WDCloud: An End to End System for Large-Scale Watershed Delineation on Cloud

In Kee Kim*, Jacob Steele*, Anthony M. Castronova†, Jonathan L. Goodall*, and Marty Humphrey*
*University of Virginia, Charlottesville, VA, 22903, {ik2sb, jss2zb, goodall}@virginia.edu, humphrey@cs.virginia.edu
†Utah State University, Logan, UT, 84322, tony.castronova@usu.edu

***Abstract*—Watershed delineation is a process to compute the drainage area for a point on the land surface, which is a critical step in hydrologic and water resources analysis. However, existing watershed delineation tools are still insufficient to support hydrologists and watershed researchers due to the lack of essential capabilities such as fully leveraging scalable and high performance computing infrastructure (public cloud), and providing predictable performance for the delineation tasks. To solve these problems, this paper reports on *WDCloud*, which is a system for large-scale watershed delineation on public cloud. For the design and implementation of *WDCloud*, we employ three main approaches: 1) an automated catchment search mechanism for a public data set, 2) three performance improvement strategies (Data-reuse, parallel-union, and MapReduce), and 3) local linear regression-based execution time estimator for watershed delineation. Moreover, *WDCloud* extensively utilizes several compute and storage capabilities from Amazon Web Services in order to maximize the performance, scalability, and elasticity of watershed delineation system. Our evaluations on *WDCloud* focus on two main aspects of *WDCloud*; the performance improvement for watershed delineation via three strategies and the estimation accuracy for watershed delineation time by local linear regression. The evaluation results show that *WDCloud* can achieve 18x–111x of speed-ups for delineating any scale of watersheds in the contiguous United States as compared to commodity laptop environments, and accurately predict execution time for watershed delineation with 85.6% of prediction accuracy, which is 23%–43% higher than other state-of-the-art approaches.**

I. Introduction

Analysis of regional-scale watershed systems is critical to understand the impact of floods, droughts, and water pollution. Watershed modelers use hydrographic data in simulation models to better understand potential impacts of these events and testing mitigation strategies [19, 22]. The starting point of many hydrologic analyses is defining a watershed boundary for the area of interest, which is called watershed delineation [10]. Watershed delineation plays an important role in hydrologic analysis because it defines the scope of the modeling domain, thereby impacting all further analysis and modeling steps [11]. There are national-scale data available for performing watershed delineation, but few convenient tools are able to leverage these data for simple and quick watershed delineation for any point in the contiguous United States.

With advancements in computing technology, scientific research has become increasingly reliant on computational tools to quickly analyze large amounts of data and provide useful information to researchers. Unfortunately, the design of scientific applications does not necessarily utilize these advances. For example, on commodity desktop hardware, watershed delineation can take several hours for large watersheds. Current approaches also rely heavily on GIS desktop software, which can have a steep learning curve for those unfamiliar with the software and tedious data preparation steps to arrive at the desired watershed boundary dataset [17, 24]. This high cost and low reward situation is an unnecessary burden on hydrologists and watershed researchers constrained to modeling smaller watersheds that can be easily accommodated by available software options.

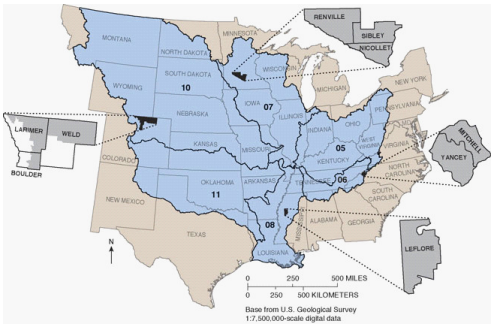


Fig. 1. Mississippi Watershed. This is the largest watershed in the United States, and is composed of 1,100,000+ catchments, which are distributed over 10 distinct regions in NHD+. (Courtesy of United State Geological Survey [5]).

Many approaches have been proposed for addressing the challenge of watershed delineation at a national-scale [3, 7, 10]. Castronova and Goodall [10] proposed an approach that leveraged pre-computed data from the National Hydrography Dataset Plus (NHD+) program [5, 6]. An advantage of this approach is that it did not require additional data pre-computation steps, which are common for many large watershed delineation algorithms. However, the approach did not scale well to large watersheds (e.g., the Mississippi watershed in Figure 1) and resulted in long execution time for delineating such a large watershed. Long execution time of large watershed delineation in the approach is related to the size of underlying data. For example, the Mississippi watershed consists of 1,100,000+ NHD+ catchments, which is more than 50% of all catchments in United States (U.S. has approximately 2 million NHD+ catchments). The algorithm requires merging of these individual catchments into a single watershed polygon, and execution time of such a large-scale watershed delineation is simply dominated by time to perform this geometric union operation of catchments.

978-1-4799-9926-2/15/$31.00 ©2015 IEEE

Estimated computation time to delineate the entire Mississippi watershed is approximately 10+ hours on commodity laptop hardware using the Castronova and Goodall algorithm. This does not lend itself to an interactive system where the majority of watersheds can be delineated and returned while the user waits. While achieving this goal for the most extreme cases such as the Mississippi watershed is very challenging without significant data pre-processing, delineation time of 10 minutes or less is desirable for an online watershed delineation tool. Therefore, the scientists need a new software architecture on HPC infrastructure for the watershed delineation process, which can dramatically reduce the execution time of watershed delineation. In terms of building a HPC (High Performance Computing) infrastructure for watershed delineation, a local HPC cluster is often technically and financially infeasible for hydrologists. Thus, leveraging the public clouds (e.g. Amazon Web Services [1] and Microsoft Azure [2]) as the HPC infrastructure is more desirable due to the elasticity, scalability and cost efficiency of public cloud [15, 21].

Moreover, another challenge of watershed delineation is highly variable execution time of delineation tasks based on input coordinates. When a scientist requests a particular coordinate for delineation, the scientist may not know how long the delineation task will take. This is often problematic when the scientist expects an instantaneous response to a large watershed request. Therefore, to improve the scientists experience, a watershed delineation system should be able to estimate and provide the execution time of watershed delineation with high accuracy.

To solve the problems, we introduce ***WDCloud***, an end-to-end system for large-scale watershed delineation on cloud. ***WDCloud*** employs following approaches; 1) an automated catchments search mechanism using NHD+ (National Hydrograph Dataset Plus)[1], 2) various performance improvement strategies, and 4) a local linear regression (LLR) based execution time estimation for watershed delineation. The automated catchments search mechanism is designed to allow scientists to delineate large-scale and multi-region watersheds. We also leverage three strategies to reduce the duration of watershed delineation. We employ a data-reuse strategy, MapReduce [14], and parallel-union depending on the scale of watersheds. ***WDCloud*** employs the data-reuse to delineate extremely large-scale and multi-region watersheds (e.g. the Mississippi watershed). ***WDCloud*** also uses MapReduce for large-scale watersheds, and leverages the parallel-union for medium- and small-scale watersheds. ***WDCloud*** automatically chooses a proper strategy based on the size of the requested watershed. LLR [20] is used to accurately estimate the execution time of watershed delineation requests from the scientists. We implement ***WDCloud*** on Amazon Web Services (AWS) with extensive use of various capabilities from AWS such as diverse types of virtual machines (VM), auto-scaling, and cost efficient S3 storage services in order to improve the performance of the system.

[1]NHD+ [5, 6] is the most recent public hydrograph dataset provided by United State Geological Survey (USGS). NHD+ contains essential information for watershed and water resource research such as stream flows and directions.

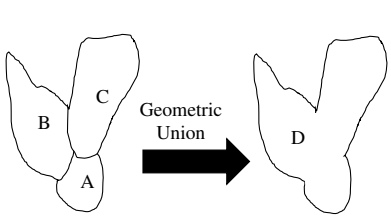


Fig. 2. Example of geometric union for catchments.

Our evaluations focus on two main aspects of ***WDCloud***; the speed up of computation time for watershed delineation tasks via three performance improvement strategies, and the accuracy of delineation time estimation via LLR. In terms of the speed up of watershed delineation tasks, ***WDCloud*** achieves 111x speed up for the Mississippi watershed (the largest watershed in U.S.) through the data-reuse strategy, up to 21x speed up for large-scale watershed via MapReduce, and 18x speed up for medium- and small-scale watershed by using the parallel-union approach. Moreover, the LLR-based delineation time predictor of ***WDCloud*** provides the reliable estimation of delineation time with 85.6% of prediction accuracy. This result is 23%–43% higher than other state-of-the-art estimation approaches such as kNN [20] and mean-based estimation [25].

The contributions of this paper are:

- We introduce ***WDCloud*** that allows the hydrologists to delineate any scale of watershed in U.S. with faster execution time.
- We introduce three performance improvement strategies (data-reuse, MapReduce, and parallel-union), which enable ***WDCloud*** to achieve 18x–111x speed up of watershed delineation as compared to commodity laptop hardware.
- We use LLR-based execution time estimator that provides predictable performance of watershed delineation requests.

The rest of this paper is organized as follows: Section II gives background of this work. Section III highlights the design of ***WDCloud*** and main approaches. Section IV is the evaluation and discussion. Section V contains related work and Section VI concludes this paper.

II. BACKGROUND

A. Watershed Delineation

The building blocks of watershed models are geographic areas, called catchments. Each catchment is defined by its boundary coordinates, and is analogous to nodes in a tree-structured model that defines the hydrologic connectivity of catchments along a river network. A watershed is a collection of catchments that collectively define the area that drains to some point on the land surface (the outlet of the watershed). The computation that joins a set of catchments together, and creates the set of boundary points for the watershed is called the geometric union of the catchments (shown in Figure 2). The result of watershed delineation is used in conjunction with land use data and other types of geospatial data to create the complete boundary of watershed for a given region.

The watershed delineation approach proposed by Castronova and Goodall [10] is composed of several pipeline steps that

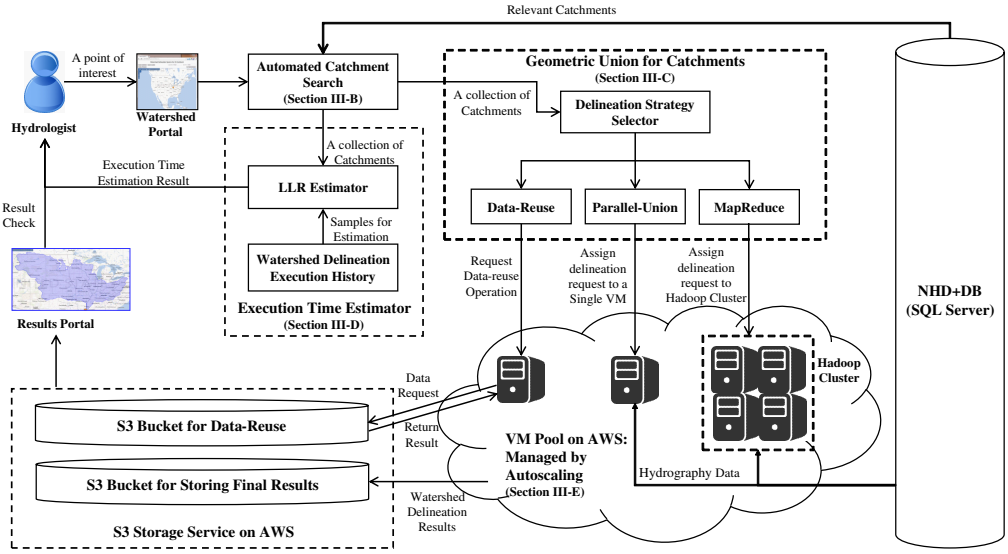


Fig. 3. Architecture of ***WDCloud*** on AWS.

manipulate and refine the NHD+ [6]. The highest computation overhead is to process the geometric union operations of catchments, which contribute to forming the target watershed. The geometric union operation must examine the catchments and merge intermediate boundaries of the catchments to build the final boundary of the target watershed. This process is time consuming due to the multiple passes required to examine and determine each catchment's effect on the boundary of the entire set of catchments.

Watershed delineation can be performed by hydrologists or other interested parties through a program such as GIS tools [12, 26] on the scientists' laptop or online watershed delineation services. However, a drawback from using exiting GIS tools is that these tools require several steps for the hydrologists to manually process underlying hydrography data to delineate the target watershed. Online water modeling services (e.g. ESRI's watershed delineation service.[2]), which are similar with ***WDCloud***, also have several disadvantages. The disadvantages are black box nature of commercial software service and restricted use of open source watershed dataset, and high licensing cost. Another option offered by USGS is StreamStats [7], which provides online watershed delineation capabilities based on NHD. However, StreamStats requires significant amounts of pre-computing steps, which can delay the incorporation of enhancements in underlying NHD data. Moreover, StreamStats is not currently available for all of the lower 48 states.

Instead, ***WDCloud*** uses nationally consistent and publicly available underlying data (NHD+) with very minimal pre-computing of these data required in the tool. Therefore, enhancements to this dataset can be quickly incorporated into ***WDCloud*** whereas other systems would require re-running data pre-computing steps. This is true for both the ESRI's ArcGIS tool [3] and USGS Stream Stats to the best of our knowledge. Furthermore, the typical manual delineation steps of Digital Elevation Model (DEM) processing that most hydrologists would perform requires knowledge of GIS software, many data processing steps to arrive at the final watershed boundary of interest, and do not scale well to large (or even regional) scale watersheds. However, ***WDCloud***, only requires the users to select a location on a map, effectively removing data management from the user.

B. National Hydrograph Dataset

The National Hydrography Dataset (NHD) [5] is an openly available vector and raster dataset. Specifically, we integrated NHD+ [6], which a version of NHD that includes catchments for each NHD reach feature, into ***WDCloud*** because it possessed several desirable qualities. Namely, the dataset is quality controlled and assured by the USGS and includes catchment delineations that enforce flow along NHD reaches and no flow across boundaries defined in the coarser scale Watershed Boundary Dataset (WBD)[3]. This dataset therefore provides a nationally consistent representation of the hydrologic connectivity of the landscape.

NHD+ encompasses the entire contiguous United States, is segmented into 21 HUC regions, and contains approximately 2 million unique catchments. HUC codes are used to segment watersheds into groupings of 2, 4, 6, and 8 digit HUCs. 2 digit HUCs are the largest watersheds and contain several 4 digit HUCs. 4 digit HUCs contain 6 digit HUCs, and 6 digit HUCs contain 8 digit HUCs. This provides a segmentation to NHD+ catchments that makes it possible to know, based on its HUC, what 2 digit, 4 digit, 6 digit, and 8 digit HUC watershed that catchment is within common geospatial dataset.

III. ***WDCloud*** Design

A. Design of ***WDCloud*** on AWS

WDCloud is composed of six components as shown in Figure 3. These six components are: 1) a web portal for watershed delineation, 2) NHD+ database, 3) automated catchment

[2]https://www.arcgis.com

[3]http://nhd.usgs.gov/wbd.html

Algorithm 1 Automated Catchment Search for Multiple Regions in NHD+

Require: $coord$: coordinate for outlet of the target watershed
1: $start\_HUC\_region \leftarrow$ get_regional_dataset ($coord$)
2: $terminal\_paths \leftarrow$ get_terminal_path_infos ($start\_HUC\_region$, $coord$)
3: $catchments \leftarrow$ get_catchments ($start\_HUC\_region$, $terminal\_paths$)
4:
5: $multi\_region\_hydroseqs \leftarrow$ get_multi_region_hydroseqs_info ($start\_HUC\_region$, $terminal\_paths$)
6: **if** length($multi\_region\_hydroseqs$) > 0 **then**
7: $related\_HUC\_regions \leftarrow$ find_related_HUC_regions ($multi\_region\_hydroseqs$)
8: $region\_index \leftarrow 0$
9: **while** $region\_index <$ legnth($related\_HUC\_regions$) **do**
10: $catchment\_for\_HUC\_region \leftarrow$ get_catchments ($related\_HUC\_regions$[$region\_index$], $terminal\_paths$)
11: $catchments$.append($catchment\_for\_HUC\_region$)
12: $region\_index$++
13: **end while**
14: **end if**

search module, 4) geometric union module, 5) execution time estimator, and 6) compute and storage resources on AWS.

Web Portal for Watershed Delineation: This portal provides a user interface to select an outlet coordinate for delineating a target watershed. Once a hydrologist selects a point of interest on the user interface on the portal and clicks on a submit button, a process for watershed delineation starts. This portal is also used to confirm the final delineation result for the input coordinate. The final result will be displayed on this portal and provided as several files, which are existing GIS tool-compliant format such as Keyhole Markup Language [4].

NHD+ Database: This component contains NHD+ dataset required for watershed delineation. Originally, NHD+ consists of 21 distinct HUC region dataset (for contiguous U.S.). Each HUC region dataset includes several raw-level hydrography data such as DEM (Digital Elevation Mode) and flow direction/accumulations. Each dataset only covers limited areas in U.S. based on HUC code. In order to facilitate the delineation for the large-scale watershed (e.g. Mississippi), we extract necessary data from NHD+ and store these data to Microsoft SQL Server.

Automated Catchment Search Module: This module is used to automatically collect relevant catchments for the target watershed, which is distributed on multiple HUC regions in NHD+. The details of this automated mechanism will be described in Section III-B.

Geometric Union Module: This module performs the geometric union operation to calculate the final result of the target watershed. We proposed three strategies to improve the performance of geometric union operation. The three strategies will be described in Section III-C.

Execution Time Estimator: This component is used to provide accurate estimation for delineation time of the target watershed. We employ LLR (Local Linear Regression) for this estimation. This estimator will be explained in Section III-D.

Compute and Storage Resources on AWS: ***WDCloud*** uses Amazon Web Service (AWS) [1] cloud infrastructure for its computing and data management environments. ***WDCloud*** utilizes various configurations (e.g. a single VM or VM cluster) and types of VMs based on the union strategy from the geometric union module. These VMs performs actual delineation process for the watershed. (most processes of watershed delineation performed on VMs are related to geometric union of catchments.) The VM resources on ***WDCloud*** are managed by autoscaling mechanism, which will be described in Section III-E. Moreover, ***WDCloud*** leverages Amazon S3 (Simple Storage Service) [1] in order to store pre-compute data for large-scale watershed and delineation results.

TABLE I
THREE ATTRIBUTES FOR AUTOMATED CATCHMENTS SEARCH MECHANISM.

Attributes	Description
HydroSeq	Unique hydrologic sequence number assigned to each region in the dataset.
TerminalPath	Hydrologic sequence number of the terminal feature of the watershed network.
DnHydroSeq	Hydrologic sequence number of downstream.

B. Automated Catchment Search Mechanism using NHD+

To automatically search and collect relevant catchments for the target watershed, we propose an Automated Catchments Search Mechanism (ACSM) using NHD+. For the ACSM, we leverage three main attributes provided by NHD+. These three attributes are $TerminalPath$, $HydropSeq$, and $DnHydroSeq$ [10], which are described in Table I.

Algorithm 1 describes the details of the ACSM. The ACSM starts with finding a proper HUC region dataset ($start\_HUC\_region$) in NHD+ for an outlet coordinate (input from a user) of the target watershed (**ln 1**). Based on the coordinate and the HUC region dataset for the input outlet, this automated mechanism finds $TerminalPath$ for the watershed (**ln 2**). Using $TerminalPath$, the ACSM finds catchments for the target watershed in that HUC region. The ACSM, then, finds $HydropSeqs$ ($multi\_region\_hydroseqs$ at **ln 5**) in the HUC region, which encompass the target watershed's hydrological flow information with other HUC regions in NHD+. If $multi\_region\_hydroseqs$ exist, this means that the target watershed is also distributed over other HUC regions in NHD+ (**ln 6**). Otherwise, the target watershed is composed of catchments in a single HUC region ($start\_HUC\_region$).

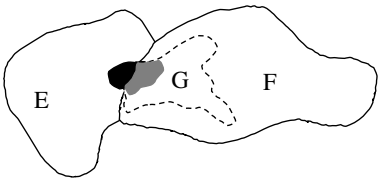


Fig. 4. Data-reuse Example – E and F are pre-defined regions. The black catchment is aggregated. Since the gray catchment flows into the black catchment, the gray catchment and its watershed, labeled G, must also be collected.

By leveraging $multi\_region\_hydroseqs$, the ACSM finds all relevant other HUC region dataset by comparing $DnHydroSeq$ in other HUC regions with $multi\_region\_hydroseqs$ (**ln 7**). The ACSM searches for all relevant HUC regions and finds catchments in those regions by using $TerminalPath$ (**ln 10**). Once the ACSM completes to explore all relevant HUC regions, all catchments to form the target watershed are collected.

C. Performance Improvement Strategies for Geometric Union

Once the ACSM collects all relevant catchments for the target watershed, the watershed delineation performs geometric union operation (Figure 2 in Section II-A) using all the catchments to build a single catchment representing the boundary of the target watershed. This geometric union is the most time consuming operation in the watershed delineation. To reduce the execution time for the geometric union operation, we employ three strategies: 1) data-reuse, 2) parallel-union, and 3) MapReduce.

Data-Reuse: The general architecture for data-reuse is to pre-compute catchment unions. When a stored point is accessed, instead of a full traversal and merge of all the relevant catchments, traversal halts and a single catchment is read. Pre-computation and storage eliminates time spent traversing the catchment network and limits the count of catchments passed to the union operation. Although, this strategy has similarities with caching, it is much different. Data-reuse provides a guaranteed performance enhancement unlike caching. Data-reuse does not require a specific point of interest to be already selected by a user in order for that point to achieve a performance increase. Data-reuse is an offline optimization that targets the large-scale and multi-region watersheds such as the Mississippi watershed.

By pre-computing every point, the union computation would require a single file read operation, greatly improving runtime performance. This level of pre-computation is infeasible because of two reasons. First, the time required to pre-compute every point would cause a delay in the adoption of new data sources. In other words, if new data was desired by the hydrologists, they would be forced to wait weeks if not months to actually work with their data. The other is the cost of storing, possibly 1-2 Gigabytes, each of the 2 million catchments in NHD+, which quickly removes some of the cost benefit our system exhibits over other systems.

By utilizing watershed domain knowledge, we create a data-reuse mechanism that efficiently stores pre-computed watershed data by targeting all points requiring multi-region results. This provides great improvement at very limited cost in terms of both time and money. In NHD+, the hydrology data for the contiguous U.S. is divided into 21 distinct regions. Each region can have a few catchments that connect to another region. Data-reuse stores results that span the regional boundaries, leveraging the natural segmentation of the data. By storing these multi-region watersheds, the performance is benefited by eliminating multiple queries to aggregate the multi-region catchments and the fact that this allows most of the work to be computed offline. Figure 4 shows an example for the multi-region data-reuse strategy. E and F are regions defined as part of the NHD+ dataset. The gray catchment in F flows into the black catchment in E. This means that when the black catchment is collected, the gray catchment must also be collected and the watershed of the gray catchment is shown as the area encompassed by the dotted line, labeled G. Data-reuse pre-computes the watershed G and stores it, so it can be retrieved when the black catchment is collected. The data-reuse limits traversal of the catchments and the number of catchments sent to the union by targeting large watersheds. This strategy has a low memory cost since it consists of storing only 16 files (total file size is 106 MB).

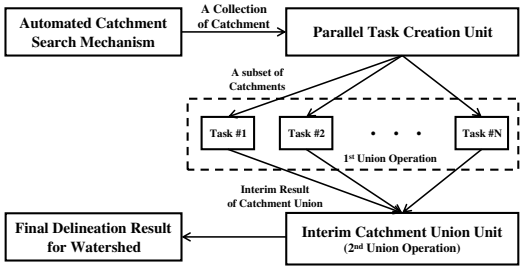


Fig. 5. Parallel-Union subsystem.

Parallel-Union: The second strategy is parallel-union that concurrently processes the geometric union via threading. Figure 5 shows the architecture of the parallel-union subsystem. The parallel-union starts with receiving a collection of all relevant catchments from the ACSM (described in Section III-B). The collected catchments are sent to the Parallel Task Creation Unit (PTCU). The PTCU partitions the catchments into k subsets evenly. The PTCU, then, initiates k parallel tasks and sends each subset of catchments to each task that performs the 1^{st} union operation on a subset of the catchments, and then sends its interim union result of catchment to the Interim Catchment Union Unit (ICUU). Once all parallel tasks have sent their interim results to the ICUU, the k interim results of catchments are merged by 2^{nd} union operation to create the final result for the target watershed.

A key issue of parallel-union is how to choose the proper number of parallel tasks for the watershed delineation. A common approach is to create the same number of tasks with the number of cores on a machine. (e.g. 4 parallel tasks for 4 core machine.) However, this approach does not necessarily work on virtualized environments such as VM on public clouds. In order to determine the proper number of parallel tasks, we will show variable evaluation results in Section IV.

The parallel-union approach was essentially designed to minimize the execution time of the watershed delineation on multi-core single machine. Even though we can leverage

TABLE II
STRATEGY SELECTION CRITERIA FOR CATCHMENT UNION PROCESSING.

Strategy	# of Catchments	# of VMs
Data-Reuse	Multi-HUC region case	1
Parallel-Union	# of Catchments $<$ 25K	1
MapReduce	# of Catchments $\geq$ 25K	$>$ 1

various VM types offered by AWS, there are limitations[4] to minimize the execution time of watershed delineation by the parallel-union strategy because of the physical limitations of the HW specifications. Leveraging a single machine is often insufficient for a certain scale of watersheds. For those large-scale watershed, we use multiple machines via MapReduce.

MapReduce: MapReduce is a common distributed programming paradigm consisting of two phases [14]. The first phase maps the data to an intermediate format. The second phase reduces the intermediate data to a final output. Although AWS offers on-demand MapReduce services such as EMR (Elastic MapReduce)[5], we pursue an Apache Hadoop[6], an open source implementation of MapReduce, cluster on AWS. We choose not to use EMR because this service limits our debugging capabilities for MapReduce jobs.

Hadoop allows us to distribute data and computation across several nodes. Hadoop parallelizes operations by creating containers that run the mapper and/or reducer. These containers consist of allocated virtual cores and memory. Also, Hadoop offers HDFS (Hadoop Distributed File System), which can redundantly store data across the cluster. HDFS partitions the data to distribute the computation across the clusters nodes and to stream as input into the map procedures. This partitioning has an impact on the performance of this system, which will be described in a later section.

This process of map and reduce intuitively resembles our system's current parallel-union model. The mapping phase of the geometric union is similar with the PTCU and the parallel tasks themselves shown in Figure 5. The reduce phase is similar with the ICUU. This strategy utilizes a Hadoop cluster on AWS to distribute the geometric union to multiple virtual machines. This allows us to achieve much more performance improvement than using the parallel-union on a single VM. Thus, when the collection of catchments for the target watershed is too large to be unioned in the required time, they are sent to the Hadoop cluster where they are processed with MapReduce and the result is then returned.

Strategy Selection Criteria: Three performance improvement strategies of ***WDCloud*** target different facets of the performance and therefore are not required for every input. Table II shows the strategy selection criteria to select a proper approach for union operation of catchments.

The data-reuse strategy is only utilized when a watershed crosses a NHD+ regional boundary. This is determined during catchment aggregation and its use largely depends on the input and catchments involved in the watershed.

The parallelization-based approaches (e.g. parallel-union and MapReduce) can be used in either single machine or a multiple number of machines. The parallel-union approach will be used when a single VM can provide the desired performance (e.g. total delineation time is less than 20 minutes). And MapReduce approach is for the case that a single VM cannot provide the desired runtime performance. Thus, the watershed delineation system automatically determines either approach based on the number of catchments to be unioned. If the number of catchment is less than 25K, the system assigns the request to a single VM on AWS and uses the parallel-union. Otherwise the system sends this request to Hadoop cluster on AWS for MapReduce operation.

[4]m1–m3 instance types in AWS normally have 1 to 8 of virtual CPU cores on a single VM [1].

[5]http://aws.amazon.com/elasticmapreduce/

[6]http://hadoop.apache.org

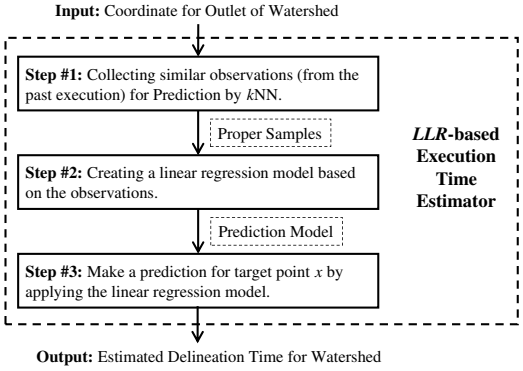


Fig. 6. Local Linear Regression-based execution time estimator for watershed delineation.

D. Execution Time Estimation for Watershed Delineation

As we pointed out in Section I, an accurate estimation of the execution time for the watershed delineation is an important issue to improve the scientists' productivity for their research. To estimate execution time of the delineation, we use LLR (Local Linear Regression) [20] based execution time estimator, which is investigated by our previous work [23]. Figure 6 shows the procedure of LLR estimator. This estimator takes a coordinate for outlet of the target watershed as its input parameter. The LLR estimator collects similar execution samples (from the past execution history) with the input coordinate using kNN method. kNN methods uses three features:

- The number of catchments for the target watershed.
- Geographical closeness to the input coordinate.
- Execution environments (e.g. VM type).

The next step is to create a simple linear regression model based on the collected samples from the kNN. The parameters (α and β represent the intercept and slope of the linear model) for the linear regression model can be calculated by minimizing the objective function in equation-1. In equation-1, x_0 represents the outlet coordinate for the watershed and V means the set of similar samples with the outlet coordinate.

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{x_i \in V} [y_i - \alpha(x_0) - \beta(x_0)x_i]^2 \quad (1)$$

LLR estimator, then, provides the estimated execution time ($f(x_0)$) for the input coordinate by the linear regression model obtained by the previous step.

$$f(x_0) = \alpha(x_0) - \beta(x_0)x_0 \quad (2)$$

E. VM Resource Management: Autoscaling

VM resources used by ***WDCloud*** are managed by an autoscaling mechanism. The autoscaling mechanism is designed to automatically manage both under- and over-provisioning of VMs for ***WDCloud***. Note that autoscaling of ***WDCloud*** is different mechanism from "Auto Scaling" offered by AWS [1]. The under-provisioning of VM resources can result in poor performance (e.g. slow response time) of ***WDCloud*** due to the lack of computing resources. Over-provisioning can hurt the cost-efficiency of ***WDCloud*** due to a number of idle VMs.

Algorithm 2 Scaling-Up Operation of Autoscaling

Require: job_{new}: new request for watershed delineation
1: $VMs \leftarrow$ get_all_running_VMs ()
2:
3: $i \leftarrow 0$
4: **while** $i <$ length(VMs) **do**
5: $comp\_time \leftarrow \sum Exec\ Time\ of\ Jobs$ in $Job\ Queue$ on $VM[i]$ + Job_{new}'s $Estimated\ Exec\ Time$
6: **if** $comp\_time <threshold$ **then**
7: $CandidateVMs$.append($VM[i]$, $comp\_time$)
8: **end if**
9: i++
10: **end while**
11:
12: **if** len($CandidateVMs$) > 0 **then**
13: sort($CandidateVMs$, "$comp\_time$")
14: assign_job_to_VM($CandidateVMs$[0], Job_{new})
15: **else**
16: $newVM$ = create_new_VM()
17: assign_job_to_VM($newVM$, job_{new})
18: **end if**

Algorithm 2 shows the scaling-up mechanism of the autoscaling. The scaling-up decision is triggered when a new delineation job (job_{new}) arrives. The autoscaling, then, obtains the information of currently running VMs (**ln 1**). The next step is that the autoscaling calculates the estimated job completion time on each running VM by the sum of estimated execution times of all existing jobs in work queue on the VM and the estimated execution time of job_{new} (**ln 5**). For this step, the autoscaling collaborates with LLR execution time estimator in Section III-D. The autoscaling compares the estimated completion time of the new job (job_{new}) with a threshold (**ln 6**), which is defined by the user (e.g. 30 minutes or 1 hour). If the estimated completion time of the new job is earlier then the threshold, the autoscaling stores the VM into a candidate VM list ($CandidateVMs$) for the job execution (**ln 7**). If existing VMs can completes the job_{new} within the threshold (**ln 12**), the job_{new} will be assigned to a VM that offers earliest completion of job_{new} (ln 13–14). Otherwise, the autoscaling creates a new VM (scaling-up) and assigned the new job to the new VM (**ln 16–17**).

For the scaling-down operation (Algorithm 3), the autoscaling uses the billing boundary-based VM scaling-down. Because

Algorithm 3 Scaling-Down Operation of Autoscaling

1: **while** true **do**
2: $VMs \leftarrow$ get_all_running_VMs()
3:
4: **for** $VM \leftarrow VMs$ **do**
5: **if** VM's running time % $Billing\ Bound$ == 0 && VM's status is $Idle$ && VM's Q is $Empty$ **then**
6: terminate_instance (VM)
7: **end if**
8: **end for**
9: **end while**

WDCloud runs on AWS, the autoscaling uses the hourly billing bound of AWS (**ln 5**). A key step of the scaling-down operation is that a VM will be terminated when the VM's running time is approaching the billing bound, the VM's status is idle state, and the Queue of the VM is empty (**ln 5–6**).

IV. Evaluation

In the evaluations of ***WDCloud***, we focus on two main aspects of ***WDCloud***, which are the performance improvement via three strategies (Section III-C) and the accuracy of the execution time estimation via LLR estimator (Section III-D).

A. Performance Improvement via Three Strategies.

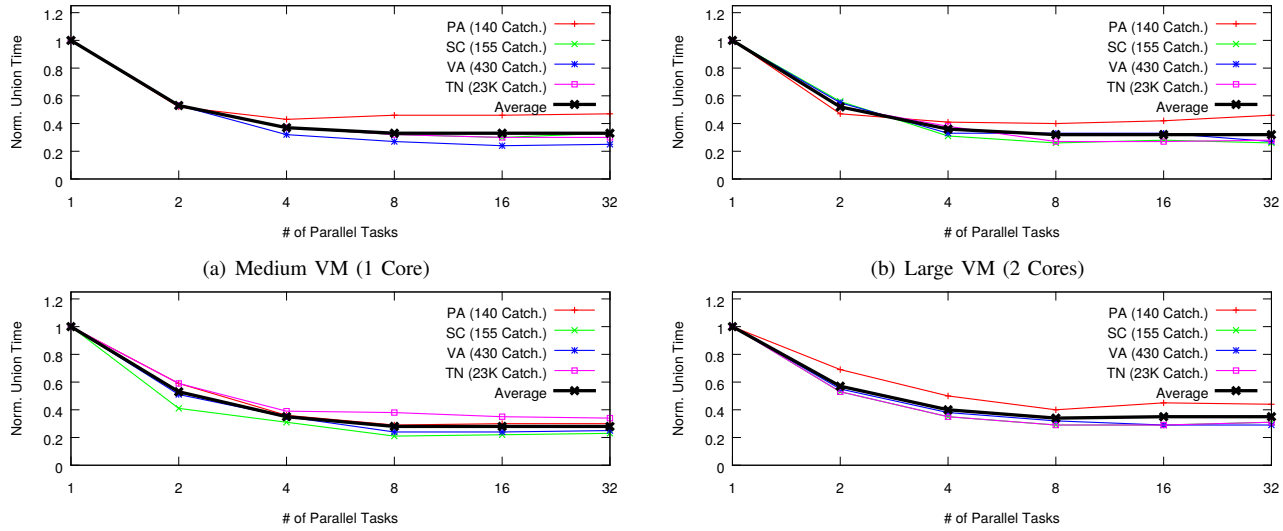
Data-Reuse: The main focus of data-reuse was the specific targeting of the largest watersheds (e.g. the Mississippi), those spanning multiple regions. By leveraging the segmentation of the NHD+, data-reuse achieves a 111x speedup as shown in Table III. This paragraph discusses about this large speedup.

TABLE III
Speed-Ups by Data-Reuse for the Mississippi Watershed.

Commodity Laptop	Data-Reuse	Speed-Up
10+ hours	5.5 mimutes	111x

The Mississippi watershed of 1,100,000+ catchments originally required 10+ hours to perform the geometric union. That same example required 5.5 minutes to union when utilizing the data-reuse strategy. To explain this large speed-ups, which only required the storage of 106 MB of pre-computed data, we describe an example execution. Our Mississippi example aggregates a total of 1,117,172 catchments. Once the data-reuse is used, the Mississippi example aggregates only 29,137 catchments. This equates to pre-computing the union of 1,088,035 catchments, which is 97% of the original catchments. This 111x speedup exceeded our initial estimates (e.g. less than 20 minutes) and resolved the largest class of watersheds, but several watersheds of approximately 100K to 250K catchments that took 4+ hours to delineate still remained. These large watershed will be dealt with other two parallelization strategies.

Parallel-Union: This strategy was designed to maximize the performance of a single VM, and is used for small- and medium-scale watersheds (# of catchments $<$ 25K). A key issue of the parallel-union is how to choose the proper number of parallel tasks for the watershed delineation. To determine the proper number of parallel tasks, we executes four example watersheds with 1 to 32 tasks on different types of VMs on AWS. These four watersheds contains less than 25K catchments, and



(a) Medium VM (1 Core)

(b) Large VM (2 Cores)

(c) Xlarge VM (4 Cores)

(d) 2Xlarge VM (8 Cores)

Fig. 7. Normalized geometric-union time of watershed delineation by parallel-union strategy on four types of VMs.

they are in Pennsylvania (140 catchments), South Carolina (155 catchments), Virginia (430 catchments), and Tennessee (23K catchments). We also used four different types of general purpose VMs (e.g. m1 instances) [1] for this evaluation. The results for the parallel-union evaluation are shown in Figure 7, and all results are normalized to the geometric union time from non-parallelization case (a single task). Back bold line in all graphs is an average of normalized geometric union time of four watersheds by the parallel-union strategy. The results show that, on average, the parallel-union provides the best performance improvement when ***WDCloud*** creates 8–32 tasks for the geometric union. (3x speed up on the medium VM, 3.1x speed up on the large VM, 3.6x speed up on the xlarge VM, and 2.9x speed up on the 2xlarge VM.) By using the parallel union strategy, we can complete the delineation for the four watersheds with 28–150 seconds (8 parallel tasks on 2xlarge instance). Without the parallel-union, these four watershed take approximately 500–3200 seconds (single tasks on medium instance). These results implies ***WDCloud*** with the parallel-union can handle small- and medium-scale watershed (# of catchments $<$ 25K), but this strategy is not sufficient to obtain enough performance improvement for large-scale watersheds (# of catchments $\geq$ 25K). Those large-scale watersheds will be handled by the MapReduce strategy.

MapReduce: To show the performance improvement by the MapReduce strategy, three large-scale watersheds on Hadoop cluster are examined. These three large-scale watersheds are located in Maine (66K catchments), Kentucky (107K catchments), and South Dakota (253K catchments). In this evaluation, ***WDCloud*** uses 4 to 32 cores of Hadoop cluster[7]. The evaluation results by the MapReduce strategy are shown in Figure 8. As shown in the graph, by leveraging 32 core Hadoop cluster, ***WDCloud*** can achieve 7x of speed-ups for Maine watershed (66K), 11x of speed-ups for Kentucky watershed (107K), and 21.2x of speed-ups for South Dakota watershed (253K). These results also show that the more catchments a watershed includes, the higher speed-ups ***WDCloud*** can achieve. For the South Dakota watershed, the delineation takes 4.2 hours with no parallelization, but the same delineation takes only 11.8 minutes with 32 core Hadoop cluster.

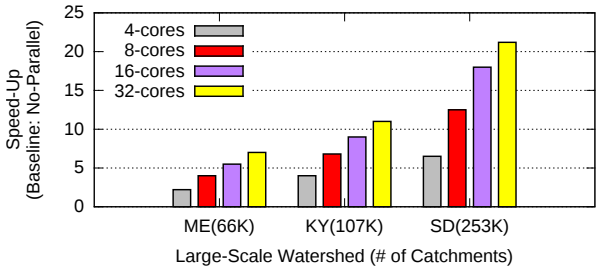


Fig. 8. Speed-up for geometric union of large-scale watershed by MapReduce.

B. Execution Time Estimation for Watershed Delineation

The next evaluation is to measure the performance of LLR estimator of ***WDCloud*** for predicting the execution time of watershed delineation. We employ prediction accuracy and MAPE (Mean Absolute Percentage Error) for this evaluation and these metrics are shown in equation (3)–(4). A higher result of prediction accuracy means better, and lower result of MAPE indicates better performance.

$$Pred.\ Accuracy = \begin{cases} \frac{T_{actual}}{T_{predicted}}, & T_{predicted} \geq T_{actual} \\ \frac{T_{predicted}}{T_{actual}}, & T_{predicted} < T_{actual} \end{cases} \quad (3)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{T_{actual,i} - T_{predicted,i}}{T_{actual,i}}\right| \quad (4)$$

[7]4 cores of Hadoop cluster uses 4 medium VMs (4 $\times$ 1 core). 8 cores of Hadoop cluster has 4 large VMs (4 $\times$ 2 cores). 16 cores of Hadoop cluster consists of 4 xlarge VMs (4 $\times$ 4 cores). 32 cores of Hadoop cluster is composed of 4 2xlarge VMs (4 $\times$ 8 cores).

TABLE IV
OVERALL EVALUATION RESULTS FOR EXECUTION TIME ESTIMATION

	LLR **Estimator**	***k*NN**	***mean***
Prediction Accuracy	85.6%	65.7%	42.8%
MAPE	0.19	0.93	1.97

As the baselines of this evaluation, we use kNN [20] and $mean$ [25]. For the execution time estimation, kNN uses three features that are 1) geographical closeness to the target watershed, 2) the number of catchments, and 3) the type of VM instances, which are the same with LLR estimation.

We measures 420 random coordinates (20 random coordinates of watershed outlets $\times$ 21 HUC regions) for execution time estimation. The overall results are shown in Table IV. As shown in Table IV, LLR estimator outperforms other two approaches. The prediction accuracy of LLR estimator is 85.6%, which is 19.9% and 42.8% higher than kNN and $mean$-based estimator. The MAPE result of LLR is 0.19, which is 4.9x and 10.4x lower (better) than others.

Moreover, to show the performance of LLR estimator that can precisely estimate the delineation time for watershed outlets on all 21 HUC regions in NHD+, we show the estimation results of three estimators based on each HUC regions. Figure 9 shows the estimation results on all 21 HUC regions. For the prediction accuracy on all 21 HUC regions (Figure 9(a)), LLR estimator shows over 80% of prediction accuracy for all 21 regions. For the MAPE results (Figure 9(b)), LLR estimator has accurate MAPE results, which are less than 0.23, except for only two HUC regions (08 and 13 HUC regions). These results show that LLR estimator provides reasonable estimation for the execution time of the watershed delineation, and can provide precise estimation results for almost all HUC regions in NHD+.

V. RELATED WORK

Geospatial data analysis research has benefited from the technical advancements of cloud computing. Several works have shown the clouds ability to provide performance improvements to these data and compute intensive applications.

A system to increase the performance of watershed calibration by utilizing the cloud was designed by Humphrey et al [22]. This system reduced an 11 hour computation to a 5 minute computation by using cloud computing. They focused on core utilization and the parallelization of the application; instead we also focus on using specific characteristics of the application (e.g. Data-reuse) to improve the runtime without such large compute clusters. Furthermore, we incorporate MapReduce to distribute our computation instead of performing it manually.

Caching is similar in style to the data-reuse strategy of this work. Chiu et al [13] proposed a strategy for caching in the cloud. This caching is most useful for the results of a Service-Oriented Application (SOA), but not useful for our applications storage of intermediate data formats. The data-reuse is similar to caching, but not equivalent because the pre-computed data will never be swapped while the system is online.

Several works [8, 18, 27] utilize MapReduce to enhance their GIS and spatial data analysis applications. Hadoop-GIS [8] was designed to improve spatial query processing capabilities of GIS via adopting MapReduce. Hadoop-GIS supports several spatial enhancement capabilities such as spatial data partitioning for parallel processing, and spatial query processing. Hadoop-GIS is also integrated with Hive. SpatialHadoop [18] is a low-level extension of Hadoop and supports spatial indexing for its input dataset to facilitate the spatial data processing. Dart [27] is another type of GIS on Hadoop. Dart is collaborating with HBase and provides a hybrid table schema to store spatial data in HBase. Dart utilizes public cloud infrastructure such as Amazon EC2. These research are relevant to our work, but our work is different because we employ MapReduce as a part of our three strategies to improve the performance of geometric union in the watershed delineation process.

Alencar et al [9] have reported an on-going research project to employ cloud infrastructure and capabilities to watershed research. Their work is to build a cloud-based collaborative platform for watershed research. A difference from our work is that they focus on giving collaborative capabilities (e.g. hydrology data sharing) for stakeholders (e.g. scientists and decision makers) to watershed research system via CometCloud [16]. ***WDCloud*** does not consider having collaborative capabilities for watershed delineation. Another difference is that it is unclear which raw dataset they use for their project.

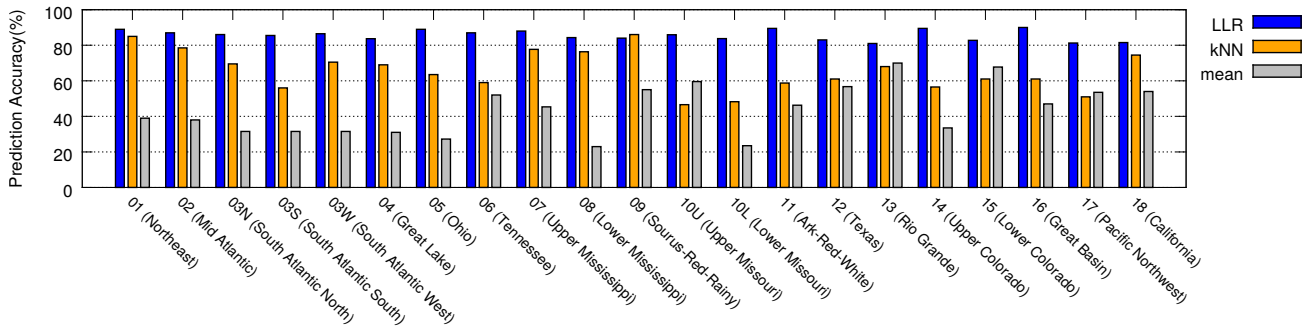
VI. CONCLUSION

Watershed delineation is a process to determine the area draining to a point on the land surface. This plays a critical role for hydrologic and water resources research because watershed delineation is often the first step of an analysis. However, existing watershed delineation tools are insufficient to support hydrologists because they have not kept pace with new datasets that allow for national-scale watershed delineation over the web without requiring extensive data preprocessing steps. Watershed delineation applications lack the capabilities to fully leverage scalable and high performance computing infrastructure (e.g. public cloud), and provide predictable performance for the delineation tasks.

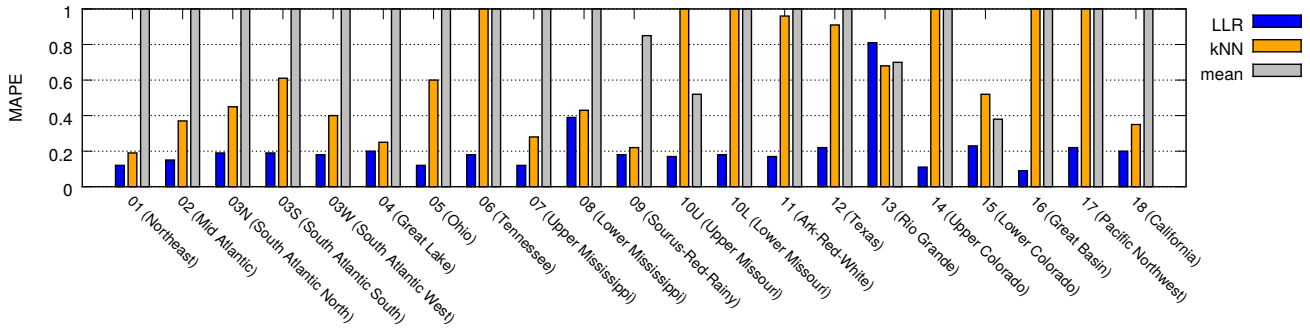
To solve these problems, this paper reports on ***WDCloud***, which is a system for large-scale watershed delineation on AWS. ***WDCloud*** employed three key approaches:

- An automated catchment search mechanism for NHD+, which is a public watershed dataset from USGS.
- Three performance improvement strategies: Data-reuse, parallel-union, and MapReduce.
- LLR execution time estimator for watershed delineation.

Our evaluations on ***WDCloud*** mainly focus on 1) the performance improvement for watershed delineation via three strategies and 2) the prediction accuracy for delineation time by LLR estimator. In terms of the speed up of watershed delineation tasks, ***WDCloud*** achieves 111x speed up for the Mississippi watershed (the largest watershed in U.S.) through the data-reuse strategy, up to 21x speed up for large-scale watershed via MapReduce, and 18x speed up for medium- and small-scale watershed by using the parallel-union approach.



(a) Prediction Accuracy of three estimators on 21 HUC regions



(b) MAPE of three estimators on 21 HUC regions

Fig. 9. Execution time estimation results of three estimators on 21 HUC regions.

Moreover, the LLR estimator of ***WDCloud*** provides the reliable execution time estimation of watershed delineation with 85.6% of prediction accuracy. This result is 23%–43% better than other state-of-the-art estimation approaches.

REFERENCES

[1] Amazon Web Services. http://aws.amazon.com.

[2] Microsoft Azure. http://azure.microsoft.com.

[3] ESRI – ArcGIS Watershed. http://www.arcgis.com/home/item.html?id=8e48f6209d5c4be98ebbf90502f41077.

[4] Wikipedia – Keyhole Markup Language. http://en.wikipedia.org/wiki/Keyhole_Markup_Language.

[5] National Hydrography Dataset – USGS. http://nhd.usgs.gov, .

[6] NHDPlus Version 2 – Horizon Systems. http://www.horizon-systems.com/nhdplus/NHDPlusV2_home.php, .

[7] USGS – StreamStats. http://water.usgs.gov/osw/streamstats/.

[8] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. Hadoop-GIS: A High Performance Spatial Data Warehousing System over Mapreduce. In *Proc. VLDB Endowment*, 2013.

[9] P. S. C. Alencar, D. D. Cowan, F. McGarry, and R. M. Palmer. Developing a Collaborative Cloud-based Platform for Watershed Analysis and Management. In *Proc. IEEE CollaborativeCom*, 2014.

[10] A. M. Castronova and J. L. Goodall. A Hierarchical Network-based Algorithm for Multi-Scale Watershed Delineation. *Computers & Geosciences*, 72, 2014.

[11] C. L. Chang. The Impact of Watershed Delineation on Hydrology and Water Quality Simulation. *Environment Monitoring and Assessment*, 148, 2009.

[12] D. Chen, S. Shams, C. Carmona-Moreno, and A. Leone. Assessment of open source GIS software for water resources management in developing countries. *Journal of Hydro-environment Research*, 4, 2010.

[13] D. Chiu, A. Shetty, and G. Agrawal. Elastic Cloud Caches for Accelerating Service-Oriented Computations. In *Proc. SC*, 2010.

[14] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. USENIX OSDI*, 2004.

[15] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The Cost of Doing Science on the Cloud: The Montage Example. In *Proc. SC*, 2008.

[16] J. Diaz-Montes, M. AbdelBaky, M. Zou, and M. Parashar. CometCloud: Enabling Software-Defined Federations for End-to-End Application Workflows. *IEEE Internet Computing*, 19, 2015.

[17] D. Djokic and Z. Ye. DEM Preprocessing for Efficient Watershed Delineation. In *Proc. '99 ESRI Intl. User Conference*, 1999.

[18] A. Eldawy and M. F. Mokbel. SpatialHadoop: A MapReduce Framework for Spatial Data. In *Proc. 31th IEEE ICDE*, 2015.

[19] M. B. Ercan, J. L. Goodall, A. M. Castronova, M. Humphrey, and N. Beekwilder. Calibration of SWAT models using the cloud. *Environmental Modeling & Software*, 62, 2014.

[20] T. Hastie, R. Tibshirani, and J. Friedman. The Element of Statistical Learning: Data Mining, Inference, and Prediction. 2011.

[21] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the Use of Cloud Computing for Scientific Workflows. In *Proc. IEEE eScience*, 2008.

[22] M. Humphrey, N. Beekwilder, J. L. Goodall, and M. B. Ercan. Calibration of Watershed Models using Cloud Computing. In *Proc. IEEE eScience*, 2012.

[23] I. K. Kim, J. Steele, Y. Qi, and M. Humphrey. Comprehensive Elastic Resource Management to Ensure Predictable Performance for Scientific Applications on Public Iaas Clouds. In *Proc. 7th IEEE/ACM UCC*, 2014.

[24] S. Kopp. Custom Watersheds at the Click of a Button: Watershed Delineation in ArcGIS Online. *ArcGIS Resources ESRI*, Aug, 13, 2013.

[25] W. Smith, I. Foster, and V. Taylor. Predicting Application Run Times with Historical Information. In *Proc. JSSPP*, 1998.

[26] M. P. Strager, J. J. Fletcher, J. M. Strager, C. B. Yuill, R. N. Eli, J. T. Petty, and S. J. Lamont. Watershed analysis with GIS: The watershed characterization and modeling system software application. *Computers & Geosciences*, 36, 2010.

[27] H. Zhang, Z. Sun, Z. Liu, C. Xu, and L. Wang. Dart: A Geographic Information System on Hadoop. In *Proc. IEEE Cloud*, 2015.