This is an Accepted Manuscript of an article published in Environmental Modelling & Software in 2014, available online: doi::10.1016/j.envsoft .2014.09.002

Calibration of SWAT models using the Cloud

Mehmet B. Ercan[a], Jonathan L. Goodall[b,a,d], Anthony M. Castronova[a], Marty Humphrey[c], Norm Beekwilder[c]

[a] *University of South Carolina, Department of Civil and Environmental Engineering, 300 Main St., Columbia, SC 29208 USA*

[b] *University of Virginia, Department of Civil and Environmental Engineering, 351 McCormick Road, P.O. Box 400742, Charlottesville, VA 22904-4742*

[c] *University of Virginia, Department of Computer Science, 85 Engineers Way, P.O. Box 400740, Charlottesville, VA 22904-4740*

[d] *corresponding author: goodall@virginia.edu*

Abstract

This paper evaluates a recently created Soil and Water Assessment Tool (SWAT) calibration tool built using the Windows Azure Cloud environment and a parallel version of the Dynamically Dimensioned Search (DDS) calibration method modified to run in Azure. The calibration tool was tested for six model scenarios constructed for three watersheds of increasing size each for a 2 year and 10 year simulation duration. Results show significant speedup in calibation time and, for up to 64 cores, minimal losses in speedup for all watershed sizes and simulation durations. An empirical relationship is presented for estimating the time needed to calibration a SWAT model using the cloud calibration tool as a function of the number of Hydrologic Response Units (HRUs), time steps, and cores used for the calibration.

Keywords: Model Calibration, Cloud Computing, Watershed Modeling, SWAT, Windows Azure

1. Introduction

In recent decades, computer simulation of hydro-environmental systems has been driven by the need to provide estimates of non-point source pollution and its impact on waterbodies. While various approaches have been used for watershed-scale simulation, distributed continuous time-step simulation modes are the most advanced. This is attributed to their ability to accurately simulate overland flow and its interaction with soil and plants, which is a primary source of chemical activities that influence water quality (Arnold et al. 1993; Kirkby et al. 1996; Graham and Butts 2005). Watershed models often require data such as soil and land cover type, terrain elevation and slopes, and historical weather data to perform a simulation. For watershed models of even moderate complexity, model execution can consume a considerable amount of time.

Calibration of a simulation model is a process which aims to provide estimates of model parameters values that minimize the error between model predictions and measured observations. In watershed modeling, calibration is arguably the most computationally demanding step in creating an accurate model. There has been significant work in the area of watershed model calibration. One contribution to highlight is the Multi-Objective Complex Evolution (MOCOM-UA) method proposed by Yapo et al. (1998), which is a global optimization algorithm based on the Shuffled Complex Evolution (SCE) (Duan et al., 1993). This method has been widely applied and illustrates an effective method for performing multi-objective calibration using the Daily Root Mean Square (DRMS) and Heteroscedastic Maximum Likelihood Estimator (HMLE) objective functions. A second contribution to highlight is Vrugt et al. (2003) that presented a Markov Chain Monte Carlo sampler calibration method that efficiently and effectively solves the multi-objective optimization problem for hydrologic models. While these approaches offer innovative solutions to multi-objective optimization, they do not drastically reduce the time necessary to calibrate a model. Using these or other calibration methods, it can often take days to complete a single calibration for models depending on the size of the watershed, simulation duration, and data resolution.

While there are numerous examples of algorithms applicable for watershed calibration, there are few examples of approaches aimed at overcoming the computational challenges needed to speedup calibration time. One example of such an attempt is Rouholahnejad et al. (2012) which introduced a parallel calibration routine for the Soil and Water Assessment Tool (SWAT). In this work, the authors tested the Sequential Uncertainty Fitting (SUFI2) optimization algorithm (Abbaspour et al., 2004) using three different watershed models of various sizes within a high performance computing environment. Their results show how computational efficiency can be achieved for SWAT models by leveraging multiple CPUs in parallel. This past work, however, did not make use of a cloud computing infrastructure. A second example is our recent work that presented an Azure-based SWAT calibration tool that uses a parallel version of the Dynamically Dimensioned Search (DDS) method for calibrating a SWAT model (Humphrey et al., 2012). DDS was proposed by Tolson and Shoemaker (2007) as a calibration method and is capable of optimizing a hydrologic model parameter set in fewer iterations than the aforementioned SCE calibration method (Duan et al., 1993). With a parallel version of this calibration routine (Tolson et al., 2007), it was possible to implement the DDS Method in the Azure cloud and provide calibration runs that used up to 256 cores (Humphrey et al., 2012). In Humphrey et al. (2012) we presented the design and implementation of the cloud-based SWAT calibration tool, but did not offer a detailed evaluation or testing of the tool across a range of typical watershed sizes and simulation durations.

Cloud computing offers quick and easy access to shared pools of configurable computing resources that can be utilized with minimal management effort and essentially no service provider interaction (Mell and Grance, 2011). It presents an attract means for calibrating watershed models because calibration is performed relatively infrequently by watershed modelers, making it a good candidate for a pay-for-use cost model rather than having to invest in computer hardware capital and maintenance costs. However, there has not been work completed to date that quantifies the cost of calibrating a SWAT watershed model using the cloud, so modelers do not have the information needed to understand the tradeoffs between using a personal computer, a cluster, or the cloud for performing model calibrations.

Given this motivation, the goals of this study are to (i) evaluate the ability of a parallel, cloud-based calibration tool for SWAT presented in Humphrey et al. (2012) to converge on an objective function as additional cores are used for the calibration, (ii) quantify calibration time and speedup gained by using the cloud calibration tool across different sized watersheds, model durations, and number of cores used for the calibration, and (iii) quantify the cost of calibrating a watershed model using the cloud tool for different sized watersheds, model durations, and number of cores used for the calibration. In the following section we provide a brief background of the SWAT model, the DDS calibration algorithm, and the Humphrey et al. (2012) implementation of DDS in the Azure cloud. Next, the design of several SWAT simulations and the methodology used for calibrating them using the DDS algorithm is presented. This is followed by an analysis of the calibration results, including speedup and cost analyses for the different sized watershed models and simulation durations. Finally, we conclude with brief summary of the study findings.

2. Background

Background information on SWAT, the DDS calibration method, and cloud computing are presented to orient readers to the key concepts and terminology used in this study. The cloud-based calibration tool evaluated through this work is also briefly summarize from the perspective of a SWAT modeler; readers interested in a more technically detailed description of the system should refer to Humphrey et al. (2012).

2.1. Soil and Water Assessment Tool (SWAT)

SWAT is a distributed, continuous time watershed model that is capable of running on a daily and sub-daily time steps (Gassman et al., 2007). It was originally developed to better understand the impact of management scenarios and non-point source pollution on water supplies at a watershed scale (Arnold et al., 1998). It has been used in a variety of watershed studies that include both water quantity and quality simulations (Lee et al., 2010; Liu et al., 2013; Setegn et al., 2010; Zhenyao et al., 2013). The SWAT model uses the concept of a Hydrologic Response Unit (HRU) for representing variability within subbasins of a watershed. HRUs are unique representations of land cover, soil, and management characteristics within a single subbasin and are used for water balance calculations within the model. HRUs are not spatially contiguous and therefore are often composed of many disjointed parcels land within a watershed.

2.2. Dynamically Dimensioned Search (DDS)

Dynamically Dimensioned Search (DDS) is a calibration method developed by Tolson and Shoemaker (2007) to reduce the number of iterations needed to achieve optimal parameter values for a watershed model. DDS is a heuristic global search algorithm in which the number of iterations is defined by the user. The algorithm starts globally by changing all the parameter values and changes to a more local search when the iterations approaches the user defined maximum allowable iteration. This is done by reducing the number of parameters in the calibration parameter set. The parameters in the calibration parameter set and the perturbations magnitudes are selected randomly without reference to sensitivity. Tolson and Shoemaker (2007) used the Town Brook (37 km^2) and the Walton/Beerston (913 km^2) SWAT watershed models to test DDS algorithm. The Town Brook watershed calibrated with 14 flow calibration parameter. Their results showed that the DDS method with 2500 iterations outperformed the well established Shuffled Complex Evolution (SCE) calibration method as well as two Matlab optimization tools (the fmincon and fminsearch functions).

2.3. Cloud Computing

The broad definition of cloud computing encapsulates applications used over the Internet, as well as the hardware and system software provided from data centers (Armbrust et al., 2010). While there are currently several public and private cloud computing services, this work utilizes the Microsoft Azure Platform. Microsoft categorizes its platform as a hierarchy of service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). SaaS provides business-level functionality in which users can quickly develop and deploy software applications on the cloud. PaaS offers less abstraction than SaaS by providing access to the virtualized infrastructure that the software systems run on. Finally, IaaS offers the least amount of abstraction and is likened to a physical server (or Virtual Machine, VM) requiring a high level of interaction, but also providing the most control (Vaquero et al., 2008). The cloud-based calibration tool evaluated through this work leverages the Azure IaaS functionality (Humphrey et al., 2012).

2.4. Parallel DDS in the Cloud

Adapting DDS to the Azure environment presented some issues due to Azure's parallel nature. The Microsoft Windows Azure HPC Scheduler (AzureHPC, 2012) allows launching and managing high-performance computing (HPC) applications and parallel computations within the cloud environment. Thus, the Windows Azure HPC Scheduler was used to perform job submissions. To function in a parallel environment it was necessary to modify the DDS algorithm. In the single-threaded version of DDS, during each iteration of DDS the previous model execution results are evaluated and, if better than the current best parameter set, are used to create a new parameter set for the next model execution. This "lock-step" approach does not trivially work in a multi-core environment. Building from prior work describing a parallel DDS algorithm (Tolson et al., 2007), this problem was solved by producing numerous initial parameter sets based on the number of cores available and submitting them in parallel to VMs in the cloud. As simulations completed, their results were applied to an objective function and stored in a high availability SQLAzure database. This allowed all the workers to easily find the current best parameter set. Thus, if more satisfactory result was obtained, the next parameter set is produced based on it. This is a slight difference compared to the Tolson et al. (2007) approach where cores do not need to wait for all jobs in a current batch to complete before proceeding. The system architecture of the DDS SWAT calibration tool on Windows Azure platform is further described by Humphrey et al. (2012).

The user provides the SWAT input files and various settings through a Web browser interface (Figure 3) to calibrate the watershed of interest on the cloud resources (Figure 1). The settings and options provided on the Web browser include streamflow gage ID, calibration parameters, objective function, and stopping criteria (number of iterations). Once the user uploads the SWAT input files and inputs the settings, the streamflow observations for the provided gage ID are downloaded using Web services from the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) (Tarboton et al., 2009). Next, the cloud calibration tool begins as previously described. The Web browser allows the user to monitor job submissions and download the model input/output file directory of the resulting calibrated model.

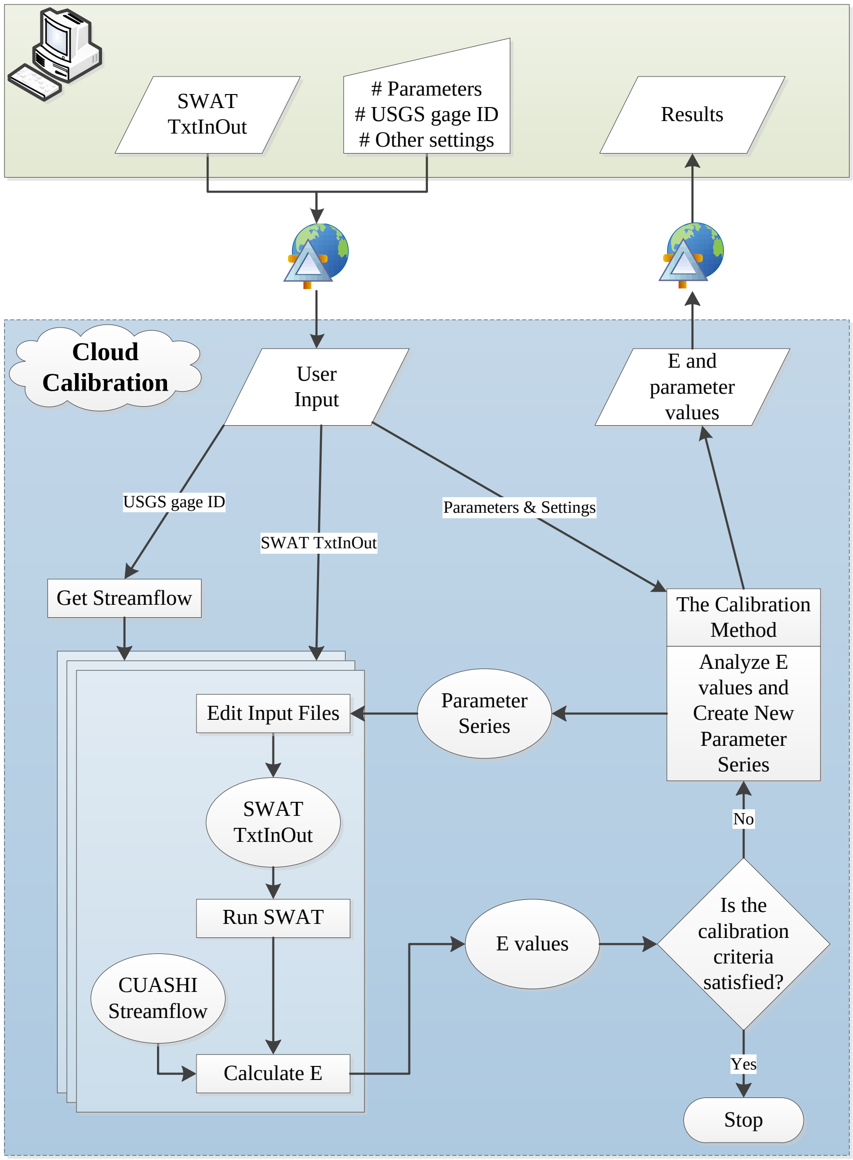

Figure 1: Cloud calibration tool system architecture (adapted from Humphrey et al., 2012)

3. Methodology

Fist, the cloud calibration tool was evaluated using an increasing number of cores and the results were compared to the execution of the tool using a single core. We used a SWAT model of the Eno watershed in North Carolina (171 km^2) that had 6 subwatersheds and 65 HRUs for a 2 year simulation period to perform the study. The parallelized DDS scenarios were compared to the one core execution best efficiency value in terms of the number of iterations required to reach the one core best efficiency value. The evaluation tests were each executed for 1000 iterations using 8 calibration parameters. Results of this evaluation are presented in Section 5.1.

We next ran a series of tests using the cloud calibration tool to quantify calibration time, speedup, and cost across three different watershed sizes (small, medium, and large) and two different model durations (short and long). The *small* watershed model was the Eno watershed model described in the prior paragraph. The *medium* watershed model was built for the Upper Neuse watershed (6,210 km^2) with 91 subwatersheds and 1064 HRUs. The Upper Neuse watershed is an 8-digit Hydrologic Unit Code (HUC) watershed using the U. S. Geological Survey (USGS) hydrologic unit system. Finally, the *large* watershed model was built for the Neuse watershed (14,300 km^2) with 177 subwatersheds and 1,762 HRUs (Figure 2). For comparison purposes, the Neuse includes 4 different 8-digit HUCs and is a 6-digit HUC itself. The *short* model duration was a 2 year simulation with a daily time step while the *long* model duration was a 10 year simulation also with a daily time step. The first half of these simulation durations were used as an equilibration (spin-up) period needed to establish initial conditions in the hydrologic model.

For comparison, we first ran the model scenarios on a personal computer with a serial implementation of the DDS method. We then used the cloud to calibrate the same model scenarios using 1, 2, 4, 8, ... and 256 cores. For consistency we used 1000 iterations and 8 flow calibration parameters for each test. We used 1000 iterations because the DDS algorithm is generally able to produce an optimized model with 1000 iterations and a greater number of iteration changes only results in insignificant changes in the objective function (Tolson and Shoemaker, 2007). The results of these tests are included in Sections 5.2 (calibration time), 5.3 (speedup), and 5.4 (cost). Finally, an empirical cost model is presented in section 5.5 for estimating the cost to calibrate a SWAT model in the cloud-based calibration tool based on characteristics of the SWAT model.

4. Model Development

The Neuse watershed (Figure 2) includes both the Upper Neuse and Eno watersheds. The Neuse watershed is a mostly rural, although it includes the Research Triangle Park region that includes the cities of Durham, Chapel Hill, and Raleigh. The climate is mild and the watershed has gently rolling topography. The soil type of the watershed is dominated with sandy clay loam in the lower portions of the basin and silty clay and loam soils in the upper part of the basin. The land cover of the watershed is dominated with forest and cultivated crops, in addition to the urbanized areas in Research Triangle Park.

Terrain and land cover data for the Neuse watershed were obtained from the United States Geological Survey (USGS) National Elevation Dataset (NED) and National Land Cover Database (NLCD) products with the resolution of 10 and 30 m, respectively. Soil data were obtained from an ArcSWAT-

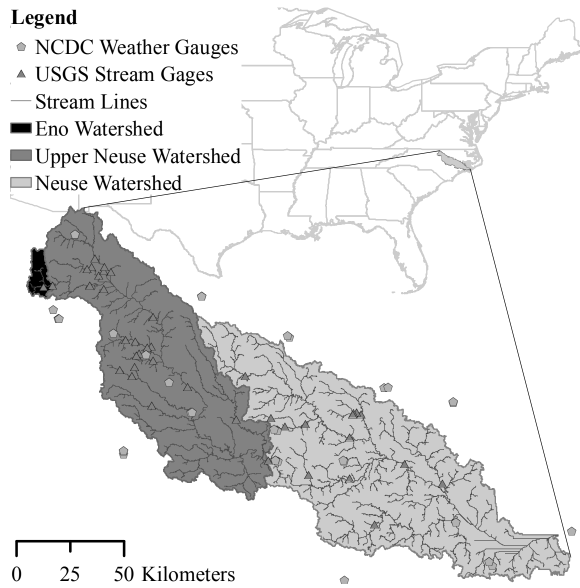

Figure 2: The three nested watersheds used for the analysis

provided soils raster with a 250 m resolution. This soils raster is based on the State Soil Geographic (STATSGO) dataset provided by the United States Department of Agriculture (USDA). Weather data including precipitation, temperature, wind speed and humidity were obtained for the period 2000 to 2010 from the National Climatic Data Center (NCDC) and included 6, 21, and 40 weather stations near the Eno, Upper Neuse, and Neuse watersheds, respectively. Daily average streamflow data were obtained for each watershed's outlet station (station numbers 02085000, 02089000 and 02091814) for the simulation period 2000 to 2010. These data were used to create the Eno, Upper Neuse, and Neuse SWAT watershed models using ArcSWAT (Winchell et al., 2008).

We divided each watershed model into subbasins based on the USGS streamflow station locations and the river network topology. When creating Hydrologic Respond Units (HRUs) for each subbasin, we used threshold values of 10% for soil, slope, and land cover to reduce variability within the subbasins. The final models for the Eno, Upper Neuse, and Neuse watersheds were divided into 6, 91, and 177 subbasins, respectively. The SWAT documentation recommends between 1 to 10 HRUs per subbasins. Therefore the Eno model included 65 HRUs while the Upper Neuse and Neuse models had 1064 and 1762 HRUs, respectively. The model was configured to use the Natural Resources Conservation Service (NRCS) Curve Number (CN) method (Kenneth, 1972) to calculate surface runoff, the Penman-Monteith method (Allen 1986; Allen et al. 1989) to calculate potential evapotranspiration (PET), and the variable storage routing method for channel routing. These are commonly used settings for performing simulations with SWAT.

4.1. Model Calibration

Once the SWAT model input files were prepared, the I/O directory for the SWAT model was compressed and submitted for calibration through the SWAT cloud calibration website interface (Figure 3). The objective function can be set to maximize either the daily or monthly Nash-Sutcliffe model efficiency coefficient (E) value (Nash and Sutcliffe, 1970). We selected to maximize the daily E value because there were available data (e.g. precipitation, streamflow) to support a daily time step model simulation. We used a fixed number of iterations as the stopping criterion. Finally, the USGS streamflow gage ID, outlet subbasin number, and eight calibration parameters were supplied through the SWAT calibration interface. Once a model has been submitted for calibration, the tool returns a job ID that can be used to track the calibration status and download the final, calibrated model.

The Eno, Upper Neuse, and Neuse watershed models for 2 and 10 year simulations were calibrated three times for each number of cores (from 1 to 256). When there was an inconsistency in execution time for a scenario, we increased the number of executions up to 8 to reach agreement. When analyzing the results, the time required to upload and download models to and from the cloud was not taken into account as any variability in this time for a given model size and duration was attributed to variability in network connection speed between the client and Azure head node. The size of compressed model input files were 0.6, 6.8 and 10.5 MB for the Eno, Upper Neuse, and Neuse watersheds, respectively. Therefore, it should take approximately 17 seconds to upload the largest of the three models assuming a 5 Mbps network speed, which is minor compared to the overall model calibration time.

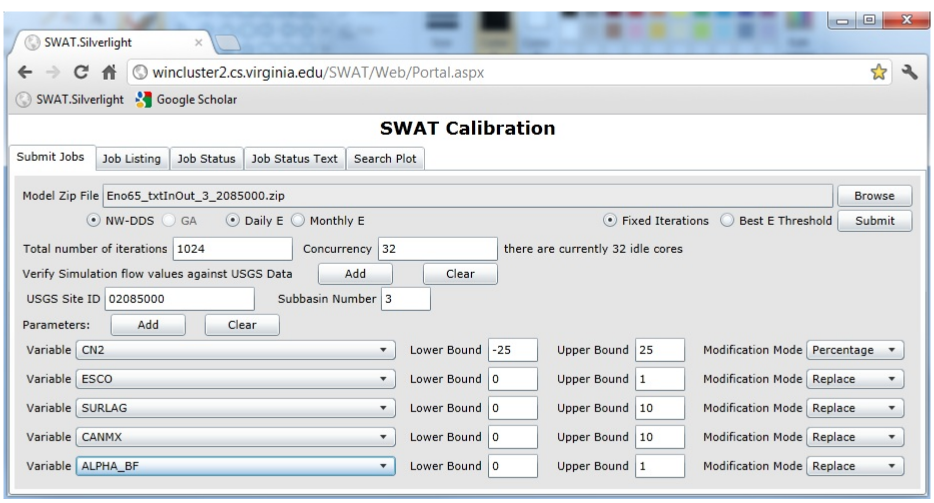

Figure 3: Cloud calibration tool user interface

5. Results and Discussion

5.1. Tool Evaluation

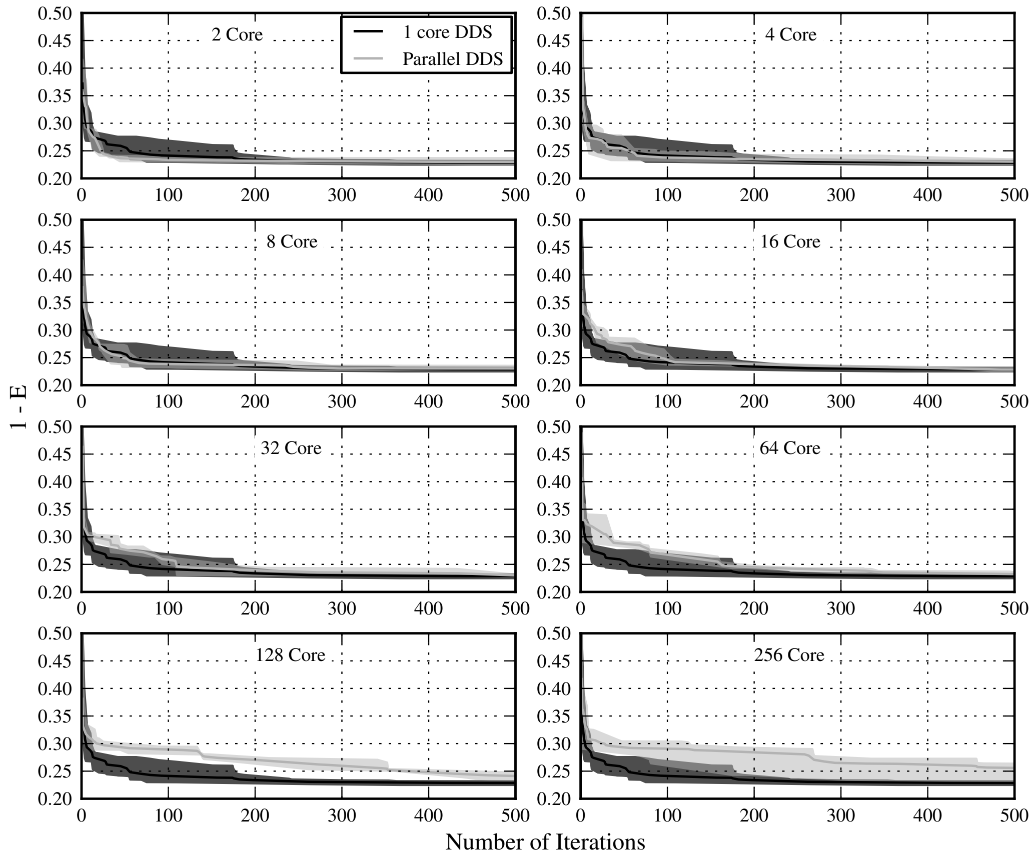

Figure 4: Objective function convergence with respect to cloud core number

Figure 4 shows a comparison between parallelized DDS on 2 to 256 cores and non-parallelized (1 core) version of DDS in the cloud. The objective function is "1 - Nash-Sutcliffe coefficient (E)" which indicates a better model as the value approaches zero. The shaded area shows the variability between different executions of the same scenario and solid lines show the average objective function value across all executions of the same scenario. This observed variation for the same scenario is a property of the DDS algorithm (Tolson and Shoemaker, 2007). The results show that on average a 1 core DDS execution will converge on the 200th iteration with an objective function value of 0.234 (Figure 4 and Table 1). The number of iterations required to coverage on this objective function value range between 71 and 285 over the 1 core test runs we conducted. Using this convergence value as a basis for comparison, the 2, 4, 8 and 16 core DDS executions on average converged on this same objective function value on the 96th, 288th, 242th and 319th iterations, respectively. Taking the range of required iterations for convergence into account (Table 1) shows similarity between the scenarios using 16 or fewer cores. For higher core number executions, the best objective function from previous runs are updated less frequently resulting in addition iterations required for convergence. Convergence was achieved on average on the 438th, 443th and 698th iterations for 32, 64, and 128 core executions, respectively. For the 256 core execution, the objective function value was on average within 98% of the convergence value of 0.234 after 1000 iterations. Although slower convergence speed was observed for these higher core executions, the approach still produces continuous improvement in the objective function value in part because Virtual Machines (VMs) do not need to wait for all jobs in a batch (i.e., the initial 256 jobs set out when using 256 cores) to complete before starting a new iteration (Humphrey et al., 2012).

5.2. Calibration Times

For comparison purposes, the DDS calibration algorithm was first executed on a personal computer (64-bit Intel Core i7 2.8 Ghz CPU with 4 GB

Table 1: Number of iterations for convergence ($1 - E = 0.234$)

Core Number	Iteration Number		
	Average	Minimum	Maximum
1	200	71	285
2	96	42	696
4	288	35	529
8	242	47	575
16	319	286	355
32	438	110	485
64	443	314	509
128	698	542	1000+
256	1000+	669	1000+

of RAM) running Windows 7. A two year calibration of Eno, Upper Neuse, and Neuse watersheds took 1 hour, 28 hours (1.2 days), and 51 hours (2.1 days), respectively. Ten year calibration executions took 6 hours, 113 hours (4.7 days), and 207 hours (8.6 days) for the Eno, Upper Neuse, and Neuse watersheds, respectively.

For the cloud implementation of the DDS calibration algorithm, we ran the Eno, Upper Neuse and Neuse watershed simulations over 2 and 10 year simulation durations. The results are shown in Figure 5 where the solid lines for each plot represent the average calibration time and the shaded areas represent the minimum and maximum calibration times. As expected, the general trend shows a decrease in calibration time with more cores, smaller watershed size, and shorter simulation durations. Although the models have different sizes and simulation durations, their calibration times decrease at a similar rate.

In general, the variability in calibration time increases when more cores are used for the calibration or for a model simulation with a longer duration (Figure 5). Less variability was seen in the 2 year simulation duration for up to 64 cores, whereas there was more variability in 10 year simulation starting with even 8 cores. It is difficult to explain the cause of the variability in calibration times in part because Azure is a shared platform and the network traffic and performance of an individual VM will be impacted by the number of active users at any given time. Furthermore, VMs are rented to users with an estimated rather than exact specification (CPU and RAM), causing additional variability in calibration times. Nonetheless, there are general patterns in the results that can be used to provide rough estimates of calibration time, a topic explored further in Section 5.5.

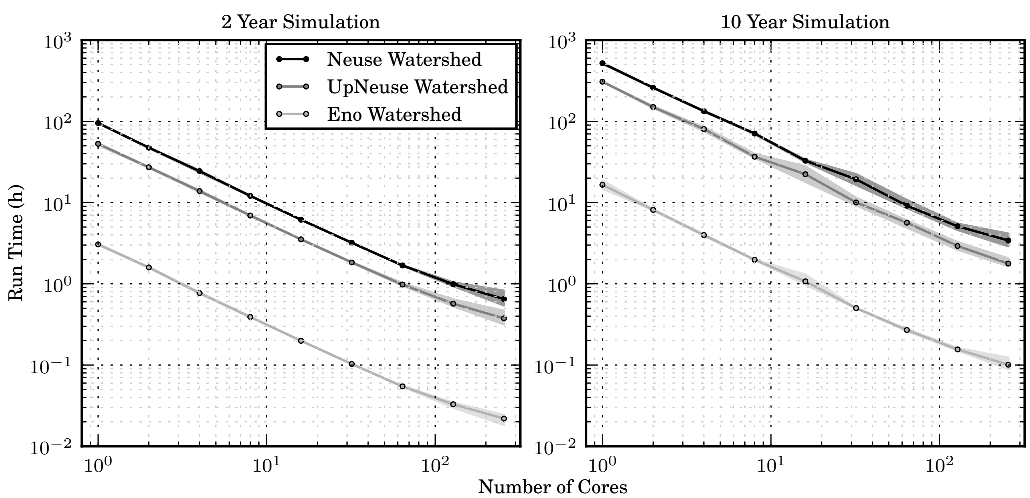

Figure 5: Run time to calibrate the Neuse, Upper Neuse, and Eno watersheds for 2 and 10 year simulation durations using different numbers of cores

5.3. Speedup

Speedup was calculated for each number of cores as the ratio of the execution time using one core to the execution time using a higher number of cores. We used a fixed number of iterations rather than an objective function convergence stopping criteria for the speedup calculation because the DDS algorithm is designed assuming a user-specified number of iterations (Tolson and Shoemaker, 2007). Figure 6 shows the speedup for Eno, Upper Neuse, and Neuse watershed models for 2 and 10 year simulation durations. The solid line in the figure represents the averaged cloud calibration times across 3 to 8 runs while the shaded region shows the maximum and minimum cloud calibration times.

The results show nearly linear scaling up to 64 cores and then a decrease from ideal speedup for core numbers above 64. This is due to an increase in the number of idle cores during initialization and finalization that become significant when the calibration procedure uses 64 or more cores (Humphrey et al., 2012). The results also suggest that the size of the watershed model and the simulation duration do not have a significant impact on speedup. The results show only a slight increase in the average speedup times for the longer duration model runs compared to the shorter duration model runs. This is likely due to the fact that the data exchanged between the head node and the compute nodes for the calibration runs are relatively small consisting of new parameter sets sent to the compute nodes and efficiency values sent back from the compute nodes (Humphrey et al., 2012). Therefore, speedup increases for longer duration model runs because model runtime is a more dominate term in the total calibration time compared to data exchange times.

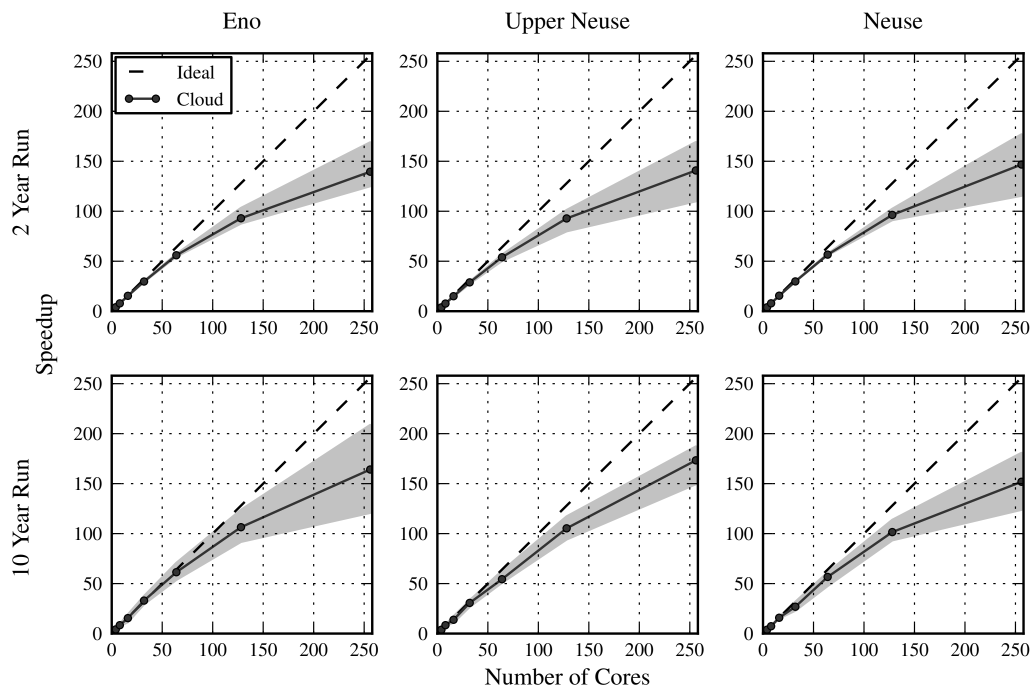

Figure 6: Speedup for different watershed sizes and time spans

5.4. Calibration Cost Analysis

For many users considering commercial cloud services, the decision whether to use a tool like the SWAT cloud calibration tool will be determined by cost. This tool was built using Microsoft's Azure cloud and current prices for renting VMs in Azure are \$0.09 per hour for a small VM (1.6GHz CPU, 1.75GB RAM), \$0.18 per hour for a medium VM (2 x 1.6GHz CPU, 3.5GB RAM), \$0.36 for a large VM (4 x 1.6GHz CPU, 7GB RAM), and \$0.72 for an extra large VM (8 x 1.6GHz CPU, 14GB RAM) (AzurePricing, 2014). Based on these current prices and the calibration test results, Table 2 shows the estimated costs for calibrating the different model scenarios. The estimates assume \$0.09 per core and that VMs can be rented by the minute rather than by the hour, which is the current billing model for Azure (Azure-Update, 2013). Given the costs associate with purchasing and maintaining multicore computers and clusters and the frequency with which a modelers is tasked with calibrating a SWAT model of a given size and duration, these cost estimates can help inform the modeler of a break-even-point where renting machines through a cloud service would be more cost effective than purchasing and maintaining local hardware.

Table 2: Cost of calibrating a SWAT model for a ten year model simulations for different watershed sizes

Number of	Eno Watershed		Upper Neuse Watershed		Neuse Watershed	
cores	Run Time (h)	Cost ($)	Run Time (h)	Cost ($)	Run Time (h)	Cost ($)
1	16.58	1.49	307.78	27.70	518.59	41.49
2	8.13	1.46	149.91	26.98	259.69	41.55
4	3.98	1.43	80.31	28.91	133.63	42.76
8	1.98	1.42	36.71	26.43	70.75	45.28
16	1.07	1.54	22.32	32.14	32.79	41.98
32	0.50	1.45	10.04	28.90	19.33	49.48
64	0.27	1.56	5.66	32.60	9.16	46.88
128	0.16	1.79	2.92	33.68	5.11	52.31
256	0.10	2.33	1.78	40.92	3.42	69.94

There are certain advantages to being able to calibrate a watershed model either overnight (i.e. about 12 hr) or during half of a workday (i.e. $<$ 4 hr). Therefore we have somewhat arbitrarily chosen 12 hours as an acceptable amount of time to complete a model calibration and 4 hours as a preferred amount to complete a model calibration. Using these two reference point, a 10 year SWAT calibration of the Eno watershed model would cost $1.46 to be performed in under 12 hr and $1.43 to be performed in under 4 hr. The Upper Neuse watershed model would cost $28.90 to be performed in under

12 hr and \$33.68 to be performed in under 4 hr. Finally, the Neuse watershed model would cost \$46.88 to chosen be performed in under 12 hr and \$69.94 to be performed in under 4 hr.

5.5. Estimating Calibration Time based on SWAT Model Properties

Assuming no speedup loss, the slope on Figure 5 should be -1 so that each additional core provides the same reduction in calibration time. Therefore it is possible to estimate the time to calibrate a SWAT model for a given watershed and simulation duration under the assumption of no speedup loss as

$$log(T) = (-1) * log(C) + \beta \tag{1}$$

where T is estimated cloud calibration time (hr), C is the number of cores, and β is a coefficient. This equation can be simplified to find T as function of C and the coefficient β (Equation 2).

$$T = C^{-1} * 10^{\beta} \tag{2}$$

The β coefficient in Equation 2 represents the y-intersect for each linear fit line to the data in Figure 5. These values were determined by fitting a power function to the calibration times using 1 and 2 cores, to reduce the impact of speedup loss for higher core numbers. This equation takes the form $y = ax^k$ where $k \approx -1$ (signifying minimal speedup loss) and $\beta = log(a)$. Estimated β values for each model scenario derived using this approach are given in Table 3.

Through further analysis of the data we found that β is dependent on two key properties of a SWAT model. These properties are the number of HRUs in the model (U) and the number of simulation time steps (N). We

Table 3: β coefficients for each calibration scenario

	10 Year Simulation			2 Year Simulation		
	Eno	U. Neuse	Neuse	Eno	U. Neuse	Neuse
β	1.22	2.49	2.71	0.49	1.72	1.98

fit a relationship between β and the log of the product of U and N (Figure 7). In our case, the models were executed on a daily time step interval, therefore with a 10 year simulation duration each model had 3,653 time steps (with three leap years in the simulation period) and with a 2 year simulation duration each model had 730 time steps. Given this, we can express β as a function of U and N as shown in Equation 3.

$$\beta = 1.05 * log(N * U) - 4.41 \tag{3}$$

Combining Equation 2 with Equation 3 gives Equation 4 that can be used to estimate the time required to calibrate a SWAT model assuming no speedup loss as a function of only two properties of that SWAT model.

$$T = 10^{-4.41} * C^{-1} * (N * U)^{1.05} \tag{4}$$

Equation 4 must be extended to account for the speedup loss observed in Figure 6. To account for this, we fit a second-order polynomial to the average of the speedup values for the six model scenarios.

$$S = -1.4 * 10^{-3} * C^2 + 0.97 * C \tag{5}$$

A correction factor to account for speedup losses (L) can then be defined as the ratio of the number of cores used (C) and the speedup loss (S).

$$L = C * (-1.4 * 10^{-3} * C^2 + 0.97 * C)^{-1} \tag{6}$$

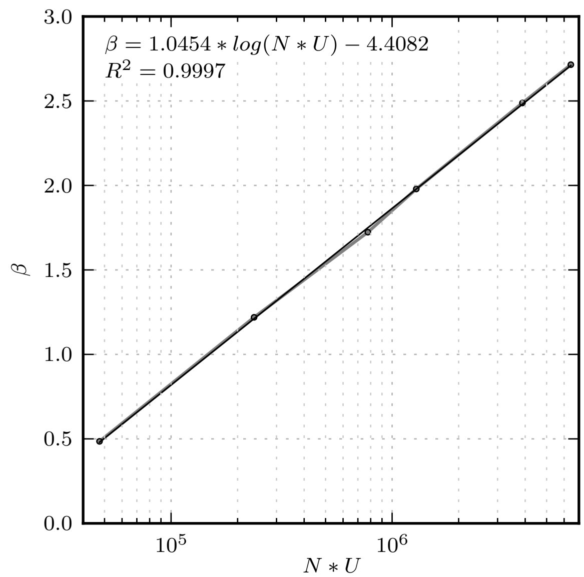

Figure 7: Relationship between intercept values β and model properties number of HRUs (U) and number of time steps (N)

Adding this factor to Equation 4 gives an equation that can be used to estimate calibration time for up to 256 cores taking into account speedup loss.

$$T = 10^{-4.41} * C^{-1} * (N * U)^{1.05} * L \tag{7}$$

Finally, as can be seen in Figure 6, there is significant variability in speedup loss for each tested scenario. It is possible to convey this variability using upper and lower limits for L. We did this by estimating S using second-order polynomials fit to averages of the lower and upper bounds for speedup loss shown in Figure 6.

$$L_{UB} = C * (-1.7 * 10^{-3} * C^2 + 0.92 * C)^{-1} \tag{8}$$

$$L_{LB} = C * (-1.2 * 10^{-3} * C^2 + 1.02 * C)^{-1} \tag{9}$$

These terms can be used as the L term in Equation 7 to estimate lower and upper bounds for calibration time (T) when accounting for observed variability in speedup loss.

We applied Equation 4 with estimates of L using the average case (Equation 6) and for the lower and upper bound cases (Equations 8 and 9, respectively) to compare predicted vs. observed calibration times (Figure 8). On average, using Equation 6 for L resulted in estimated calibration times for the model scenarios were within 4.1% of measured cloud calibration times (Figure 8). The worst case estimation was for the 10 year Upper Neuse watershed simulation on 256 cores that was 11.6% over estimated. However, this observed calibration time was bracketed by lower and upper bounds.

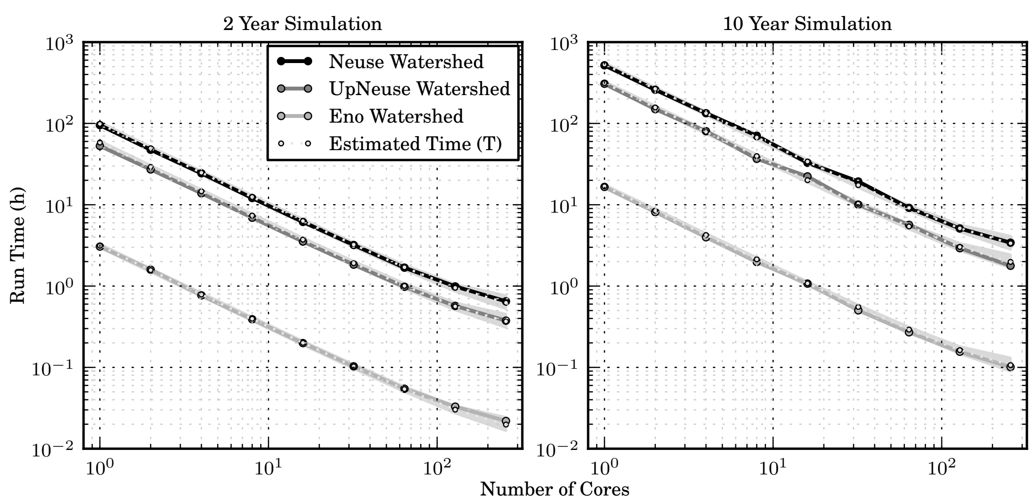

Figure 8: Estimated execution times using Equation 4 with various core numbers for Neuse, Upper Neuse, and Eno watersheds for 2 and 10 years simulation durations

6. Summary and Conclusions

We evaluated the convergence speed of a parallel DDS-based cloud calibration tool for SWAT described in Humphrey et al. (2012) for an increasing number of cores. The evaluation showed that the parallel DDS executions require a similar number of iterations (between 96 and 319 iterations, on average) for convergence for up to to 16 cores. For higher core numbers, additional iterations are needed to reach the same objective function value. The 32 and 62 core executions converged within 509 iterations for all tests. The 128 core executions took on average 698 iterations to coverage but did in some cases take over 1000 iterations to converge. The 256 core executions were within 98% of the convergence objective function value after 1000 runs, on average. Based on these results, 1000 iterations should still be sufficient to achieve convergence of an objective function for the parallel, cloud-based DDS tool for up to 256 cores. However, the results also suggest that the speedup times discussed in the paper would be different if a stopping criteria were used for calibration rather than a fixed number of iterations, given that executions using fewer cores (16 or less) converge with less iterations than executions that use a higher number of cores.

We quantified calibration time as a function of number of cores used for the SWAT cloud calibration tool across three different sized watersheds and two simulation durations. The results show that, for the large watershed (Neuse, 14,300 km^2) calibration with a 5 year warm-up period and a 5 year calibration period took 207 hours (8.6 days) on a personal computer. The cloud calibration tool completed the same calibration in 3.4 hours using 256 cores. Similarly, the small watershed (Eno, 171 km^2) and the medium watershed (Upper Neuse, 6,210 km^2) took 6 hours and 113 hours (4.7 days) to complete calibration on a personal computer, respectively. Using the cloud calibration tool with 256 cores, these two simulations were completed in 0.1 and 1.8 hours, respectively. While the 256 core results are presented here as the upper limit for our tests, we found based on a speedup analysis that 64 cores is the most cost efficient way to calibrate a SWAT model on the cloud because there was little speedup loss for each model scenario when using 64 cores.

We used the current Azure pricing model to estimate the cost of calibrating a watershed model. For the 256 core results presented earlier in this section, the small model calibration (Eno, 171 km^2), cost \$2.33, the medium watershed (Upper Neuse, 6,210 km^2) cost \$40.92, and the large watershed (Neuse, 14,300 km^2) cost \$69.94 to calibrate. These costs can be reduced by using fewer cores, but of course at the cost of increased wait time for the calibration to be completed. This information is meant to aid watershed modelers in selecting an optimal balance between cost and wait time for a particular application. Care should be taken to understand the limitations of the execution time and cost estimates, which may vary due to a number of factors including load on the cloud's compute and network resources, as well as specifics of the model not considered in this study (e.g., different numbers of parameters used in the calibration or selection of different process representations within the model).

Finally, we derived a relationship to estimate the calibration time for a SWAT model as a function of the number of HRUs and time steps used for the model, and a given number of cores used for the calibration. This relationship can be used to estimate calibration times using the cloud calibration tool to generally within 4% of observed cloud calibration time. We provide a method for estimating upper and lower bounds for calibration time estimates based on observed variability in speedup times. Applying this relationship for specific model applications provides a way for modelers to decide the number of cores needed to calibrate a SWAT model within a desired period of time. We caution, however, that the equations may not hold for scenarios outside of the range that we tested, for example SWAT models with more than 1,762 HRUs or simulation periods that extend beyond a 10 year duration.

Software Availability:

The SWAT cloud calibration software is available for use at the following URL: http://gale.cs.virginia.edu/SWAT/Web/portal.aspx.

References

Abbaspour, K., Johnson, C., van Genuchten, M., Nov. 2004. Estimating uncertain flow and transport parameters using a sequential uncertainty fitting procedure. Vadose Zone Journal 3 (4), 1340–1352, WOS:000227469200029.

Allen, R., 1986. A Penman for all seasons. Journal of Irrigation and Drainage Engineering-ASCE 112 (4), 348–368.

Allen, R., Jensen, M., Wright, J., Burman, R., 1989. Operational estimates of reference evapotranspiration. Agronomy Journal 81 (4), 650–662.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M., Apr. 2010.

A view of cloud computing. Communications of the ACM 53 (4), 50–58, WOS:000276841200025.

Arnold, J., Srinivasan, R., Muttiah, R., Williams, J., 1998. Large area hydrologic modeling and assessment part I: Model development. Journal of the American Water Resources Association 34 (1), 73–89.

Arnold, J. G., Allen, P. M., Bernhardt, G., 1993. A comprehensive surface-groundwater flow model. Journal of Hydrology 142 (1-4), 47–69.

AzureHPC, 2012. Microsoft Windows Azure HPC scheduler. http://msdn.microsoft.com/en-us/library/hh560247/, [Online; accessed 24-July-2014].

AzurePricing, 2014. Microsoft Windows Azure virtual machine pricing. https://www.windowsazure.com/en-us/pricing/calculator/, [Online; accessed 24-July-2014].

AzureUpdate, 2013. Microsoft will offer Azure by the minute to take on Amazons cloud. http://gigaom.com/2013/06/03/microsoft-will-offer-azure-by-the-minute-in-bid-to-take-on-amazons-cloud/, [Online; accessed 10-October-2013].

Duan, Q., Gupta, V., Sorooshian, S., Mar. 1993. Shuffled Complex Evolution approach for effective and efficient global minimization. Journal of Optimization Theory and Applications 76 (3), 501–521, WOS:A1993KW51600007.

Eckhardt, K., Arnold, J. G., 2001. Automatic calibration of a distributed catchment model. Journal of Hydrology 251 (1-2), 103–109.

Gassman, P., Reyes, M. R., Green, C., Arnold, J., 2007. The Soil and Water Assessment Tool: Historical development, applications, and future research directions. Transactions of the ASABE 50 (4), 1211–1250.

Graham, D., Butts, M., 2005. Flexible, integrated watershed modelling with MIKE SHE. Watershed models, 245–272.

Humphrey, M., Beekwilder, N., Goodall, J., Ercan, M., 2012. Calibration of watershed models using cloud computing. In: 8th IEEE International Conference on eScience 2012, Chicago Illinois. Oct 8-12 2012. IEEE.

Kenneth, M., 1972. Hydrology, National Engineering Handbook. In: Part 630, Chapter 15. United States Department of Agriculture. Natural Resources Conservation Service, Washington D.C.

Kirkby, M., Imeson, A., Bergkamp, G., Cammeraat, L., 1996. Scaling up processes and models from the field plot to the watershed and regional areas. Journal of Soil and Water Conservation 51 (5), 391–396.

Lee, M., Park, G., Park, M., Park, J., Lee, J., Kim, S., 2010. Evaluation of non-point source pollution reduction by applying Best Management Practices using a SWAT model and QuickBird high resolution satellite imagery. Journal of Environmental Sciences 22 (6), 826–833.

Liu, R., Zhang, P., Wang, X., Chen, Y., Shen, Z., 2013. Assessment of effects of best management practices on agricultural non-point source pollution in Xiangxi River watershed. Agricultural Water Management 117, 9–18.

Mell, P., Grance, T., 2011. The NIST definition of cloud computing. NIST special publication 800, 145.

Nash, J., Sutcliffe, J., 1970. River flow forecasting through conceptual models part I-A discussion of principles. Journal of Hydrology 10 (3), pp. 282–290.

Rouholahnejad, E., Abbaspour, K., Vejdani, M., Srinivasan, R., Schulin, R., Lehmann, A., May 2012. A parallelization framework for calibration of hydrological models. Environmental Modelling & Software 31 (0), 28–36.

Setegn, S. G., Dargahi, B., Srinivasan, R., Melesse, A. M., 2010. Modeling of sediment yield from anjeni-gauged watershed, Ethiopia using SWAT model1. JAWRA Journal of the American Water Resources Association 46 (3), 514–526.

Tarboton, D., Horsburgh, J., Maidment, D., Whiteaker, T., Zaslavsky, I., Piasecki, M., Goodall, J., Valentine, D., Whitenack, T., 2009. Development of a community hydrologic information system. In: 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, ed. RS Anderssen, RD Braddock and LTH Newham, Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation. pp. 988–994.

Tolson, B., Shoemaker, C., 2007. Dynamically Dimensioned Search algorithm for computationally efficient watershed model calibration. Water Resources Research 43 (1), W01413.

Tolson, B. A., Sharma, V., Swayne, D., 2007. A parallel implementation of the Dynamically Dimensioned Search (DDS) algorithm. In: International Symposium on Environmental Software Systems 2007, Prague. May 22-25 2007. ISESS.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M., 2008. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review 39 (1), 50–55.

Vrugt, J., Gupta, H., Bastidas, L., Bouten, W., Sorooshian, S., 2003. Effective and efficient algorithm for multiobjective optimization of hydrologic models. Water Resour. Res 39 (8), 1214.

Winchell, M., Srinivasan, R., Di Luzio, M., Arnold, J., 2008. ArcSWAT 2.1 interface for SWAT 2005: Users guide. Blackland Research Center, Temple, Texas.

Yapo, P., Gupta, H., Sorooshian, S., 1998. Multi-objective global optimization for hydrologic models. Journal of Hydrology 204 (1), 83–97.

Zhenyao, S., Lei, C., Tao, C., 2013. The influence of parameter distribution uncertainty on hydrological and sediment modeling: a case study of SWAT model applied to the Daning watershed of the Three Gorges Reservoir Region, China. Stochastic Environmental Research and Risk Assessment 27 (1), 235–251.