ISEIS

Journal of Environmental Informatics 37(2) 142-152 (2021)

Journal of Environmental Informatics

www.iseis.org/jei

Accelerating SWAT Simulations Using An In-Memory NoSQL Database

D. J. Zhang[1, 2], Q. Y. Lin[3 *], H. X. Yao[4], Y. R. He[1, 2], J. Deng[1, 2], and X. X. Zhang[1, 2]

[1] *College of Computer and Information Engineering, Xiamen University of Technology, Ligong Road 600, Xiamen, Fujian 361024, China*
[2] *Digital Fujian Institute of Big Data for Natural Hazards Monitor, Ligong Road 600, Xiamen, Fujian 361024, China*
[3] *Department of Resources and Environmental Sciences, Quanzhou Normal University, Donghai Street 398, Quanzhou, Fujian 362000, China*
[4] *Dorset Environmental Science Centre, Ontario Ministry of Environment, Conservation and Parks, 1026 Bellwood Road, Dorset, Ontario P0A 1E0, Canada*

Received 01 February 2018; revised 24 June 2018; accepted 30 June 2018; published online 20 December 2019

ABSTRACT. Due to its versatility, the Soil and Water Assessment Tool (SWAT) has been widely applied to investigate the effects of management activities and climate change on water availability and quality. However, the use of high spatial resolution data and the advantages of SWAT itself have significantly increased the input/output (I/O) demand and thus the runtime of modeling routines that require a large number of iterative simulations. In this study, we proposed a generic scheme to reduce the SWAT runtime by caching the model inputs using the in-memory NoSQL database Redis. Then the SWAT source codes (rev 488) was modified according to this proposed scheme to develop the MA-SWAT (memory accelerated SWAT) model by incorporating a new subroutine known as Fortran_calls_c to retrieve the cached inputs. We then evaluated MA-SWAT with four synthetic hydrological models and five different parallel schemes in a quad-core commodity laptop. The test results showed that when applied with a parallel simulation program, MA-SWAT could achieve a speedup by a factor of 8.4 ~ 10.9 depending on model complexity. Compared with the original SWAT, MA-SWAT significantly improved the computation speed, indicating that the proposed scheme is a desirable method for solving high computational demand problems such as calibration, sensitivity and uncertainty analysis. Moreover, the proposed concept of linking the SWAT model with Redis via the minimalistic C client driver of Redis is a generic method, and it is possible to apply this method to other Fortran-implemented environmental model to alleviate I/O demands.

Keywords: environmental models, generic method, MA-SWAT, Redis, speedup.

1. Introduction

Distributed hydrological and water quality models have played important roles in various fields, such as modeling water availability and quality, plant growth, evaluating effects of watershed management options, and land use and climate change impacts (Gassman et al., 2007; Chen et al., 2013; Boluwade and Madramootoo, 2015; Shen et al., 2015; Hu et al., 2018; Xiao et al., 2015). However, the increasing precision requests on spatial resolutions and the evolution of distributed hydrological models themselves have largely increased the computational time of these models, especially for large spatial and temporal scales. Moreover, due to a large number of immeasurable parameters (due in turn to measurement limitations and scaling issues) and various sources of uncertainty, these models require careful calibration and uncertainty analysis before they can be used to understand and investigate the underlying system. Unfortunately, these processes usually require a large number of model simulations and thus require a prohibitively long computational time. Therefore, the calibration and uncertainty analysis may become increasingly labor-intensive and time-consuming tasks for model practitioners and may hamper the ability of rapid and proper modelling (Humphrey et al., 2012; Rouholahnejad et al., 2012; Joseph and Guillaume, 2013). For example, as noted by Joseph and Guillaume (2013), the ability to use advanced approaches such as the Bayesian Markov chain Monte Carlo (MCMC) framework is hindered in computationally intensive models such as the Soil and Water Assessment Tool (SWAT).

Driven by study and policy needs, over the past few decades, environment modelers have devised many methods or technologies to reduce computation time in large-scale socio-environmental modeling (Bryan, 2013, Liu et al., 2013, Liu et al., 2014; Hu et al., 2015). Taking the SWAT model (i.e., a continuous, distributed-parameter, long-term and widely applied watershed model) as an example, many efforts to reduce computation time for SWAT or common modeling practices such as calibration, uncertainty and sensitivity analysis can be found in the literature. These endeavors are roughly divided into three categories. The first category (i.e., model reduction) uses a lightweight surrogate model such as Synthetic Neural Networks (ANNs) and Support Vector Machine (SVM) to approximate the computationally intensive SWAT. For example,

* Corresponding author. Tel.: +86 159 60753255; fax: +86 595 22796091.
E-mail address: qiaoyinglin@163.com (Q. Y. Lin).

ISSN: 1726-2135 print/1684-8799 online
© 2019 ISEIS All rights reserved. doi:10.3808/jei.201900425

Zhang et al. (2009) trained ANNs and an SVM to map the response of the objective function into the input parameter space of SWAT, and compared with the SWAT model, the two learning machines could deliver 20 to 50% time conservation for parameter calibration or uncertainty analysis. Recently, Sun et al. (2015) developed three meta-models that were trained using an existing SWAT model to support real-time decision-making with respect to activities related to surface water quality in a coastal watershed in Texas, USA. These studies reveal that surrogate-modelling generally increases the overall computational efficiency and model quality compared to when these methods are not applied. However, in some cases, when computational budget is not very limited, surrogate-modelling can be misleading and a hindrance. Besides, these methods can be referred to as 'black-box models', thus are subject to same shortcomings such as lack of flexibility and physical meaning, and thus not appropriate for any form of sensitivity analysis.

The second category applies parallel technologies to either simultaneously execute multiple model runs for model routines that involve larger numbers of simulations or parallelize the computation tasks within one model run. For example, Rouholahnejad et al. (2012) developed a parallel Sequential Uncertainty Fitting II (SUFI2) algorithm and greatly improved the efficiency of model calibration. Zhang et al. (2013) developed Python-based optimization software for SWAT that can improve the efficiency of SWAT optimization by approximately eight-fold on a Linux server with 16 cores. Wu and Liu (2012) developed a parallel parameter estimation program for SWAT using the R system. Joseph and Guillaume (2013) also proposed a parallel Differential Evolution Adaptive Metropolis (DREAM) algorithm in the R environment to reduce barriers to the use of the Markov chain Monte Carlo algorithms. However, the scalability of these tools or methods is generally poor due to the performance limit of a single machine.

Recently, the growth of high-performance computing systems and parallel programming techniques have sparked research on efficient solution of complex high-dimensional computational problems (and no exception to the SWAT modeling domain) in the forms of cluster, grid and cloud computing. For example, Whittaker (2004) and Confesor and Whittaker (2007) presented a parallel method by combining the cluster-based parallel computing technique with the non-dominated sorting genetic algorithm II on a Beowulf cluster consisting of a server and 12 computation nodes to facilitate the analysis of uncertainty of SWAT. Zhang et al. (2013) established a Python-based parallel computing package PP-SWAT by combining Python, MPI for Python and OpenMPI to parallelize A Multi-method Genetically Adaptive Multi-objective Optimization Algorithm (AMALGAM), for simultaneously addressing multiple objectives in calibration of SWAT. Test results on the Evergreen computing cluster showed that PP-SWAT could achieve a speedup of 45 ~ 109 times depending on model complexity. By leveraging the grid computing technologies and infrastructures, Gorgan et al. (2012) implemented the gSWAT application, a practical web solution for environmental specialists to use in calibration of extensive hydrological models and running of scenarios. With the interest in enabling large-scale environmental and hydrological models to execute and deliver results at near real time speeds, Yalew et al. (2013) developed generic tools and techniques by parallelizing the model structure on the Enabling Grids for E-science projects in Europe (EGEE). In recent years, cloud computing has been applied to fields with massive computing and data storage demands, and SWAT is no exception. For example, Humphrey et al. (2012) and Ercan et al. (2014) established a calibration system for SWAT based on Microsoft Windows Azure and the Dynamically Dimensioned Search method and obtained a significant speedup of SWAT calibration. Recently, Zhang et al. (2016) implemented a cloud-based Calibration and Uncertainty analysis Tool for SWAT (CUT-SWAT) using Hadoop as an open source cloud platform and the Generalized Likelihood Uncertainty Estimation method. Test results on a computer cluster consisting of ten virtual machines built on five commodity servers showed that CUT-SWAT could significantly accelerate the calibration and uncertainty analysis with a speedup range from 21.7 to 26.6 depending on model complexity. These studies take better advantage of computational power of modern computational facilities, in the form of parallel, grid, cluster and cloud computing, and therefore largely increase the computational efficiency. Nevertheless, increasing the processor or thread count beyond a certain threshold does not necessarily improve efficiency, because intensified resource competition may result in an I/O bottleneck.

The third category optimizes the model structure and/or its associated programs to reduce/eliminate barriers in application of the SWAT model. For example, Rouholahnejad et al. (2012) modified SWAT-edit.exe to allow it to cache a number of input files on the system's RAM and thus alleviated the I/O demands during model calibration and/or uncertainty analysis procedures. Yen et al. (2014) developed the Consolidated SWAT (C-SWAT) by consolidating 13 groups of SWAT input files from the sub-basin and Hydrologic Response Unit (HRU) levels into a single file for each category to enhance the computational speed of the SWAT model. Ki et al. (2015) identified the slowest routines in SWAT and later modified these routines using the Open Multiple Processing library to develop a new version of original SWAT (namely, iOMP-SWAT). The test results on an 8-core shared memory system showed a universal speedup ratio of 2.3. It is noted that the methods between the second and third categories do not need to be mutually exclusive, and in fact, they are usually applied together to solve high-dimension computing problems in modeling routines (e.g., Rouholahnejad et al. (2012), Yen et al. (2014) and Ki et al. (2015)). While these methods are effective solutions to reduce conflicts of the computational resources by alleviating the I/O demands, they usually require refactoring the original model. In addition, these methods usually dedicate to solve the intensive computational issues of certain specific model.

In this study, we introduced a generic scheme (belonging to the third category) to reduce the runtime of large-scale environmental models in general, and SWAT models in particular, by caching the model inputs with the in-memory NoSQL database Redis. To the best of our knowledge, this scheme is the first attempt to reduce the runtime of Fortran-based, I/O inten-

sive environmental models by using an in-memory NoSQL database. Specifically, the goal of this study is achieved by catching model input in a NoSQL database and revising the model input subroutines, such as the hruallo, readmgt, readhru and readchm, to read in input from Redis. This study distinct from and enhances previous studies in the following aspects: first, the proposed method provides an new way to improve model computational performance by using an in-memory NoSQL database; second, it can not only enhance the computing efficiency for an individual model run, but also support the acceleration of modeling routines with large number of model runs; third, it can be even useful where model simulations were parallel executed, as it can alleviate the conflict of I/O demands which usually becomes more serious when parallel executed; fourth, the proposed concept of linking the Fotran with Redis is a generic method and it may have a potential to apply this method to other Fortran-based environmental model with high I/O demands. As demonstrated with the SWAT model, we modified the SWAT source codes (rev 488) according to this proposed scheme to develop the MA-SWAT model. Specifically, modifications to the original SWAT included the following: a) incorporation of a new subroutine known as Fortran_calls_c to connect, disconnect and retrieve data to/from Redis server; b) initiation and release of a connection to/from Redis server at the beginning and ending respectively of the entry subroutine of the SWAT model; c) abandonment of the open and close operations of the HRU level files in subroutines such as hruallo, readmgt, readhru, readchm, etc.; d) replacement of the read operations in subroutines such as hruallo, readmgt, readhru, readchm, etc., which read in parameter values from external files to internal files. Finally, we evaluated MA-SWAT with four synthetic hydrological models and five different parallel schemes in a quad-core commodity laptop.

2. Materials and Methods

2.1. Soil and Water Assessment Tool (SWAT)

The Soil and Water Assessment Tool (Arnold and Fohrer 2005, Arnold et al. 1998) is a semi-distributed, continuous, watershed-scale hydrological model initially developed by the Agricultural Research Service of the United States Department of Agriculture (USDA-ARS). This tool was developed to predict the impact of watershed management practices on water, sediment, nutrient, pesticide, and fecal bacteria yields in the agricultural landscapes of North America. Due to its open-access, distributed and physically based nature, SWAT was adapted and applied to different landscapes and diverse land uses throughout the world. To date, over two thousand academic papers on SWAT have been found in the peer-reviewed literature (Center for Agricultural and Rural Development (CARD) 2015).

As a semi-distributed watershed model, SWAT first subdivides a watershed into sub-basins and further delineates Hydrologic Response Units (HRUs), which are unique combinations of soil, land cover and slope range representing the smallest unit of SWAT for each sub-basin. Each concept or logical component, including the basin, sub-basins, reaches, and HRUs, uses several text-based files to store their specific information. For example, files such as *.hru, *.mgt, *.sep, *.sol, *.chm, *.gw and *.ops are attributed to HRUs, and files such as *.pnd, *.sub, *.rte, *.swq, *.wgn and *.wus are attributed to the second level of units or the sub-basins. Although these separate text-based inputs offer merits such as ease of accessibility and high portability, they are not optimal for model simulations. For example, the HRU files (including *.hru, *.chm, *.mgt, *.sol, and *.gw), which are the largest proportion of the model inputs, are accessed twice during a model simulation (e.g., the *.hru files are read in subroutines hruallo and readhru in source files hruallo.f and readhru.f, respectively; see Figure 1). In addition, SWAT reads parameters from these files sequentially, which means that to obtain a certain parameter, the SWAT model might have to read in extra parameters locating in front of the desired parameter (see, e.g., lines 110 ~ 112 in subroutine hruallo in source file hruallo.f). No problems occur when performing small SWAT watershed model, but if executing a larger watershed model delineated at a finer resolution, the consecutive process of opening-reading-closing of the model input files consumes large amounts of time and computer resources.

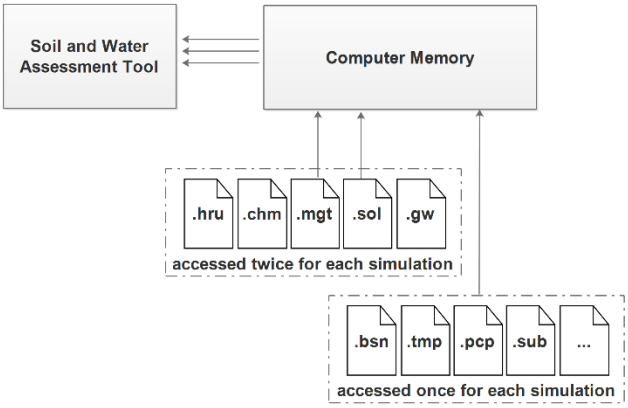

Figure 1. Process diagram of SWAT for reading in model inputs.

Moreover, common modeling routines such as model calibration, sensitivity and uncertainty analysis (simply referred to as calibration hereafter) usually require large numbers of iterative model simulations. Additionally, in each simulation, a process exists for updating the model parameters (shadowed components in Figure 2). This process can be I/O (input/output) intensive depending on the updated parameter type and parameterization strategy (i.e., percentage change, added, and replacement). For example, if the HRU-level parameters are incorporated into the calibration process, additional model input files must be edited, and if the percentage change and/or added approaches are used to update the model parameters, the parameter editor process must refer to a backup of the model files as well, thus increasing access to the hard disk.

2.2. Redis

Redis is one of the most popular in-memory NoSQL databases used as cache and message broker. It is built to provide the highest throughput (millions of operations/second) at

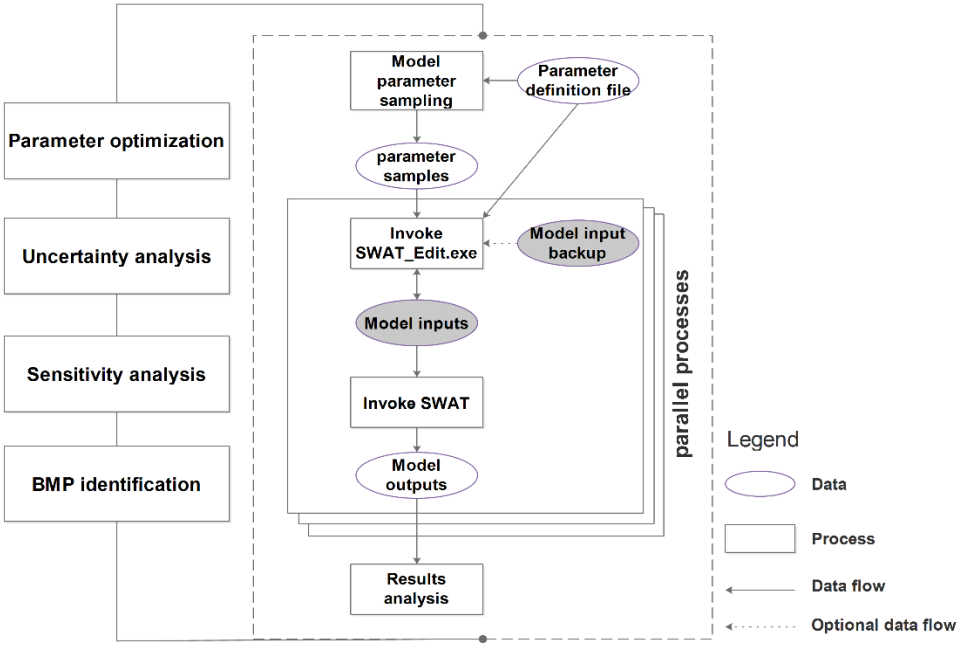

Figure 2. Common processes in parameter optimization, uncer-tainty, and sensitivity analysis and BMP identification for the SWAT model

the lowest latencies (< 1 ms), with the least system resources. As a result, this database can achieve much higher throughput than traditional solutions such as the relational database and document-oriented systems. Redis is notably fast, and thus it is perfectly suited for I/O intensive applications. Similar to relational databases, Redis uses the server-client system architecture. The server is a key-value storage component used to store, retrieve and manage associative arrays, a data structure commonly known as a dictionary or hash. Dictionaries contain a collection of records. These records are stored and retrieved using a key that uniquely identifies the record and is used to quickly find the data within the database. Since version 2.6, Redis includes server-side scripting with the Lua programming language, which acts like the role of stored procedures in relational database. This feature allows developers to write more advanced queries that are executed within Redis and are thus more effective. Redis supports several clients or drivers that are written in different program languages for different applications to connect and send commands to the server. In this study, Hiredis, a minimalistic C client library, is used to link SWAT with Redis.

2.3. Loading SWAT Inputs

In this study, to reduce disk I/O operations, the in-memory key-value store Redis is used to store SWAT inputs instead of the traditional document-oriented system. In the original SWAT, input files are read into the SWAT program in a line-by-line manner, and accordingly, SWAT inputs are stored in Redis on a line basis. In other words, each line of the SWAT input file is attempted as a record in Redis, which is identified by a unique combination of file name and line number (in fact, we used the colon character as a separator between portions of keys). Parameters that are involved in calibration, sensitivity or uncertainty analysis are attempted in a different manner. After the parameter sets are drawn by the sample algorithm, each independent parameter is loaded into Redis as a record and is defined by a unique combination of file name, line number and simulation number/sample number (colons were used to link adjacent portions of keys). For ease of references, the keys of regular parameters and those involved in calibration are denoted as S1 and S2, respectively.

The values of categories S1 are derived from the SWAT input files. However, the comment of each parameter is not included when loaded to the Redis server to minimize the utilized memory. The values of categories S2 are determined based on the parameter change strategies (i.e., percentage change, added, and replacement), which is method adopted by the iSWAT program (Abbaspour et al., 2007). For example, v_CN2.mgt = 72 means a global replacement of the original CN2 value in the *.mgt files with 72, and a_CN2.mgt = 1.82 causes a replacement of the CN2 value in the *.mgt files by a value of original CN2 values added by 1.82, etc. It is worth noting that the values of both categories must maintain the same format as the original SWAT file except that the comments for parameters are discarded. In other words, spaces around the value and length of value must be exactly that the same as in the original SWAT file.

2.4. Development of MA-SWAT

Redis supplies client drivers in 49 languages (e.g., ActionScript, Bash, C, C#, and C++) for client applications that communicate with the Redis server. However, drivers written in

Fortran are not currently available, which makes it impossible for SWAT (which was originally programmed in Fortran) to connect to the Redis server using a single programming language (i.e., Fortran). Fortunately, since Fortran 2003, an official standardized mechanism exists for interoperation with C. Therefore, it is possible to link SWAT with the Redis server via an intermediate component written in C. The schematic diagram illustrated in Figure 3 was proposed. In this scheme, a minimalistic C client driver for Redis known as Hiredis was incorporated into SWAT using the intrinsic module iso_c_binding and the bind attribute of the Fortran programming language. The iso_c_binding module contains named constants, derived types, and module procedures that are useful in mixed language situations. The bind attribute allows variables to interoperate with C and bridges the gap in syntax between Fortran and C identifiers. Details of the use of these two entities for interoperability are beyond the scope of this study. For detailed information on the calling mechanism between the Fortran and C programming languages, please consult a manual or reference book, such as the Fortran 2003 Handbook.

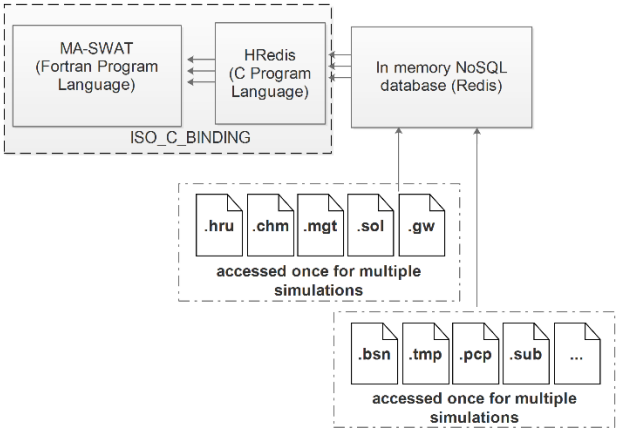

Figure 3. Sensitivity analysis on important parameters.

The most important modification to the original SWAT in development of MA-SWAT includes a new subroutine known as Fortran_calls_c (source codes are presented in Appendix A). In this subroutine, an interface that describes the C procedures was defined to map procedures in Fortran and C. The C procedures (source codes are presented in Appendix B) include connect_to_redis for creating a connection to the Redis server, disconnect for disconnecting from the Redis server, retrieve for acquiring the model input from the Redis server, and releasereply for releasing the reply object (global variable) that was created in the retrieve function. Two wrapper functions were also defined in this subroutine to prepare the input arguments and interpret the response of the C-implemented retrieve function. These wrapping functions eventually invoke the redisCommand function defined in Hiredis to send a command to and retrieve the response from the Redis server. In particular, in this function, an evalsha command is send to the Redis server, which in turn invokes a preloaded Lua script (see Appendix C). This Lua script accepts two arguments. The first argument is an identifier of a regular parameter, which is a string concatenation of file name, a colon and line number, and the second argument is a string consisting of a colon and simulation number. This script first checks whether the value of the combination of these two arguments (identifier for a certain simulation) exists, and if not, it returns the regular value of the desired parameter. This approach is beneficial because this script groups two commands into a custom-built cohesive function to reduce the communication overhead. In addition, using this script, MA-SWAT does not need to know whether a parameter is involved in the calibration processes. In other words, we do not need to write codes that treat parameters differently in MA-SWAT whether the parameters are involved in calibration processes.

Other modifications to the original SWAT include the following: (a) initiating and releasing a connection to/from the Redis server at the beginning and ending, respectively, of the entry subroutine (in the source file of main.f) of the SWAT model; (b) abandoning the open and close operations of the HRU level files in hruallo, readmgt, readhru, readchm, etc.; and (c) changing the read operations in subroutines such as hruallo, readmgt, readhru, readchm, etc., which read in parameter values from external files to internal files (i.e., in-memory character variables; code snippets are presented in Appendix D).

2.5. Development Tools and Data Sources

The MA-SWAT model was developed with Visual Studio 2013 and Intel(R) Visual Fortran Compiler XE 14. Other tools such as the program to load SWAT model files into Redis (loading program), program to create synthetic models (Synthetic model creator), and test programs of SWAT and MA-SWAT were implement in JAVA language (for the purpose of reusing the components that were formerly built in Java in Zhang et al. (2015) and Zhang et al. (2016)) with Eclipse. The synthetic models (see 3.1 for details of these models) and source codes of MA-SWAT and other tools are freely available at GitHub (Table 1).

Table 1. Tools and data used to implement and test MA-SWAT

Tools and models	Program Languages	GitHub repositories
MA-SWAT	C and Fortran	~/MASWAT.git
Redis Fortran driver	C and Fortran	~/redis-Fortran-driver.git
Loading program	Java	~/Loading-program.git
Synthetic model creator	Java	~/models-and-tools.git
Test program for MA-SWAT	Java	~/test-programs.git
Test program for SWAT	Java	~/test-programs.git
Synthetic models	NA	~/models-and-tools.git

*NA stands for "not applicable"; ~ is the root path of the repositories which is https://github.com/djzhang80.

3. Case Study

All other hydrological and nutrient components and interactions between components in MA-SWAT model were kept unchanged. When feeding with the same inputs, the generated

simulation results of MA-SWAT and SWAT should be exactly the same. A simple test to verify this assumption was conducted by comparing the outputs of MA-SWAT and SWAT with same model inputs. As expected, same results are generated.

Another assumption about this newly developed MA-SWAT is that MA-SWAT should be particularly efficient when used in parallel-implemented calibration, sensitivity or uncertainty analysis algorithms. To validate this assumption, we evaluated MA-SWAT and SWAT with 40 test scenarios, i.e., combinations of four synthetic hydrological models with five different parallel schemes in a commodity laptop, and compared the performance gains of these two models. The hydrological models, evaluated programs, parallel schemes are introduced as follows.

3.1. Hydrological Models

Four synthetic hydrological models (denoted as Models 1, 2, 3 and 4), representing different I/O burdens or model complexity, were created based on the hydrological model that is shipped with the SWAT-CUP program to evaluate how model complexity affects the efficiency of MA-SWAT and SWAT. These four synthetic hydrological models have the same sub-basin numbers, simulation periods and other configurations but different HRU numbers. Specifically, each of these models has 20 sub-basins and is set to run for three years. Each sub-basins consist of an equal number of HRUs within one hydrological model. The sub-basins in Models 1, 2, 3 and 4 consist of 5, 50, 100 and 150 HRUs, respectively.

Table 2. Parameters Involved for Testing the Performances of SWAT and MA-SWAT

Parameter	Changed method	File type	Layer	Adjustable range
CN2	a	.mgt	NA	-5 ~ 5
OV_N	r	.hru	NA	-0.1 ~ 0.1
CANMX	r	.hru	NA	-0.1 ~ 0.1
ESCO	v	.hru	NA	-0.01 ~ 1
EPCO	v	.hru	NA	-0.01 ~ 1
GW_DELAY	r	.gw	NA	-0.1 ~ 0.1
ALPHA_BF	r	.gw	NA	-0.1 ~ 0.1
GWQMN	r	.gw	NA	-0.1 ~ 0.1
GW_REVAP	r	.gw	NA	-0.1 ~ 0.1
REVAPMN	r	.gw	NA	-0.1 ~ 0.1
RCHRG_DP	r	.gw	NA	-0.1 ~ 0.1
SOL_Z	v	.sol	1	0 ~ 3500
SOL_AWC	v	.sol	1	0 ~ 1
SOL_K	v	.sol	1	0 ~ 2000

* Characters ‘a’, ‘r’ and ‘v’ denote added, percentage change, and replacement parameter changed methods, respectively, and ‘NA’ denotes that this attribute is not applicable to the associated parameter.

The number, type and change approach of the parameters used in model calibration, sensitivity and uncertainty analysis can affect how many model input files must be accessed and edited. In this case study, the parameters listed in Table 2 were selected as the calibrated parameters. All selected parameters are HRU-level parameters and thus cover most model input files. Additionally, for most of the parameters, we adopted the percentage change and added approaches, which double the accessed number of model input files in which these parameters lie. It is noted that the parameters and their value ranges were selected to test the efficiency of MA-SWAT and SWAT and not to conduct a meaningful calibration.

3.2. Test Programs

To evaluate and compare the performances of MA-SWAT and SWAT, two programs were implemented in Java (for the purpose of reusing the components that were formerly built in Java in Zhang et al. (2015) and Zhang et al. (2016)) to parallelize simulations of these two models. The schematic diagrams of these test programs are illustrated in Figures 2 and 4, respectively. As shown in these diagrams, these test programs are not fully functional calibration tools but rather a common subset that is involved in parallel calibration, sensitivity and uncertainty analysis algorithms, including parameter sampling, parameter editing, and model execution, etc.

Figure 2 shows the key procedures (within the dashed frame) of the test program used to parallelize simulations of the SWAT model (hereafter denoted as P1). First, the parameters are sampled using a Java parameter sample tool (JLHS) (Zhang et al., 2015) that implements the Latin hypercube sample (McKay et al., 2000) method. Some threads (the number of threads is determined by an argument that is passed to the program) are created, and in each of the threads, individual backups of the model inputs are edited according to the assigned parameter set via JSWAT-Edit (Zhang et al., 2015), and the SWAT model is subsequently executed. In addition to the threads for parallel model simulations, another thread (not presented in Figure 2) is created to gather performance data during the processes of the model simulations, including the CPU, memory, and time consumption. The test program (key processes of this program are presented in the dashed frame in Figure 4) for the MA-SWAT model (hereafter denoted as P2) is slightly different from that of the SWAT model. After the parameters are sampled, the model inputs and sampled parameters are loaded into Redis in the manner described in Section 2.3. The model input modification procedure is removed in all threads that perform model simulations because MA-SWAT can directly retrieve the model inputs from Redis that are specified for certain simulations.

3.3. Test Environment, Schemes and Measures

The performances of MA-SWAT and SWAT were tested on a laptop with a quad-core Intel Core CPU and 4-GB memory. The principal frequency of the CPU is 2.3 GHz. The operating system is 64-bit Windows 10. In general, the increase in the parallel-executed models causes a proportional increase in the I/O burdens of the model calibration procedures for traditional calibration tools. To evaluate how the performances are affected under different numbers of parallel execution models, test scenarios of the four aforementioned hydrological models

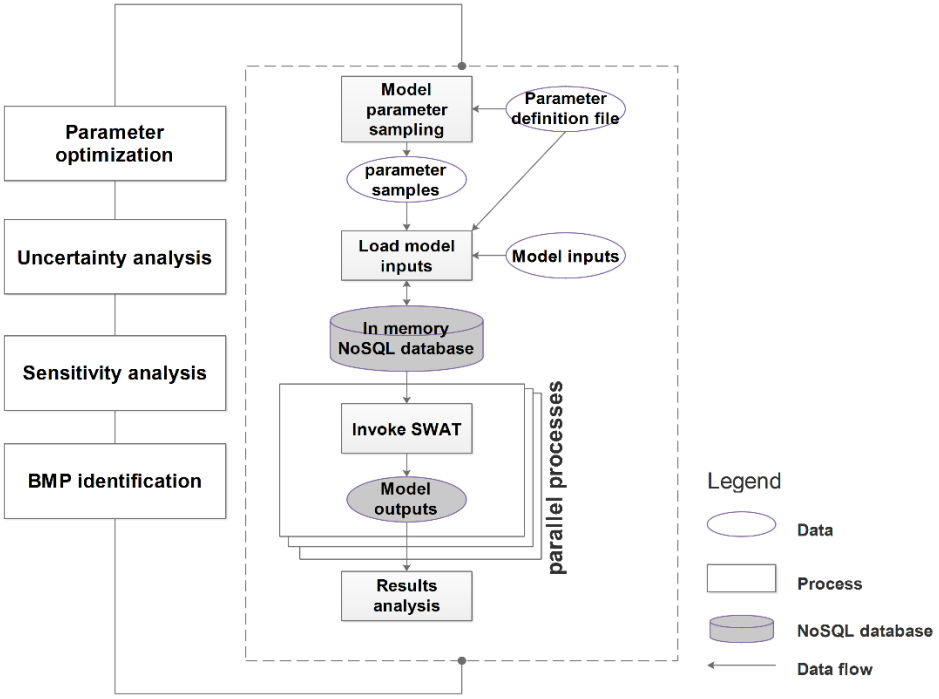

Figure 4. Common processes in parameter optimization, uncer-tainty, and sensitivity analysis and BMP identification for the MA-SWAT model.

combined with the different parallel schemes were assessed with the test programs P1 and P2. In other words, the four hydrological models were run with test programs P1 and P2 on 1 to 5 parallel threads (not including the one used to gather the performance information). Each of the parallel threads performed ten model simulations, and the average values for one simulation were reported in this study. Additional information on the test environment and schemes can be found in Table 3. Besides these scenarios, an additional scenario was added to evaluate the proposed modules that could be used to SWAT with large simulations. In this scenario, 3,000 simulations were parallel executed with the best speedup settings identified by aforementioned 40 scenarios (i.e., the best settings for MA-SWAT to speedup model simulations).

In addition to the average execution times for a single simulation and the CPU and RAM usage rates, we also used speedup (Pacheco, 2011), which indicates how much faster a program runs in parallel, to measure and compare the efficiency improvement achieved by MA-SWAT and SWAT when applied to parallelize simulations for modeling routines such as calibration, uncertainty analysis, etc. This measure is defined as follows:

$$Speedup_{p,m} = AT_{1,0} / AT_{p,m} \quad (1)$$

where the subscript p is the number of parallel threads, m is the symbol of model (o for SWAT, d for MA-SWAT), $AT_{1,o}$ is the average time taken by the test program P_1 to run with one thread, and $AT_{p,m}$ represents the average execution time for SWAT or MA-SWAT with p threads.

4. Results and Discussion

4.1. Performance Comparison and Analysis

The average time consumed by one simulation (AT) of the four synthetic hydrological models using MA-SWAT and SWAT is plotted against the number of parallel threads (not including the monitor thread used to collect performance information) in Figure 5a ~ d. While running with two or more parallel threads (hereafter referred to as parallel mode), the AT values were significantly reduced for both the MA-SWAT and SWAT. Overall, the AT of MA-SWAT was much smaller than that of SWAT, particularly when only one thread was used (hereafter referred to as sequential mode). For the MA-SWAT model, the AT decreased gradually up to four parallel threads and slightly increased at five parallel threads for all synthetic hydrological models. For the SWAT model, the AT changed following the trend of the MA-SWAT model for Models 1 and 2, but the reduction of AT ceased at three parallel threads for Models 3 and 4. The early suspension of the time reduction of the SWAT model (compared with that of MA-SWAT) for Model 3 and 4 is mostly due to I/O saturation. The large number of HRUs of Models 3 and 4 together with the relatively high number of parallel simulations causes a large number of simultaneous accesses of the hard disk when the simulations are parallelized.

Figure 5e ~ h shows the speedups achieved by parallelized simulations of MA-SWAT and SWAT. The speedups achieved by the MA-SWAT for Models 1, 2, 3 and 4 range from 2.4 to 3.4 when run in sequential mode. Better performances were achieved by further parallelizing the model simulations, espe-

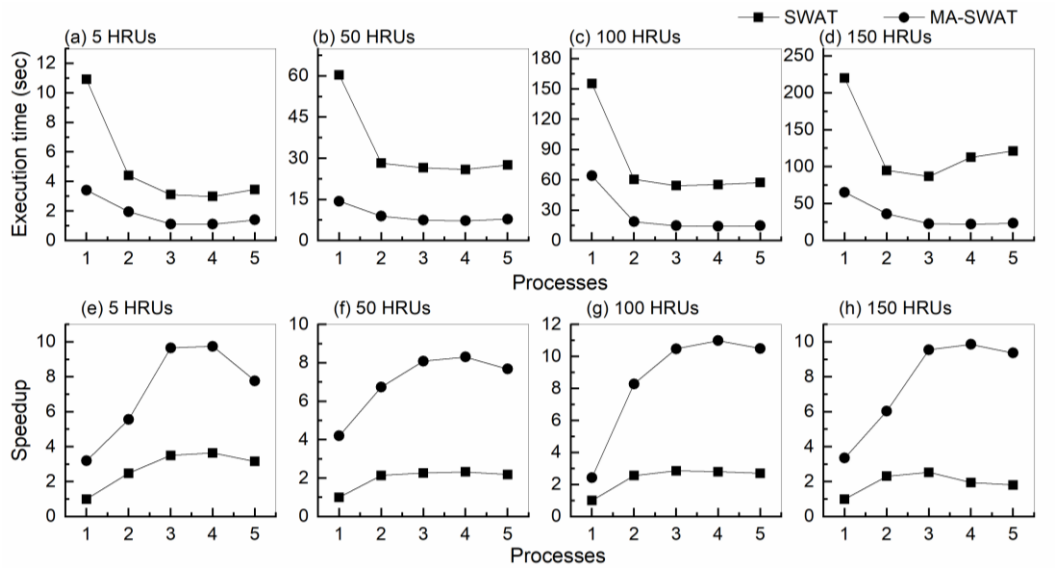

Figure 5. Average execution times versus number of parallel threads (a ~ d) and speedup versus number of parallel threads (e ~ h) for the four synthetic hydrological models.

Table 3. Schemes and environment for evaluating the performances of SWAT and MA-SWAT

Model	Test program	Parallel scheme	Hydrological model	Environment
SWAT	P1	1 ~ 5 synchronous model executions	Models 1 ~ 4	Hardware: CPU: Quad-core Intel Core 2.3 GHz RAM: 4 GB Software: OS: 64-bit Windows 10 Java Development Kit 7 Redis 2.6 (for MA-SWAT only)
MA-SWAT	P2	1 ~ 5 synchronous model executions	Models 1 ~ 4	

cially for MA-SWAT. For example, the best speedups achieved by MA-SWAT for Models 1 ~ 4 were 9.7, 8.4, 10.9 and 9.9, whereas the best speedups for these models archived by SWAT were 3.6, 2.4, 2.9 and 2.5, respectively. The additional performance gains for MA-SWAT compared with the original SWAT can be attributed to the ability to retrieve model inputs from the in-memory key-value database and elimination of the parameter editing procedure from the calibration processes. As observed in Figure 5g ~ h, the SWAT speedups stopped increasing at three threads (lower than the CPU cores). The phenomenon could be due to the disk limitation caused by simultaneous model inputs access. The best speedups achieved by SWAT for Models 1 and 2 and MA-SWAT for all four models ceased to increase at four parallel threads. One possible reason for the decrease in speedups beyond 4 threads is the limit of the available computational resources (i.e., the available cores of the CPU) because in these tests, the conflicting demands for disk I/O operations were relatively lower and thus the CPU (with 4 available physical cores) became the bottleneck of the system as the number of parallel threads exceeded the number of available CPU cores.

As mentioned previously, we also used a lightweight thread to collect the CPU and memory usage information while conducting these tests. The average memory usage is plotted against the number of parallel threads in Figure 6a ~ d. The memory usage rate increased with the increasing complexity of the tested hydrological models, and in most cases, the memory usage rate increased with the number of parallel threads for both MA-SWAT and SWAT. However, the memory usage rates of MA-SWAT were generally less than those of SWAT, which might be attributed to the absence of reading and editing large numbers of HRU level files in MA-SWAT while conducting these tests. Certain fluctuations were noted in the memory usage rates for both MA-SWAT and SWAT, which might be caused by the interferences of the monitor thread and other system processes or a relative larger sample interval of the monitor thread (and thus less precision). Figure 6e ~ h shows the average CPU usage rates of MA-SWAT and SWAT versus the number of parallel threads. As expected, the average CPU usage rates increased with the number of parallel threads for both MA-SWAT and SWAT. It is interesting to note that the average CPU usage rates of MA-SWAT gradually surpassed those of SWAT with the increasing complexity of the hydrological models. This phenomenon was mostly caused by the I/O overstress of SWAT, which made the CPU unable to run at full capacity, whereas MA-SWAT was less affected by the disk I/O and thus enabled the CPU to run more efficiently.

To further evaluate the acceleration support of modeling routines with large number of iterative runs, we added another test scenario. In this scenario, 3,000 simulations of Model 3 (with 20 sub-basins and 2,000 HRUs) were performed by MA-

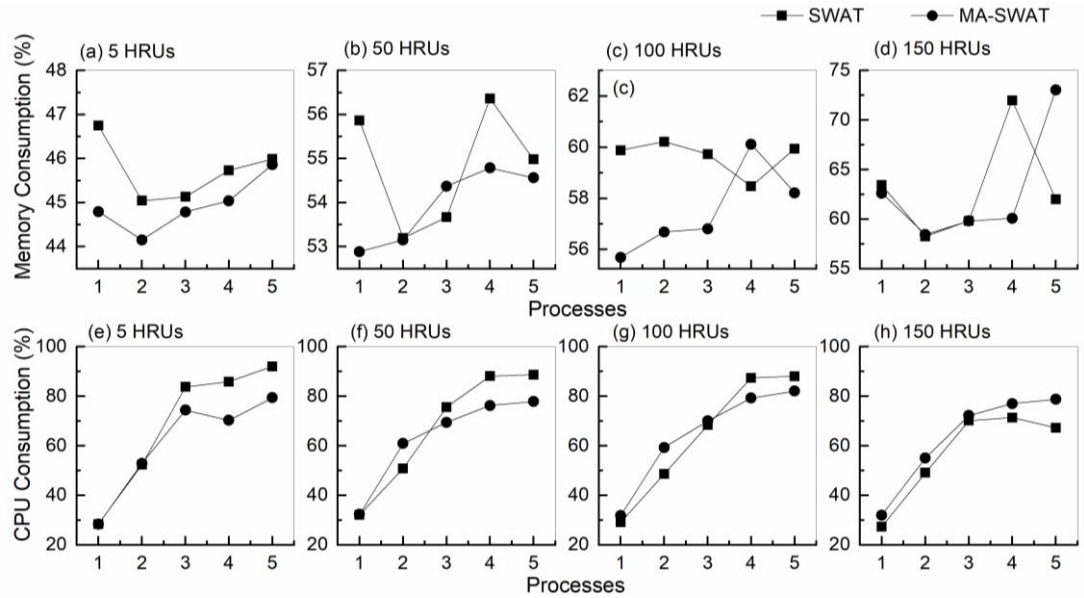

Figure 6. Memory consumption rates versus number of parallel threads (a ~ d) and CPU consumption rates versus number of parallel threads (e ~ h) for the four synthetic hydrological models.

SWAT and SWAT with four parallel threads (best speedup was achieved by MA-SWAT with these settings). The total execution time to perform such large number of simulations of the Model 3 could sum up to 3.1 days if sequentially executed. The test results of the scenario showed that the total execution times of MA-SWAT and SWAT were close to 7 and 26.6 hours, respectively. Results of this scenario indicate that MA-SWAT can not only enhance the computing efficiency for an individual model run, but also support the acceleration of modeling routines with large number of model runs. In addition, MA-SWAT can be even useful where model simulations were parallel executed, as MA-SWAT can alleviate the conflict of I/O demands.

4.2. Advantages and Disadvantages

MA-SWAT is the first attempt to reduce the runtime of a highly computational-intense SWAT model using an in-memory NoSQL database. Certain tools exist that share model inputs in memory during the SWAT model calibration. For example, Green and Griensven (2008) incorporated the Shuffled Complex Evolution (SCE) algorithm (Duan et al. 1992) into SWAT to build an auto-calibration procedure in which sequential runs of SWAT can share the same cached model inputs, and Rouholahnejad et al. (2012) modified the SWAT-edit.exe by caching a number of input files on the system's RAM, and thus all relative changes in the parameters can be made with respect to the static cached inputs. While sharing the same concept for reducing I/O demands of SWAT model by caching the model inputs in system RAM, MA-SWAT uses an external in-memory NoSQL database (Redis) to manage these model inputs, which differentiates it from these methods and thus offers certain advantages over other methods. First, MA-SWAT cached the model inputs in an external tool, which means the model inputs can be shared among parallel model simulations, i.e., iterative MA-SWAT simulations can be parallelized within one machine or in a distributed environment consists of many machines (although not tested in this study). Second, users do not have to consider issues such as whether sufficient system RAM exists to fit all model inputs because Redis can store the data to disk using a mechanism know as snapshotting.

Additionally, as demonstrated by the results of the MA-SWAT test program, MA-SWAT can be easily applied in other situations with large model simulation requirements, such as global sensitivity analysis and optimization of watershed management practices and can significantly reduce the computational times of these processes. Moreover, the proposed concept of linking the SWAT model with Redis via a minimalistic C client driver for Redis via the intrinsic module iso_c_binding and the bind attribute of the Fortran programming language is a generic method. Therefore, it is possible to apply this method to other Fortran-based environmental model with high I/O demands. The source codes of the newly developed subroutine Fortran_calls_c, which includes functions to connect, disconnect and retrieve data to/from Redis server, can be reused with minimal modifications (e.g., changing the IP address of the Redis server) when incorporating this method into other Fortran-based environmental models. To boost and ease the usage of this method, an open-access static library (Redis Fortran driver; Table 2) was implemented for Fortran-based environment models/applications to link with Redis. To work with Redis, programmers or users just need to include the *.lib and *.mod files to their projects, extend their codes to work with this library, and recompile their projects. The Redis Fortran driver is available at a GitHub (https://github.com) repository hosts at https://github.com/djzhang80/redis-fortran-driver.

The computational time of SWAT can be affected by many factors. These factors include the spatial-resolution, spatial-

scale, model building-up settings (i.e., the threshold values of sub-basin and HRU), temporal-resolution, length of simulation, and model version/structure as well as the execution environment. The first three are the major factors that determine the number of sub-basins and HRUs that will be generated when building up a hydrological model, and therefore determine the I/O demands of studied hydrological model. The following three factors usually determine the amount of computations and thus the execution model time. The model execution environment can also affect the required model run time, when the computational resources (such as CPU and RAM) are inadequate. Because MA-SWAT focuses on reducing the I/O demands of SWAT, it is efficient in solving high-intensive computational problems associated with the first three factors. Nevertheless, MA-SWAT cannot solve problems associated with the remained factors as it does not extend to other computation-intensive portions of the SWAT model. For example, increasing the simulation length can dilute the performance gains of MA-SWAT.

5. Conclusions and Future Work

In this study, we proposed a generic scheme designed to reduce the run times of environmental models implemented in Fortran by caching the model inputs with an in-memory key-value store, the popular NoSQL database Redis, and incorporating a new subroutine known as Fortran_calls_c to retrieve the cached inputs. Taking SWAT model as an example, we modified the original SWAT according to the proposed scheme to develop the MA-SWAT model. We believe that MA-SWAT is particularly efficient for use in parallel-implemented calibration, sensitivity or uncertainty analysis tools. Therefore, we evaluated MA-SWAT with four synthetic hydrological models and five different parallel schemes in a commodity laptop. The results revealed that MA-SWAT can significantly improve the performance compared with the original SWAT if running at the same configurations, and proved that the proposed scheme is a desirable method for solving high computational demand problems in hydrological model calibration, sensitivity and uncertainty analysis.

As a proof-of-concept prototype tool, MA-SWAT is currently under restructuring to cache the HRU-level files with an in-memory NoSQL database. Future possible studies include the following: 1) continued improvement of the MA-SWAT performance by caching all model inputs to Redis, 2) parallelization of the MA-SWAT simulations in a distributed environment, and 3) application of the proposed scheme to other environmental models such as the Hydrologic Simulation Program-Fortran model to validate its flexibility and universality.

Acknowledgments. This work was financially supported by the Natural Science Foundation of Fujian Province [grant number 2015J01176 and 2018J01481]; the Science and Technology Project of Quanzhou Municipality [grant number 2015Z136]; the Advance Research Program of Quanzhou Normal University for National Science Foundation [grant number 2016YYKJ07]; and the Talent Project of Xiamen University of Technology [grant number YKJ16017R].

References

Abbaspour, K.C., Yang, J., Maximov, I., Siber, R., Bogner, K., Mieleitner, J., Zobrist, J., and Srinivasan, R. (2007). Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT. *J. Hydrol.*, 333(2-4), 413-430. https://doi.org/10.1016/j.jhydrol.2006.09.014

Arnold, J.G. and Fohrer, N. (2005). SWAT2000: current capabilities and research opportunities in applied watershed modelling. *Hydrol. Process.*, 19(3), 563-572. https://doi.org/10.1002/hyp.5611

Arnold, J.G., Srinivasan, R., Muttiah, R.S., and Williams, J.R. (1998). Large area hydrologic modeling and assessment part I: Model development. *J. Am. Water Resour. Assoc.*, 34(1), 73-89. https://doi.org/10.1111/j.1752-1688.1998.tb05961.x

Boluwade, A. and Madramootoo, C. (2015). Determining the influence of land use change and soil heterogeneities on discharge, sediment and phosphorus. *J. Environ. Inf.*, 25(2), 126-135. https://doi.org/10.3808/jei.201500290

Bryan, B.A. (2013). High-performance computing tools for the integrated assessment and modelling of social-ecological systems. *Environ. Model. Software*, 39, 295-303. https://doi.org/10.1016/j.envsoft.2012.02.006

Center for Agricultural and Rural Development (CARD). SWAT Literature Database for Peer-Reviewed Journal Articles. https://www.card.iastate.edu/swat_articles/ (accessed Jan 18, 2018)

Chen, Y., Cheng, S.Y., Liu, L., Guo, X.R., Zhang, Z., Qin, C.H., Hao, R.X., Lu, J., and Gao, J.J. (2013). Assessing the effects of land use changes on non-point source pollution reduction for the Three Gorges Watershed using the SWAT model. *J. Environ. Inf.*, 22(1), 13-26. https://doi.org/10.3808/jei.201300242

Confesor, R.B. and Whittaker, G.W. (2007). Automatic calibration of hydrologic models with multi-objective evolutionary algorithm and pareto optimization. *J. Am. Water Resour. Assoc.*, 43(4), 981-989. https://doi.org/10.1111/j.1752-1688.2007.00080.x

Duan, Q., Sorooshian, S., and Gupta, V. (1992). Effective and efficient global optimization for conceptual rainfall-runoff models. *Water Resour. Res.*, 28(4), 1015-1031. https://doi.org/10.1029/91WR029 85

Ercan, M.B., Goodall, J.L., Castronova, A.M., Humphrey, M., and Beekwilder, N. (2014). Calibration of SWAT models using the cloud. *Environ. Model. Software.* 62, 188-196. https://doi.org/10.1016/j.envsoft.2014.09.002

Gassman, P.W., Reyes, M.R., Green, C.H., and Arnold, J.G. (2007). The soil and water assessment tool: Historical development, applications, and future research directions. *Trans. ASABE*, 50(4), 1211-1250. https://doi.org/10.13031/2013.23637

Gorgan, D., Bacu, V., Mihon, D., Rodila, D., Abbaspour, K., and Rouholahnejad, E. (2012). Grid based calibration of SWAT hydrological models. *Nat. Hazards Earth Syst. Sci.*, 12(7), 2411-2423. https://doi.org/10.5194/nhess-12-2411-2012

Green, C.H. and Griensven, A.V. (2008). Autocalibration in hydrologic modeling: Using SWAT2005 in small-scale watersheds. *Environ. Model. Software*, 23(4), 422-434. https://doi.org/10.1016/j.envsoft.2007.06.002

Hu, J., Sun, L., Li, C.H., Wang, X., Jia, X.L., and Cai, Y.P. (2018) Water quality risk assessment for the Laoguanhe River of China using a stochastic simulation method. *J. Environ. Inf.*, 31(2), 123-136. https://doi.org/10.3808/jei.201800387

Hu, Y., Garcia-Cabrejo, O., Cai, X., Valocchi, A.J., and DuPont, B. (2015). Global sensitivity analysis for large-scale socio-hydrological models using Hadoop. *Environ. Model. Software*, 73, 231-243. https://doi.org/10.1016/j.envsoft.2015.08.015

Joseph, J.F. and Guillaume, J.H.A. (2013). Using a parallelized MCMC algorithm in R to identify appropriate likelihood functions for SWAT. *Environ. Model. Software*, 46, 292-298. https://doi.org/10.1016/j.envsoft.2013.03.012

Ki, S.J., Sugimura, T., and Kim, A.S. (2015). OpenMP-accelerated

SWAT simulation using Intel C and Fortran compilers: Development and benchmark. *Comput. Geosci.*, 75, 66-72. https://doi.org/10.1016/j.cageo.2014.10.017

Liu, J., Zhu, A.X., Liu, Y., Zhu, T., and Qin, C.Z. (2014). A layered approach to parallel computing for spatially distributed hydrological modeling. *Environ. Model. Software*, 51, 221-227. https://doi.org/10.1016/j.envsoft.2013.10.005

Liu, J., Zhu, A.X., and Qin, C.Z. (2013). Estimation of theoretical maximum speedup ratio for parallel computing of grid-based distributed hydrological models. *Comput. Geosci.*, 60, 58-62. https://doi.org/10.1016/j.cageo.2013.04.030

McKay, M.D., Beckman, R.J., and Conover, W.J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1), 55-61. https://doi.org/10.1080/00401706.2000.10485979

Rouholahnejad, E., Abbaspour, K.C., Vejdani, M., Srinivasan, R., Schulin, R., and Lehmann, A. (2012). A parallelization framework for calibration of hydrological models. *Environ. Model. Software*, 31, 28-36. https://doi.org/10.1016/j.envsoft.2011.12.001

Shen, Z.Y., Chen, L., and Liao, Q. (2015). Effect of rainfall measurement errors on nonpoint-source pollution model uncertainty. *J. Environ. Inf.*, 26(1), 14-26. https://doi.org/10.3808/jei.201400271

Sun, A.Y., Miranda, R.M., and Xu, X. (2015). Development of multimetamodels to support surface water quality management and decision making. *Environ. Earth Sci.*, 73(1), 423-434. https://doi.org/10.1007/s12665-014-3448-6

Whittaker, G. (2004). Use of a Beowulf cluster for estimation of risk using SWAT. *Agron. J.*, 96(5), 1495-1497. https://doi.org/10.2134/agronj2004.1495

Wu, Y. and Liu, S. (2012). Automating calibration, sensitivity and uncertainty analysis of complex models using the R package Flexible Modeling Environment (FME): SWAT as an example. *Environ. Model. Software*, 31, 99-109. https://doi.org/10.1016/j.envsoft.2011.11.013

Xiao, W.H., Wang, J.H., Huang, Y.H., Sun, S.C., and Zhou, Y.Y. (2015) An approach for estimating the nitrobenzene (NB) emission effect in frozen rivers: a case study of nitrobenzene pollution in the Songhua River, China. *J. Environ. Inf.*, 26(2), 140-147.

Yalew, S., van Griensven, A., Ray, N., Kokoszkiewicz, L., and Betrie, G.D. (2013). Distributed computation of large-scale SWAT models on the Grid. *Environ. Model. Software*, 41, 223-230. https://doi.org/10.1016/j.envsoft.2012.08.002

Yen, H., Ahmadi, M., White, M.J., Wang, X., and Arnold, J.G. (2014). C-SWAT: The Soil and Water Assessment Tool with consolidated input files in alleviating computational burden of recursive simulations. *Comput. Geosci.*, 72, 221-232. https://doi.org/10.1016/j.cageo.2014.07.017

Zhang, D., Chen, X., Yao, H., and James, A. (2016). Moving SWAT model calibration and uncertainty analysis to an enterprise Hadoop-based cloud. *Environ. Model. Software*, 84, 140-148. https://doi.org/10.1016/j.envsoft.2016.06.024

Zhang, D., Chen, X., Yao, H., and Lin, B. (2015). Improved calibration scheme of SWAT by separating wet and dry seasons. *Ecol. Model.*, 301, 54-61. https://doi.org/10.1016/j.ecolmodel.2015.01.018

Zhang, X., Beeson, P., Link, R., Manowitz, D., Izaurralde, R.C., Sadeghi, A., Thomson, A.M., Sahajpal, R., Srinivasan, R., and Arnold, J.G. (2013). Efficient multi-objective calibration of a computationally intensive hydrologic model with parallel computing software in Python. *Environ. Model. Software*, 46, 208-218. https://doi.org/10.1016/j.envsoft.2013.03.013

Zhang, X., Srinivasan, R., and Liew, M.V. (2009). Approximating SWAT model using artificial neural network and support vector machine. *J. Am. Water Resour. Assoc.*, 45(2), 460-474. https://doi.org/10.1111/j.1752-1688.2009.00302.x.