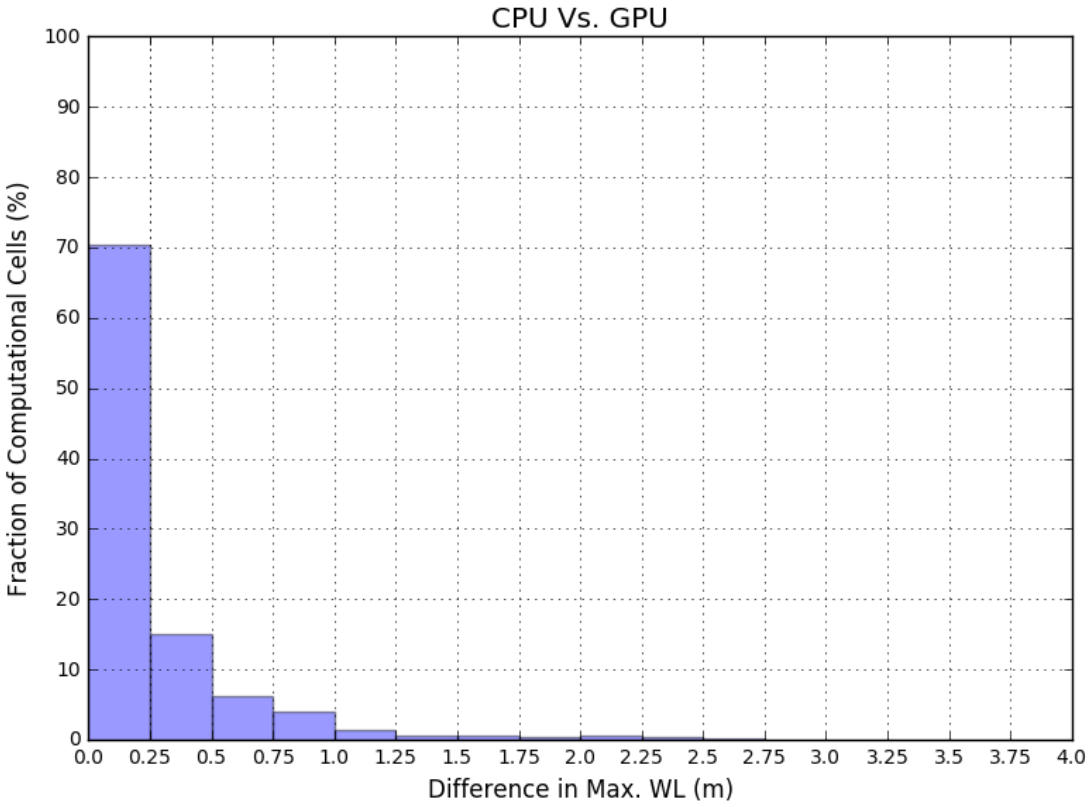


Figure 6 Differences between Max. WL generated from CPU solver and GPU solver.

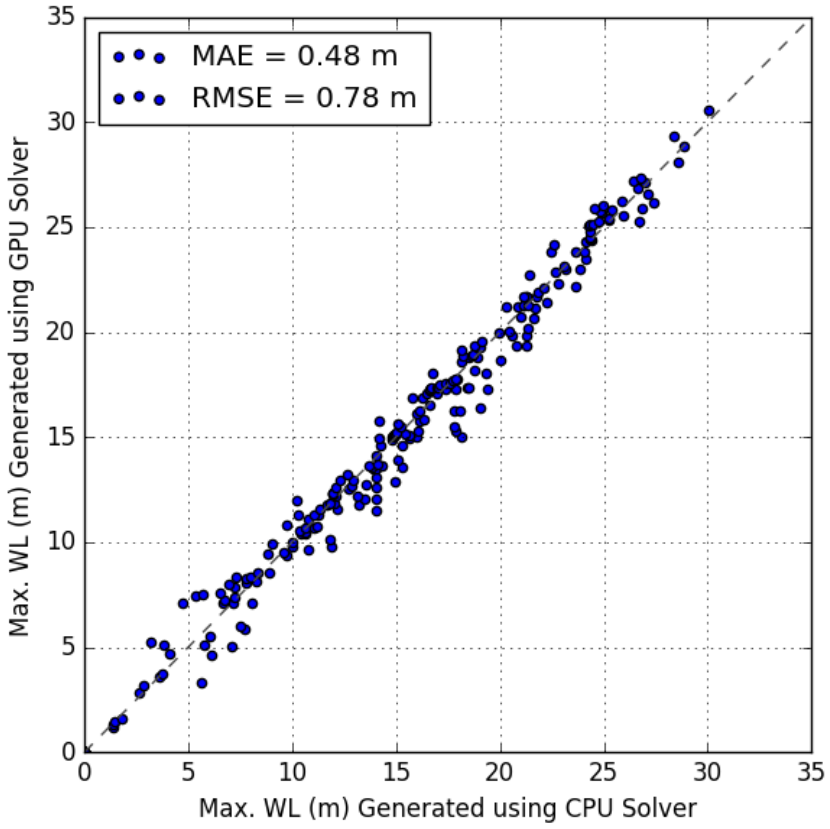


Figure 7 Max. WL at bridges and culverts locations generated from CPU solver versus GPU solver, with MAE of 0.48 m and RMSE of 0.78 m.

The model results using both the CPU and GPU solvers were compared against stream stage observations for the Hurricane Sandy event. Figure 8 and Table 6 show the USGS stations with data availability for the event. The USGS provided unpublished stage data that is considered provisional and, therefore, may contain erroneous or missing values due to instrument malfunction. This data was processed and cleaned to address this issue before being compared to the model output data. Figure 8 also shows the NOAA stations with the available recorded rainfall data for the Hurricane Sandy storm event. Hyetographs for this storm event at these stations are shown in Figure 9.

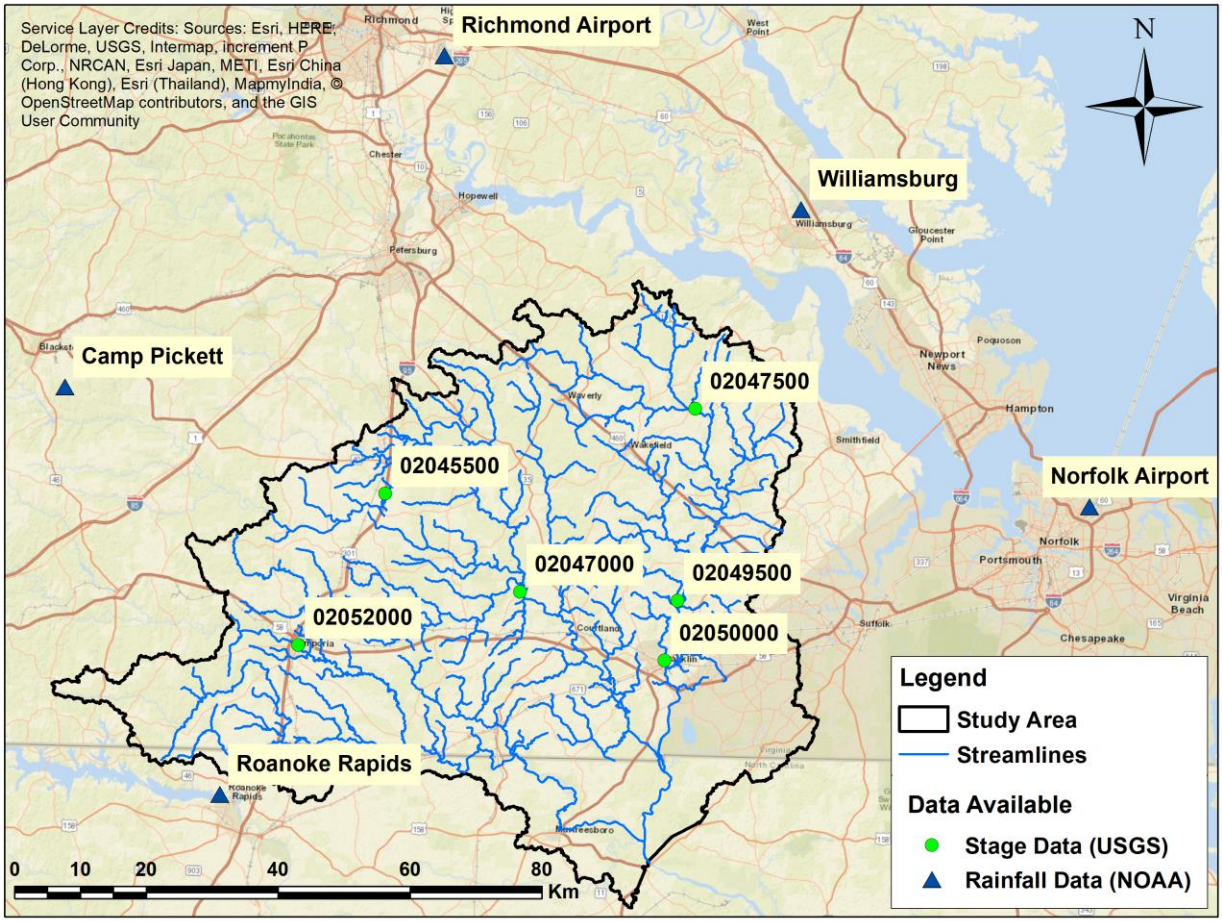


Figure 8 USGS and NOAA station locations that had Hurricane Sandy data availability in the study area.

Table 6 USGS stations in the study area with information about Hurricane Sandy availability.

Station	Name	Current Status	Stage		
			Parameter Code	Start Date	End Date
02049500	USGS 02049500 BLACKWATER RIVER NEAR FRANKLIN, VA	**Active**	00065 Gauge height	10/23/2016	2/20/2017
02047500	USGS 02047500 BLACKWATER RIVER NEAR DENDRON, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02045500	USGS 02045500 NOTTOWAY RIVER NEAR STONY CREEK, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02052000	USGS 02052000 MEHERRIN RIVER AT EMPORIA, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02047000	USGS 02047000 NOTTOWAY RIVER NEAR SEBRELL, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02050000	BLACKWATER RIVER AT HWYS 58/258 AT FRANKLIN, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017

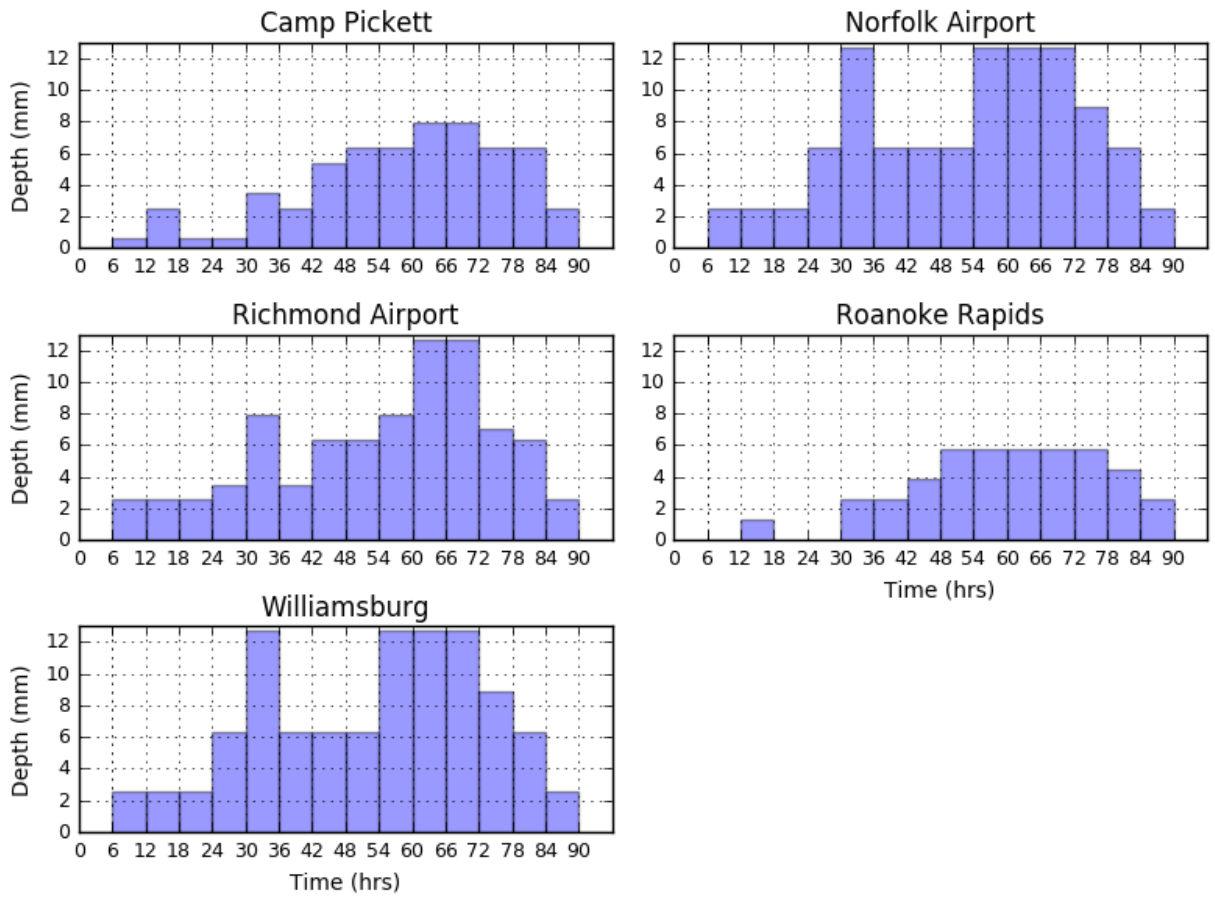


Figure 9 Hurricane Sandy hyetographs at the five NOAA stations near the study area (see Figure 8).

The finite volume schemes used by the 2D models are heavily dependent on the grid cell shape and size (LeVeque, 2002; Caviedes-Voullième et al., 2012). The TUFLOW model GPU solver uses only a Cartesian grid with the capability of changing the grid cell size. The TUFLOW model was executed using the GPU solver with grid cell sizes of 50 m, 40 m, 30 m, and 20 m. The output data from each of these runs were compared to the observed data at the six USGS stations and model results from CPU solver execution with a cell size of 50 m. The modeled peaks using the GPU solver with 50 m grid cell size were significantly higher than the observed data and the model peaks using the CPU solver at four USGS stations (02045500, 02047000, 02047500, and 02052000). However, at one of the USGS stations (02050000), the modeled peak using the GPU

solver with 50 m grid cell size was significantly lower than the observed data and the modeled peak using the CPU solver. Finally, at another USGS station (02049500), the modeled peak using the GPU solver with 50 m grid cell size was almost the same as the model peak using the CPU solver. However, both peaks were significantly lower than the observed data.

The differences between the modeled and observed peak stages could be due to the lack of adequate bathymetry data in the major rivers and tributaries. In all of the minor tributaries, and some stretches of the main rivers, bathymetry had to be assumed because of a lack of this data. This also could be due to the coarse DEM resolution (10 m) as the TUFLOW model extracts and utilizes the ground elevation at the model grid cell center for the GPU solver and at the model grid cell center and mid-sides for the CPU solver (BMT WBM, 2016). Calibration was also a challenge due to the scarcity of operating river gauges and limited available data for event-based calibration over such a large study area. In some instances, 2D models are not used due to the low resolution of the spatial data available and the difficulties faced when calibrating the model parameters (Caviedes-Voullième et al., 2012). This large study area includes only six USGS gauges that recorded stream stage during Hurricane Sandy. Three of these stations are located on the same main stream at the eastern part of the study area, one is in the middle of the study area, and the other two are located in the western part of the study area.

When the cell size of the model using the GPU solver decreases, a significant reduction in the peak stages was observed at four of the six USGS stations (02045500, 02047000, 02047500, and 02052000). At station 02050000, the modeled peak stage using the GPU solver increased with decreasing cell size, while at station 02049500 the peak stage remained nearly constant with each cell size. Decreasing model grid cell size improved the matching of observed peaks at four of the six observation sites and, therefore, we decided to use a smaller cell size in the model application.

The drawback of a smaller cell size is an increase in model execution time. Figure 10 shows the model execution time using the GPU solver with different grid cell sizes (50 m, 40 m, 30 m, and 20 m) for the M2 machine (see Table 1). Figure 10 also shows the MAE resulting from comparisons of model output generated using the GPU solver at different cell sizes and the model output generated using the CPU solver with the 50 m cell size. Based on these results, we chose the 30 m cell size since there is only a small difference between this scenario and the scenario resulting from the GPU solver with a 20 m grid cell size model and because there is a significant increase in the model runtime (2.8x from 10.2 hours to 28 hours).

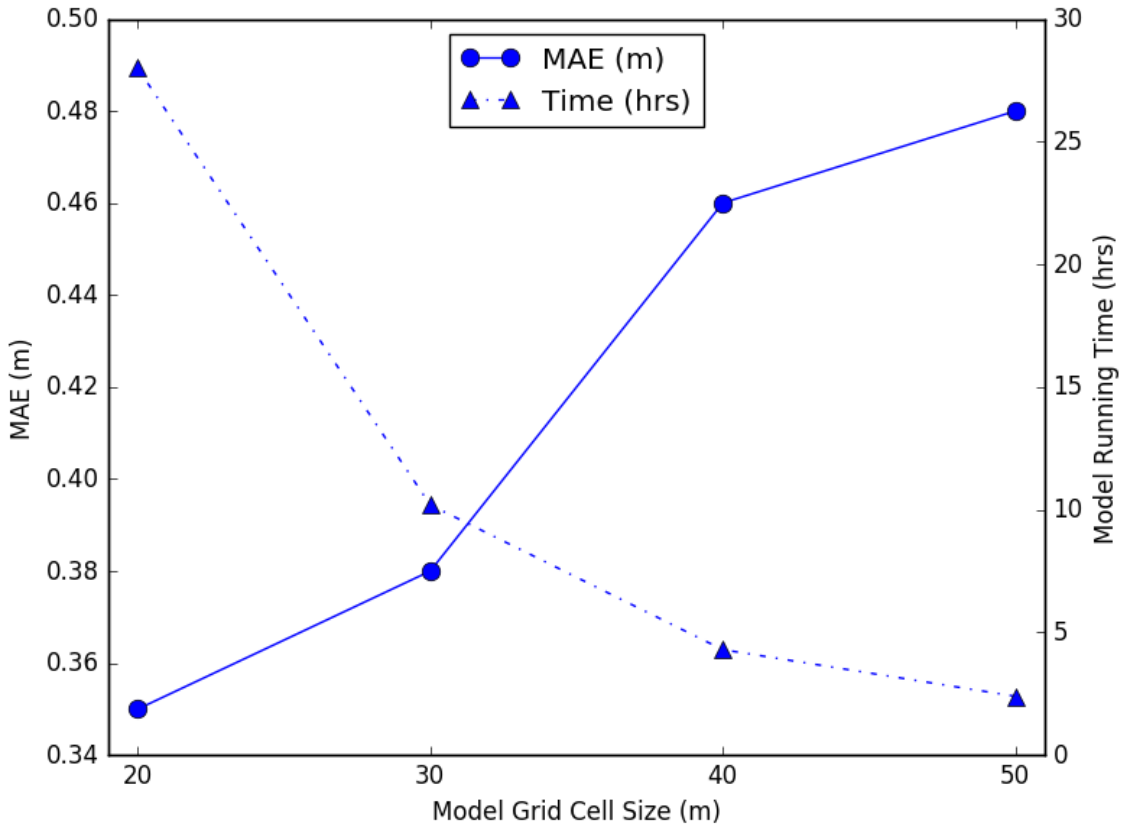


Figure 10 Model run time using GPU solver with different grid cell sizes and the corresponding MAE versus CPU solver using M2 machine (Table 1).

In addition to decreasing the cell size to 30 m, we also adjusted the Manning coefficient (n) to test its sensitivity and ability to improve matching of observed peak stages obtained from

the six USGS stations. The model initially had Manning coefficient values based on the study area land use. To assess the sensitivity of the model to changes in the Manning coefficient, this value was changed to be 0.6n, 0.8n, 1.0n, 1.4n, and 1.8n. As the Manning coefficient value decreased, the modeled peak stages became closer to the observed peaks at stations 02045500, 02047000, 20047500, and 02052000. After reducing the grid cell size from 50 m to 30 m and changing the Manning's coefficient from 1.4n to 0.6n, the model came the closest to matching observed peak river stage. This represents a preliminary calibration of the model that should be more fully explored through additional research.

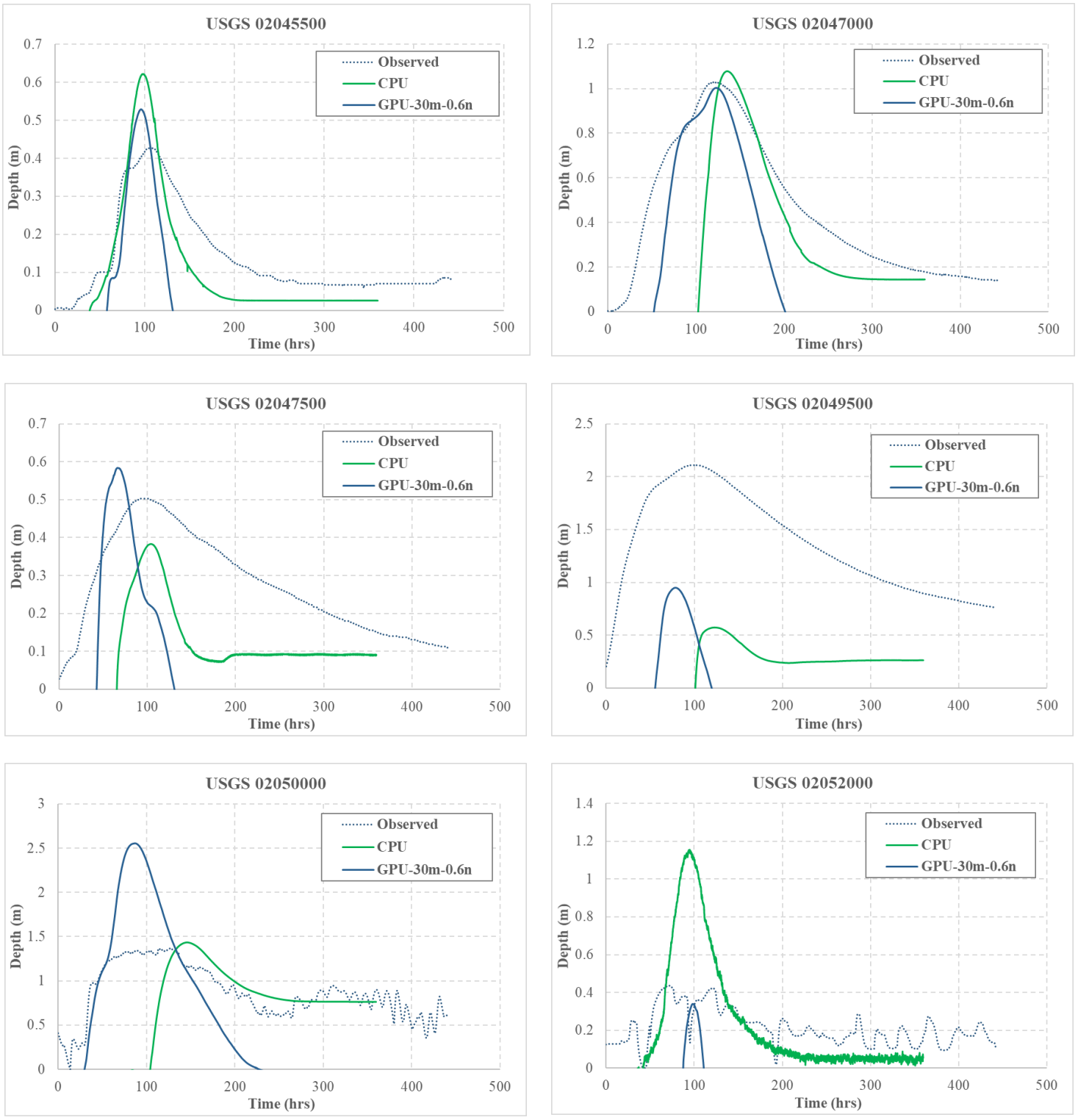


Figure 11 Comparisons between the observed stage depth data and the modeled depth generated from using a GPU solver with 30 m cell size and 0.6n Manning coefficient values.

The analysis of model response to changing the grid cell size and Manning's coefficient was done by applying rainfall time series for Hurricane Sandy from five rain gauges to polygons that each covered multiple model grid cells. TUFLFOW also has the capability of using direct rainfall data that applies input rainfall values to every cell in the 2D hydrodynamic model. When the rainfall is directly applied to the cells, the model routes flow based on the cell topography on a cell by cell basis (Huxley and Syme, 2016). Huxley and Syme (2016) investigated using this new method by applying the direct gridded rainfall data and found that GPU direct rainfall hydraulic modeling can be used as an alternative to runoff-routing hydrology modeling. To check the model behavior using the direct gridded rainfall data method with the chosen grid cell size and Manning's coefficient values, rainfall data from Hurricane Sandy was obtained from the Tropical Rainfall Measuring Mission (TRMM). This data has resolution of 0.25×0.25°, resulting in 16 cells covering the entire study area. We hoped to use rainfall data from NEXRAD, provided by NOAA, but there was no data available for the dates of Hurricane Sandy for our study area.

Table 7 Statistical analysis to compare the time to peak and peak stage depth shown in Figure 11

	USGS Station	Observation	CPU		GPU-30m-0.6n	
			Value	Relative Error (%)	Value	Relative Error (%)
Peak Stage Depth (m)	**02045500**	0.43	0.62	44.19	0.53	23.26
	02047000	1.03	1.08	4.85	1.01	-1.94
	02047500	0.5	0.38	-24.00	0.58	16.00
	02049500	2.11	0.57	-72.99	0.95	-54.98
	02050000	1.37	1.43	4.38	2.56	86.86
	02052000	0.42	1.15	173.81	0.34	-19.05
Time to Peak Stage Depth (hrs)	**02045500**	103	98	-4.85	96	-6.80
	02047000	121.5	135	11.11	123.25	1.44
	02047500	90.75	103.75	14.33	66.75	-26.45
	02049500	99.75	122.75	23.06	78.5	-21.30
	02050000	126	145.5	15.48	87	-30.95
	02052000	76.5	95	24.18	98	28.10

Figure 11 and Table 7 show the results of using the gridded rainfall data provided by TRMM when executing the model using the GPU solver with a grid cell size of 30 m and Manning's coefficient value of 0.6n. Using the gridded rainfall data with this coarse resolution produces results very similar to those found when using the rainfall gauge data and the polygon method. The model results almost match the observation peaks at the 02045500, 02047000, 02047500, and 02052000 USGS stations. The other two USGS stations, 02049500 and 02050000, where the modeled peaks are further from the observed peaks, are located on the same stream at the eastern part of the study area along with Station 02047500. This area has the mildest slopes in the study area (almost flat) (see Figure 2). The station furthest upstream is 02047500. At this station, the model predicts a slightly higher peak than the observed data and the modeled peak using the CPU model. The second station (02049500) has a much lower peak than the observed data; however, the modeled peak using the CPU solver is even lower than the modeled peak using the GPU solver. The peak at station 02050000 is much higher than the observed peak and the modeled peak using the CPU solver. The variation between the observed and modeled peaks at these three stations could be due to the coarse DEM resolution (10m×10 m) used in the model. The slightly higher peak at 02047500 may be due to slopes derived from the DEM being milder than the real slopes. The much lower peak and lower volume at 02049500 could be due to unrealistically steep slopes derived from the DEM compared to real slopes. Like with 02047500, the much higher peaks at 02050000 may be due to the DEM-derived slopes, which are milder than the real slopes. This would explain why the differences in the peaks at stations 02049500 and 02050000 are nearly the same but the one is below and the other is above the observed peak. If the slopes of the contributing areas to station 02049500 were milder, the peak there would be higher and the peak at the downstream station (02050000) would be lower, making both closer to the

observed data. This might improve if a higher DEM resolution is used within the model. Future work will explore this and the use of NEXRAD, which was unavailable for the study time period, to better understand the benefit of this rainfall data for predicting the stage depth peaks.

4.3. Post-processing and Automating Model Output Dissemination

Figure 12 shows the resulting workflow for model output post-processing and dissemination of model results. This workflow uses different Python libraries such as GDAL/OGR, and Simple KML library (SIMPLEKML) to generate the visualization of the flooded bridge locations and an email library to automatically email warnings to decision makers. The workflow and its products could be used with ArcMap, Google Maps, Google Earth, Geosheets or a custom website such as the one we have configured and hosted on the AWS, EC2 t2.micro instance (Figure 12). There are three products for visualization that can be generated from this workflow: (i) an ESRI shapefile that includes just the flooded bridges, (ii) a KMZ file that includes flood information for all bridges that can be visualized through Google Maps or Google Earth, and (iii) a dynamic and real-time visualization on Geosheets created by automatically uploading the bridges with their flooded status to a Google Sheet using the Google Drive API. Figure 13 shows an example of an advanced visualization for the flooded bridges directly on the Geosheets permanent URL. This visualization shows the bridges as not overtopped (green), nearly overtopped (yellow), and overtopped (red) from forecast rain events. Unlike hosting a website to visualize the KMZ file on the EC2 t2.micro instance, using Geosheets requires no webserver. However, hosting our own website in the long run will provide much more flexibility and the potential for more capabilities.

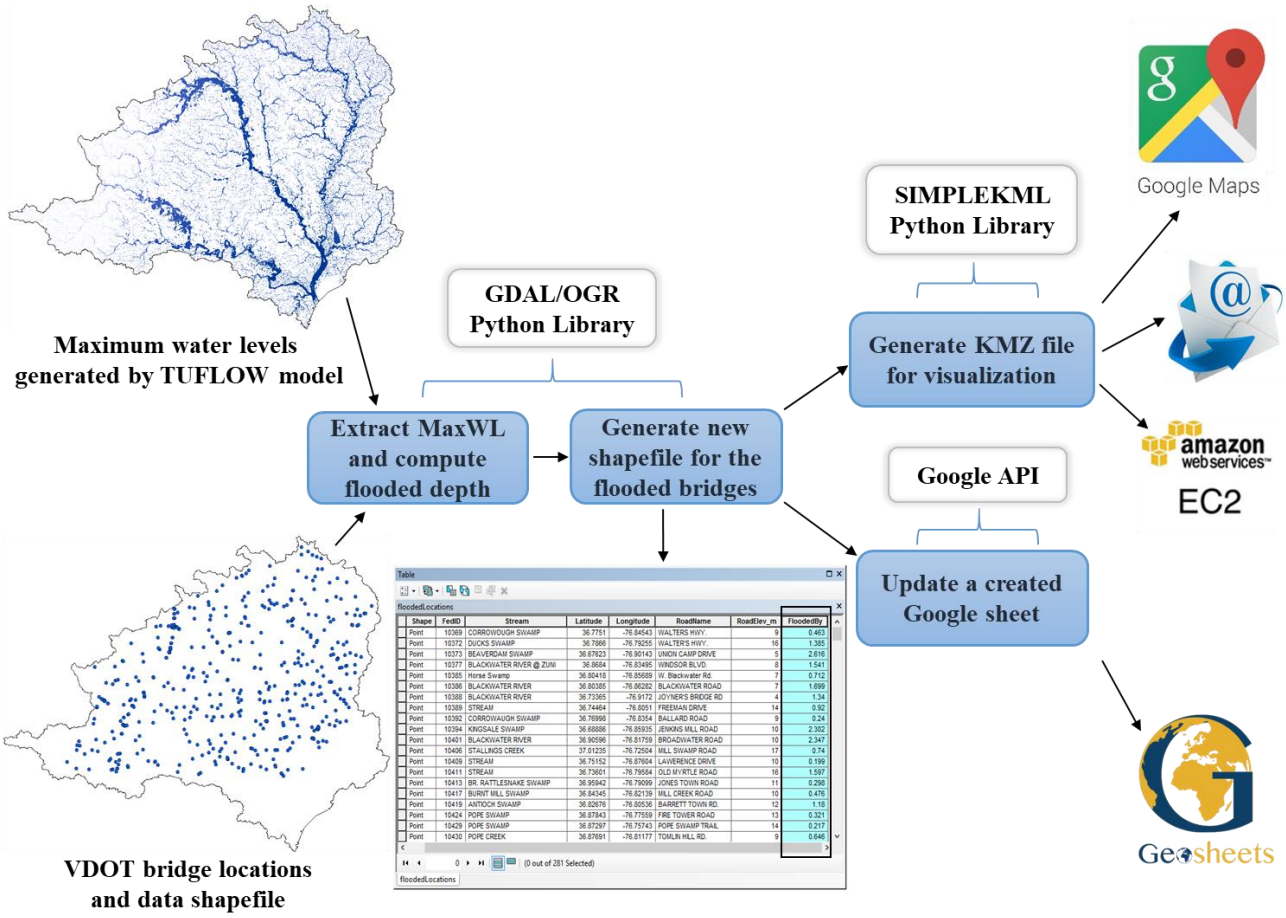


Figure 12 Post-processing workflow for producing different visualization resources.

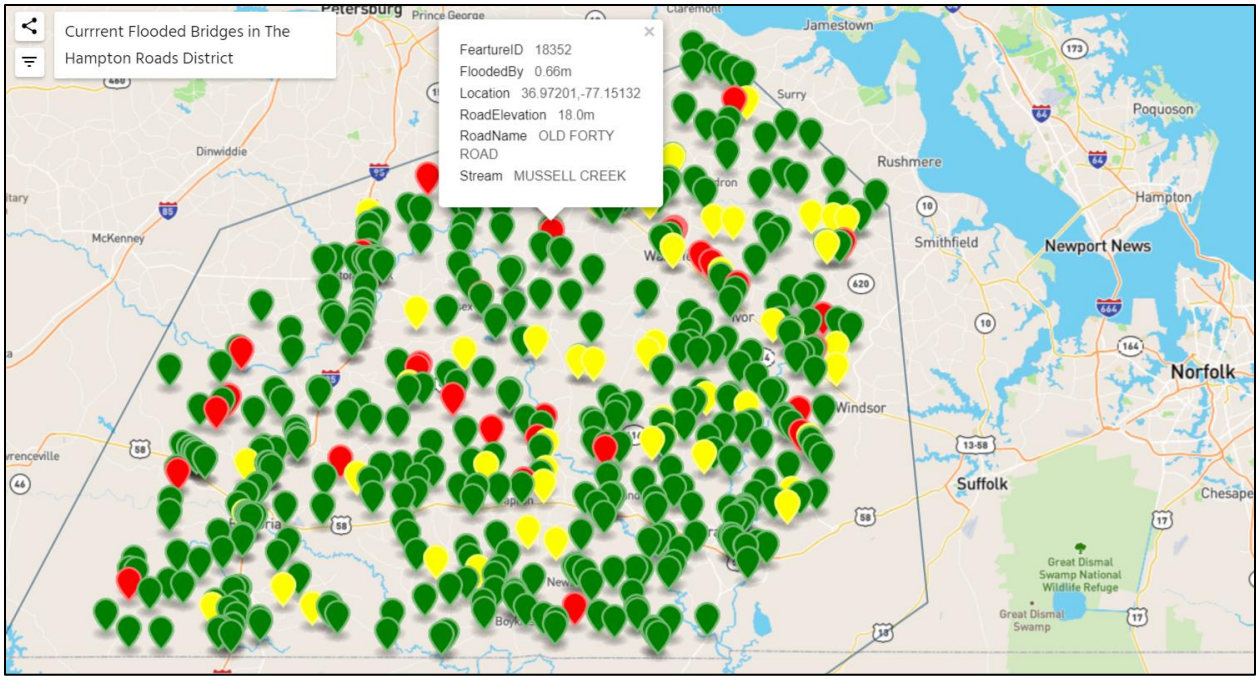


Figure 13 Real-time visualization with permanent URL for visualizing the flooded bridges location using Geosheets. (https://www.geosheets.com/map/s:Lo6Wq0Jl/Currrent-Flooded-Bridges-in-The-Hampton-Roads-District).

4.4. Automated flood warning system through AWS

Figure 14 shows the design of the automated workflow that meets the design requirements outlined in the methods section. This solution uses three AWS resources: (i) a low cost EC2 t2.micro instance running a Linux operating system, (ii) an GPU Instance (i.e. EC2 G2 or P2 instance) with Windows operating system, and (iii) a S3 Bucket. The EC2 t2.micro instance has two roles in the workflow. First, the instance continuously monitors rainfall forecasts to identify an extreme weather event. When an extreme weather event is identified, the EC2 t2.micro instance starts the GPU Instance and a model run is initiated. Second, the EC2 t2.micro instance serves the webpages used to visualize and disseminate the model results computed by the larger GPU Instance. The GPU Instance includes all of the model components and retrieves, preprocesses and prepares the forecast rainfall data for the hydrologic models. This same instance also executes the 2D hydrologic model. After the model runs, the GPU Instance sends model outputs to the EC2

t2.micro instance for visualization and dissemination. The model outputs are also sent, along with the processed forecast rainfall data used as model inputs, to the S3 bucket for archiving and reproducibility purposes.

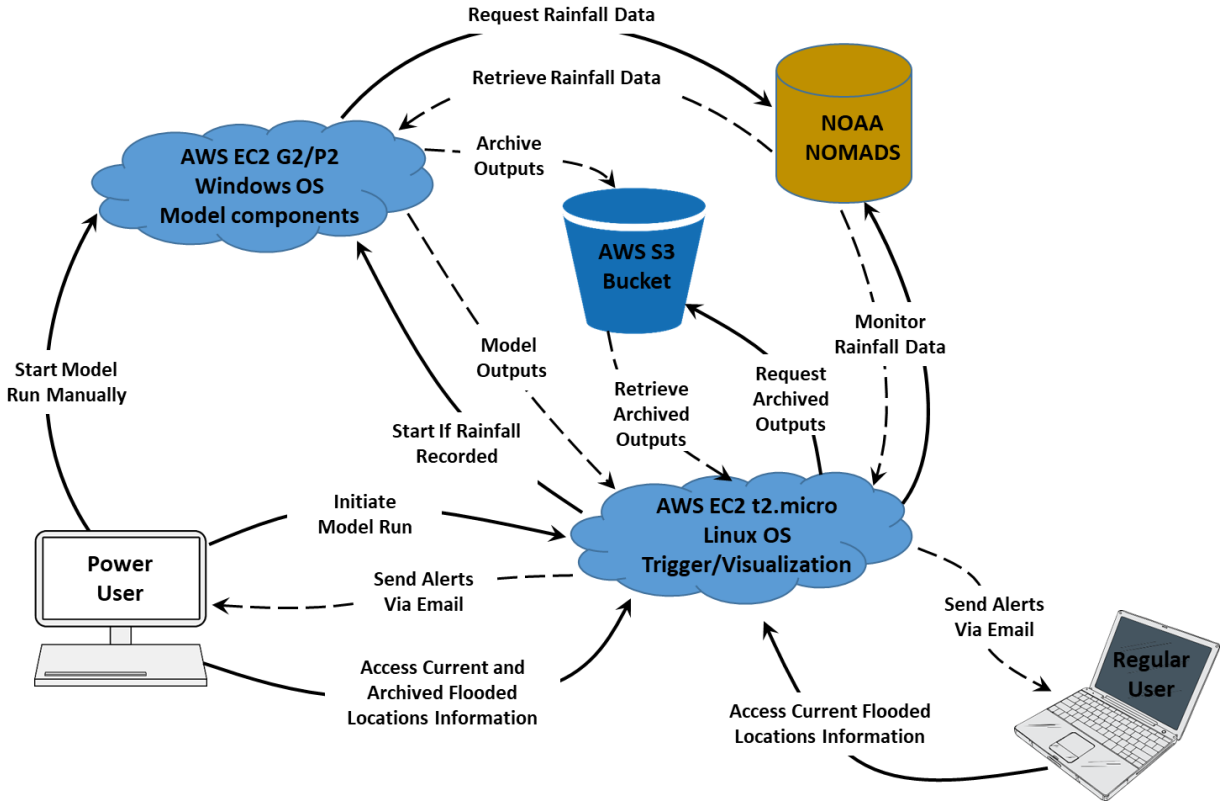


Figure 14 Design of the automated workflow for a flood warning system using AWS resources.

There are two classes of users that can access the model outputs via the webpages running on the EC2 t2.micro instance: regular users and power users. Regular users can access the current flooded locations and can register to receive alerts via email whenever locations are forecasted to flood. In the current implementation, regular users do not need to authenticate within the system. Power users have more privileges than the regular users, including access to all the archived inundation maps from the S3 bucket and the ability to run the model at any time via a powershell script or through the website hosted by the t2.micro instance.

AWS has the ability to securely control access to services and resources for specific users using the Identity and Access Management (IAM) service. This service was used to give permission to the EC2 t2.micro instance to start and stop the other GPU Instance. A new user was created and then given permission for starting and stopping the GPU Instance (Figure 15). By using the new user credentials, the GPU Instance ID, and command lines executed in a scripting language or at the AWS command line interface (CLI), the GPU Instance can be started and/or stopped automatically. The main script in the development web framework on the EC2 t2.micro instance is called server.py. Code was added to this Python script for monitoring and accessing the other GPU Instance. In this code, a process is run every hour to check the HRRR rainfall data, which is updated hourly. If the forecasted rainfall is over a certain threshold value, it will start the GPU Instance that includes the hydrologic model. The EC2 t2.micro instance keeps monitoring the GPU Instance to make sure that it is fully started (this is done by adding additional permissions to the user policy). Then, the EC2 t2.micro instance uses Secure Shell (SSH) to initiate a batch file that runs the main workflow for retrieving the data, executing the model, and generating the output. The 2D hydrologic model takes about 10 minutes to run through a forecasted period (18 hours) using a model grid resolution of 50 m on the M2 machine, while it takes about 38 minutes using the model grid resolution of 30 m on the M2 machine. The running time for the model with 30 m grid resolution is expected to be lower when using the EC2 p2.8xlarge instance. Using the p2.8xlarge AWS instance with five GPUs, it is expected that the runtime will be 6.3 minutes for a 50 m grid cell size and 24 minutes for 30 m grid cell size.

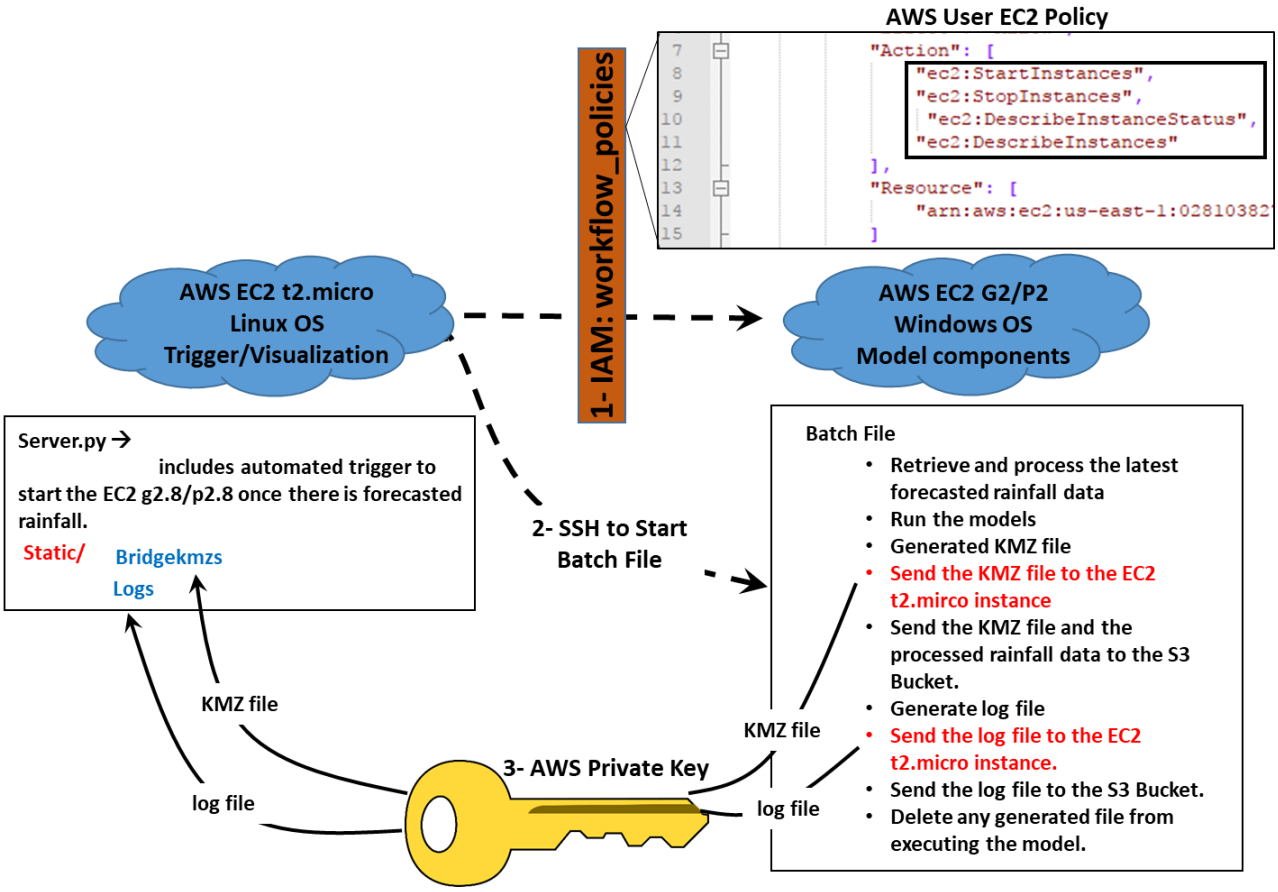


Figure 15 The policies between the EC2 t2.mico and G2 or P2 instances.

The batch file that automates the model execution operates as follows. First, the HRRR data is retrieved and processed. Following this, the hydrologic models are run and the maximum water level at each computational cell is computed and recorded for the duration of the simulation period. Once the maximum water level output file is available, the KMZ file is generated, which includes information about each bridge and culvert provided by VDOT, the maximum water level predicted by the model, and by how much each bridge would be overtopped. The KMZ file is sent to the t2.micro instance to be used for visualization using the AWS Private Key generated for the EC2 t2.micro instance. Another policy added to the IAM user is used to access the S3 Bucket and archive the processed rainfall data (Figure 16). A log file is generated that includes a record of the

parameters and scripts used in the whole process as a reference for users or decision makers. The log file is sent to both the EC2 t2.micro instance and the S3 Bucket for archiving. Finally, any files generated from running the whole workflow are deleted to minimize the storage on the GPU Instance.

A power user can use a powershell script to automatically initialize a model run. The script gives the user the option of running the workflow either locally or with the GPU Instance. When the workflow is chosen to run locally, the powershell script installs any required dependencies and then runs the batch file to start the workflow. If the user chooses to run the workflow through the cloud, the script asks for the IAM policy credentials and starts the GPU Instance. Once the instance is fully started, the script uses SSH to run the batch file to start the main workflow.

Figure 16 shows the different policies used by the EC2 t2.micro and G2 or P2 instances to access the S3 bucket folder that includes the archive information for each run. Also this figure shows the hierarchy of the S3 Bucket folders for archiving the workflow output data. The S3 Bucket folders receive data from the GPU Instance once it starts. To give full access for these specific folders and their contents to the GPU Instance, another policy was added to the IAM user (Figure 16). The GPU Instance uses the IAM user policy to access the main folder, floodwarningmodeldata, and archive the output data generated by the workflow in each specific subfolder. The EC2 t2.micro instance then retrieves the archived KMZ and log files to visualize them on the website. This is done by using a separate policy provided by the AWS S3 Bucket (Figure 16).

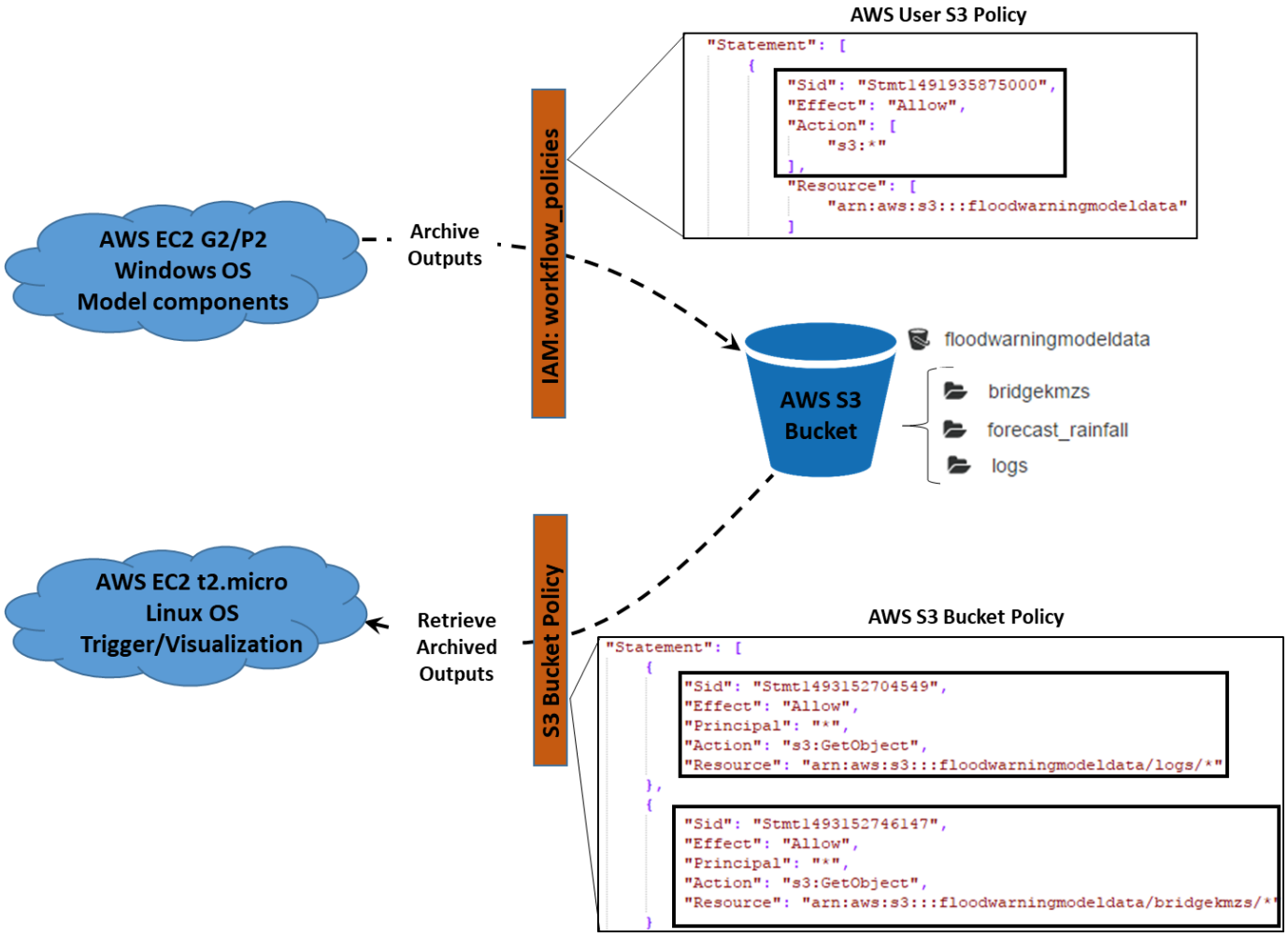


Figure 16 Different policies used to access the AWS S3 Bucket data, and the AWS S3 Bucket folder hierarchy.

The t2.micro instance handles the visualization of the output data using a Python based micro web framework, Flask (http://flask.pocoo.org/) (Figure 17). When a user accesses the website URL (https://vfis-aws.uvahydroinformatics.org) the most recent model output KMZ is displayed using the Google Maps JavaScript API. The output KMZ files, along with the corresponding log files from only the five most recent model runs, are available on the website to save storage space. NGINX (https://nginx.org/en/) and Gunicorn 'Green Unicorn' (http://gunicorn.org/) sit in between the flask application and the internet working in tandem to support many users on the website at the same time and handle the distribution of resources.

The t2.micro instance also triggers a model run when HRRR rainfall forecast data exceeds a given threshold. The forecast rainfall data is therefore retrieved every hour. If the rainfall exceeds a certain threshold value it will start the GPU Instance and initialize a model run with the latest rainfall data. An alert on the website will show users whether a model is being run, flooding is possible, or the model is up to date with no flooding predicted.

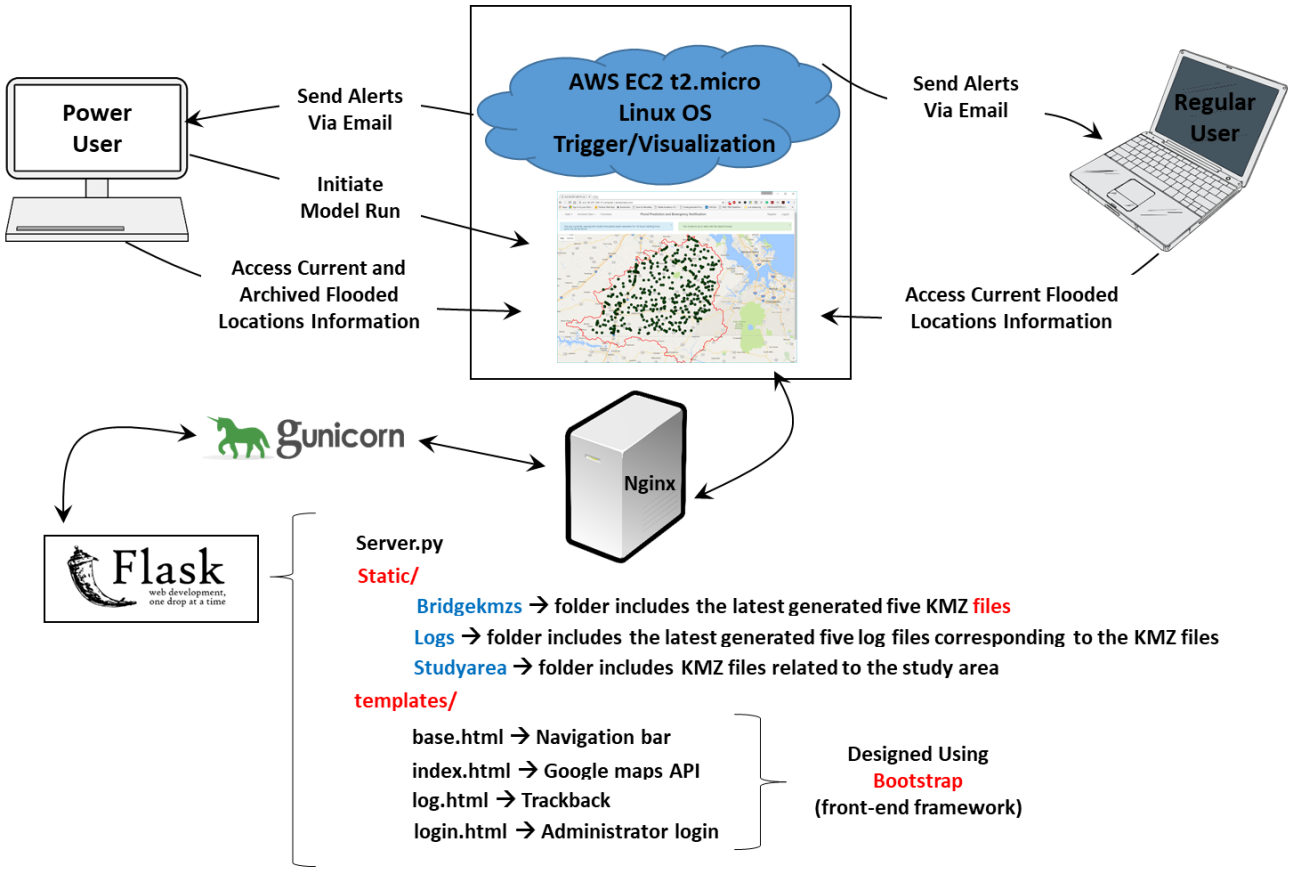


Figure 17 EC2 t2.micro instance and the web framework used to build up the website.

Figure 18 shows the architecture of the website. On the main view, the website contains a navbar allowing the selection of which data to view, a link to the log file, a login page, and a page to register for email alerts. The main section of the page is taken up by the Google Maps JavaScript API. Using the Google Maps JavaScript API allows us to easily display the map interface using all of Google's resources and overlay our output data on top of it. When a user clicks on a marker

signifying a bridge, they are presented with a box containing more information about that bridge and potential flooding events. Users can sign up and their email will be stored in a secured private Structured Query Language (SQL) database. The application will detect when flooding is possible and send an email to everyone on the list. Through the website, power users can display output data archived in the AWS S3 bucket without having to store output in the t2.micro instance, which has a limited amount of storage.

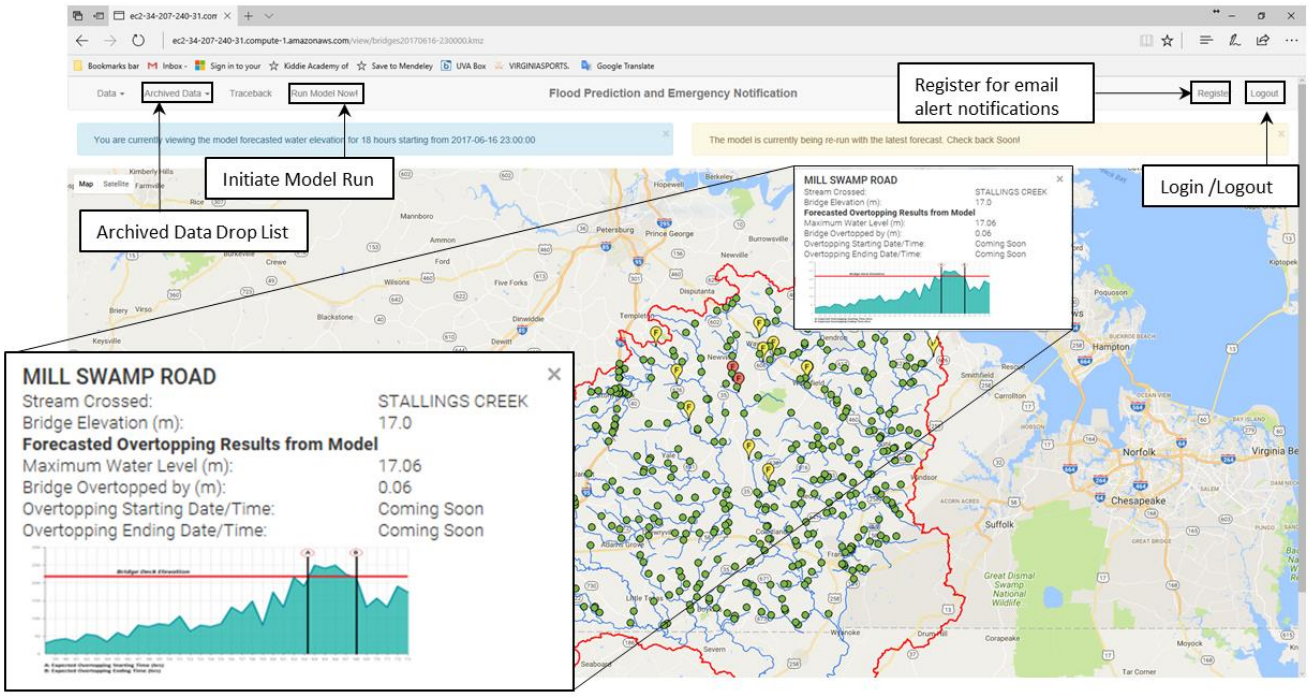


Figure 18 Main webpage of the flood warning decision support website.

5. Conclusions

This work described the creation of a cloud-based flood forecasting system designed to assist transportation decision makers in time-sensitive, emergency situations. The flood forecasting system was applied for the Virginia Department of Transportation in the Hampton

Roads region of Virginia, USA to provide decision makers with forecasts of flooded roadways and bridges in near real-time based on rainfall forecasts. By using GPU resources, the model was executed for a 15 day duration up to 80x faster (from 120 hours compared to 1.5 hours) compared to using a single CPU. An automated cloud-based workflow using AWS resources was designed and created to link and enhance the three core model components: (i) retrieval and formatting of high-resolution gridded HRRR rainfall forecast data, (ii) execution of the 2D model in a short duration to identify flood prone bridges and culverts, and (iii) real-time dissemination of model output via generation of an online map with flooded locations and the ability to automatically send alert messages via email.

Using the M2 machine described earlier, the 2D hydrodynamic model, which is the heart of the flood forecasting system, completes an analysis for the upcoming 18 forecast hours in approximately 10 minutes with a model grid cell size of 50 m and approximately 38 minutes with a model grid cell size of 30 m. Using the p2.8xlarge AWS instance with five GPUs, it is expected that the runtime will be 6.3 minutes for a 50 m grid cell size and 24 minutes for a 30 m grid cell size. For Hurricane Sandy, although the rainfall only lasted 4 days, the effects of the rainfall over the study area lasted 15 days. Assuming a 50 m grid cell size model takes 6.3 minutes to run for the upcoming 18 forecasted hours on the p2.8xlarge, if the model ran every hour through a 15 day period, running the workflow would cost about $350, assuming current AWS prices. For the same scenario, changing the grid cell size to 30 m, modeling 18 hours is expected to take about 24 minutes to run and cost $1260 for the 15 day duration. However, this assumes using five GPUs; further tests are required to identify the optimum number of GPUs to run the model with grid cell size of 30 m on the AWS EC2 P2.8 instance for this scenario.

Because the TUFLOW 2D model is expensive to run on a continuous basis, it is only used during extreme weather events. The t2.micro instance, which costs about $10 per month to run continuously, monitors the HRRR forecast rainfall data and compares it to rainfall thresholds that represent the amount of rain required to cause potential flooding. In the preliminary implementation, we used a fixed value for the threshold. In the future, we plan to find a way to compare the HRRR forecast data against the specific thresholds based on the antecedent moisture content of the soil before the start of any upcoming storm.

A main advantage of the cloud-based approach presented here is that it provides a way to strategically utilize computational resources only when flood events are likely to occur. Additionally, the workflow is automated, start to finish, without the need for any intermediate human interaction. This means that a decision maker with little or no experience regarding the details of hydrologic modeling, gridded rainfall data, pre- and post-processing procedures, and so forth, can easily execute the workflow and obtain and visualize model results. This work presents a preliminary calibration of the model, but additional work is needed to calibrate and evaluate the model across multiple historical flooding events. This calibration simply was not feasible before this work given the long model runtime. It is important to note that this model has only been tested for Hurricane Sandy. The local M2 machine, which was able to run the 15 day Hurricane Sandy model in 2.4 hours, could be used for the calibration process without excessive cloud costs. Results of this study suggest a higher resolution grid will improve model accuracy, but this too comes with an increased model runtime. A final challenge that needs more investigation is the differences between CPU and GPU generated results. This difference may become smaller with updates to the TUFLOW model software. A new version of TUFLOW was recently released, after the completion of this study, and includes a significantly enhanced version of the GPU model called TUFLOW

HPC (https://www.tuflow.com/). This version uses 2nd order solution accuracy solvers rather than the 1st order solvers that is used in the TUFLOW version used in this study. It also allows the user to add 2D bridges to the model for better representation within the system and has improvements in the multiple GPU speed performance for executing the model. This new version will be used in future work to further enhance, calibrate, and evaluate the model. Finally, more research is needed to see if improving model input data, such as using a finer DEM resolution for portions of the study area or NEXRAD rainfall data, will improve the GPU-based model results.

Acknowledgements

This work was supported by the Virginia Transportation Research Council (VTRC) under grant 107898. We also would like to acknowledge the Viz Lab, a facility for University of Virginia students, staff, and faculty to explore and investigate the power of visualization in research and education, for providing us with a local machine with GPUs to test our design and perform runs before moving the model to the cloud.

References

Anderson, J.D. and Wendt, J., 1995. Computational fluid dynamics (Vol. 206). New York: McGraw-Hill.

BMT WBM, 2016. TUFLOW User Manual. Build 2016-03-AA accessed June 6, 2017. http://www.tuflow.com/Download/TUFLOW/Releases/2016-03/AA/Doc/TUFLOW%20Manual.2016-03-AA.pdf.

Brodtkorb, A.R., Sætra, M.L., Altinakar, M., 2012. Efficient Shallow Water Simulations on GPUs: Implementation, Visualization, Verification, and Validation. Computers & Fluids, Vol. 55, pp. 1-12.

Boris, J.P., 1989. New directions in computational fluid dynamics. Annual review of fluid mechanics, 21(1), pp.345-385.

Castro, M.J., Ortega, S., de la Asunción, M., Mantas, J.M., and Gallardo, J.M., 2011. GPU Computing for Shallow Water Flow Simulation Based on Finite Volume Schemes. Comptes Rendus Mécanique, Vol. 339, No. 2-3, pp. 165-184.

Caviedes-Voullième, D., García-Navarro, P. and Murillo, J., 2012. Influence of mesh structure on 2D full shallow water equations and SCS Curve Number simulation of rainfall/runoff events. Journal of Hydrology, 448, pp.39-59.

CEIWR-HEC., 2009. HEC-DSSVue HEC Data Storage System Visual Utility Engine, User's Manual (Computer Program Documentation No. CPD-79). Davis, CA: US Army Corps of Engineers Institute for Water Resources.

David, C.H., Yang, Z.L., Hong, S., 2013. Regional-scale river flow modeling using off-the-shelf runoff products, thousands of mapped rivers and hundreds of stream flow gauges. Environmental modelling & software, 42, pp.116-132.

David, C.H., Maidment, D.R., Niu, G.Y., Yang, Z.L., Habets, F., Eijkhout, V., 2011. River network routing on the NHDPlus dataset. Journal of Hydrometeorology, 12(5), pp.913-934.

Ercan, M.B., Goodall, J.L., Castronova, A.M., Humphrey, M., Beekwilder, N., 2014. Calibration of SWAT models using the cloud. Environmental Modelling & Software, 62, pp.188-196.

Fowler, K.K., 2016. Flood-inundation maps for the Wabash River at New Harmony, Indiana: U.S. Geological Survey Scientific Investigations Report 2016-5119, 14 p.

Garcia, R., Restrepo, P., DeWeese, M., Ziemer, M., Palmer, J., Thornburg J., Lacasta, A., 2015. Advanced GPU Paralellization for Two-Dimensional Operational River Flood Forecasting. In 36th International Association for Hydro-Environment Engineering and Research World Congress. The Hague, Netherlands.

Gourley, J.J., Flamig, Z.L., Vergara, H., Kirstetter, P.E., Clark III, R.A., Argyle, E., Arthur, A., Martinaitis, S., Terti, G., Erlingis, J.M. and Hong, Y., 2017. The FLASH Project: Improving the tools for flash flood monitoring and prediction across the United States. Bulletin of the American Meteorological Society, 98(2), pp.361-372.

Granell, C., Havlik, D., Schade, S., Sabeur, Z., Delaney, C., Pielorz, J., Usländer, T., Mazzetti, P., Schleidt, K., Kobernus, M., Havlik, F., 2016. Future Internet technologies for environmental applications. Environmental Modelling & Software, 78, pp.1-15.

Hassan Water Resources PLC, 2012. Past performance accessed June 12, 2017. http://hwrgov.com/past-performance/.

Hu, Y., Cai, X., DuPont, B., 2015. Design of a web-based application of the coupled multi-agent system model and environmental model for watershed management analysis using Hadoop. Environmental Modelling & Software, 70, pp.149-162.

Huxley, C., Syme, B., 2016. TUFLOW GPU-Best Practice Advice for Hydrologic and Hydraulic Model Simulations. In Proceedings of the 37th Hydrology and Water Resources Symposium (HWRS), Queenstown, New Zealand.

Jacobsen, D., Thibault, J.C., Senocak, I., 2010. An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters. In American Institute of Aeronautics and Astronautics (AIAA) 48th Aerospace Science Meeting Proceedings. Orlando, Florida, USA.

Kalyanapu, A.J., Shankar, S., Pardyjak, E.R., Judi, D.R., Burian, S.J., 2011. Assessment of GPU Computational Enhancement to a 2D Flood Model. Environmental Modelling & Software, Vol. 26, No. 8, pp. 1009-1016.

Kurtz, W., Lapin, A., Schilling, O.S., Tang, Q., Schiller, E., Braun, T., Hunkeler, D., Vereecken, H., Sudicky, E., Kropf, P., Franssen, H.J.H., 2017. Integrating hydrological modelling, data assimilation and cloud computing for real-time management of water resources. Environmental modelling & software, 93, pp.418-435.

Lacasta, A., García-Navarro, P., Burguete, J., Murillo, J., 2013. Preprocess Static Subdomain Decomposition in Practical Cases of 2D Unsteady Hydraulic Simulation. Computers & Fluids, Vol. 80, pp. 225-232.

LeVeque, R.J., 2002. Finite volume methods for hyperbolic problems (Vol. 31). Cambridge university press.

Liu, F. and Hodges, B.R., 2014. Applying microprocessor analysis methods to river network modelling. Environmental Modelling & Software, 52, pp.234-252.

Maidment, D.R., 2017. Conceptual Framework for the National Flood Interoperability Experiment. JAWRA Journal of the American Water Resources Association.

Melillo, J.M., Richmond, T.T., Yohe, G.W., 2014. Climate change impacts in the United States. Third National Climate Assessment.

National Oceanic and Atmospheric Administration, 2012. Rapid Refresh accessed January 24, 2017. https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/rapid-refresh-rap.

National Oceanic and Atmospheric Administration, 2017a. Advanced Data Access Methods accessed January 24, 2017. http://nomads.ncdc.noaa.gov/guide/?name=advanced.

National Oceanic and Atmospheric Administration, 2017b. Data Products accessed January 24, 2017. https://www.ncdc.noaa.gov/nomads/data-products.

National Research Council, 2009. Mapping the zone: improving flood map accuracy. National Academies Press.

NFIP Statistics, 2016. NFIP Statistics. The official site of the NFIP accessed December 8, 2016. https://www.floodsmart.gov/floodsmart/pages/media_resources/stats.jsp.

Pau, J.C. and Sanders, B.F., 2006. Performance of parallel implementations of an explicit finite-volume shallow-water model. Journal of computing in civil engineering, 20(2), pp.99-110.

Perry,C. A., 2000. Significant Floods in the United States During the 20th Century - USGS Measures a Century of Floods. Kansas Water Science Center accessed December 8, 2016. https://ks.water.usgs.gov/pubs/fact-sheets/fs.024-00.html.

Rostrup, S., Sterck, H.D., 2010. Parallel Hyperbolic PDE Simulation on Clusters: Cell Versus GPU. Computer Physics Communications, Vol. 181, No. 12, p. 2164.

Sanders, B.F., Schubert, J.E., Detwiler, R.L., 2010. ParBreZo: A Parallel, Unstructured Grid, Godunov-Type, Shallow-Water Code for High-Resolution Flood Inundation Modeling at the Regional Scale. Advances in Water Resources,Vol. 33, Issue 12, December, pp. 1456-1467.

SSURGO (Soil Survey Geographic). (2012). "Soil survey staff, natural resources conservation service." http://sdmdataaccess.nrcs.usda.gov/

Steissberg, T. E., McPherson, M. M., 2011. HEC-GridUtil Grid Utility Program Managing Gridded Data with HEC-DSS, User's Manual Version 2.0 (Computer Program Documentation No. CPD-89). Davis, CA: US Army Corps of Engineers Institute for Water Resources Hydrologic Engineering Center.

Sun, A., 2013. Enabling collaborative decision-making in watershed management using cloud-computing services. Environmental Modelling & Software, 41, pp.93-97.

Syme, W.J., 2001. TUFLOW-Two & One dimensional unsteady flow Software for rivers, estuaries and coastal waters. In IEAust Water Panel Seminar and Workshop on 2d Flood Modelling, Sydney.

Tóth, G., Keppens, R. and Botchev, M.A., 1998. Implicit and semi-implicit schemes in the Versatile Advection Code: numerical tests. Astronomy and Astrophysics, 332, pp.1159-1170.

Vacondio, R., Dal Palù, A., Mignosa, P., 2014. GPU-enhanced Finite Volume Shallow Water Solver for Fast Flood Simulations. Environmental Modelling & Software, Vol. 57, pp. 60-75.

Wan, Z., Hong, Y., Khan, S., Gourley, J., Flamig, Z., Kirschbaum, D., Tang, G., 2014. A cloud-based global flood disaster community cyber-infrastructure: Development and demonstration. Environmental Modelling & Software, 58, pp.86-94.

Werner, M., Schellekens, J., Gijsbers, P., van Dijk, M., van den Akker, O., Heynert, K., 2013. The Delft-FEWS flow forecasting system. Environmental Modelling & Software, 40, pp.65-77.

Zhao, Y., 2008. Lattice Boltzmann based PDE solver on the GPU. The visual computer, 24(5), pp.323-333.