Environmental Modelling & Software 31 (2012) 28–36

Contents lists available at SciVerse ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

A parallelization framework for calibration of hydrological models

E. Rouholahnejad[a,*], K.C. Abbaspour[a], M. Vejdani[b], R. Srinivasan[c], R. Schulin[d], A. Lehmann[e]

[a] *Eawag, Swiss Federal Institute of Aquatic Science and Technology, Ueberlandstrasse 133, CH-8600 Duebendorf, Switzerland*
[b] *Neprash Technology, 1625 Sundew Pl, Coquitlam, B.C., V3E 2Y4, Canada*
[c] *Spatial Sciences Laboratory, Texas A&M University, Texas Agricultural, Experimental Station, College Station, TX, USA*
[d] *ETH Zürich Institute of Terrestrial Ecosystem, Universitätstr. 16, 8092 Zürich, Switzerland*
[e] *University of Geneva, Climatic Change and Climate Impacts, 7 Route de Drize, CH-1227 Carouge, Switzerland*

ARTICLE INFO

Article history:
Received 1 April 2011
Received in revised form
21 November 2011
Accepted 2 December 2011
Available online 5 January 2012

Keywords:
Parallel processing
SWAT-CUP
SUFI2
Hydrologic models

ABSTRACT

Large-scale hydrologic models are being used more and more in watershed management and decision making. Sometimes rapid modeling and analysis is needed to deal with emergency environmental disasters. However, time is often a major impediment in the calibration and application of these models. To overcome this, most projects are run with fewer simulations, resulting in less-than-optimum solutions. In recent years, running time-consuming projects on gridded networks or clouds in Linux systems has become more and more prevalent. But this technology, aside from being tedious to use, has not yet become fully available for common usage in research, teaching, and small to medium-size applications. In this paper we explain a methodology where a parallel processing scheme is constructed to work in the Windows platform. We have parallelized the calibration of the SWAT (Soil and Water Assessment Tool) hydrological model, where one could submit many simultaneous jobs taking advantage of the capabilities of modern PC and laptops. This offers a powerful alternative to the use of grid or cloud computing. Parallel processing is implemented in SWAT-CUP (SWAT Calibration and Uncertainty Procedures) using the optimization program SUFI2 (Sequential Uncertainty FItting ver. 2). We tested the program with large, medium, and small-size hydrologic models on several computer systems, including PCs, laptops, and servers with up to 24 CPUs. The performance was judged by calculating speedup, efficiency, and CPU usage. In each case, the parallelized version performed much faster than the non-parallelized version, resulting in substantial time saving in model calibration.

Software availability

Software:
SWAT-CUP
Developer:
Karim C. Abbaspour
Eawag, Ueberlandstr 133, 8600 Duebendorf, Switzerland
abbaspour@eawag.ch
Operating system requirements:
Windows XP or higher
Linux version of the SWAT-CUP package is also available without the interface
Language:
C++ and C#
Size:
125 MB zipped version

Availability:
- SWAT-CUP with 5 different optimization routines is available free of charge and can be downloaded from: http://www.eawag.ch/forschung/siam/software/swat/index
- The parallel processing module with SUFI2 in the SWAT-CUP is licensed. SWAT-CUP allows free use of the parallel processing scheme with a limit of 20 simulations

1. Introduction

Advances in GIS technology and interfacing of hydrological programs with advanced GIS systems allow the construction of high resolution large-scale models. However, the execution time of these models can be rather long, not allowing proper model calibration and uncertainty analysis. For this reason, in the last few years, the use of distributed computing in the form of grid and cloud computing has become increasingly prevalent. Distributed hydrologic models are especially difficult to calibrate because of

* Corresponding author. Eawag, Ueberlandstrasse 133, CH-8600 Duebendorf, Switzerland. Tel.: +41 58 765 5012; fax: +41 58 765 5375.
E-mail address: elham.rouholahnejad@eawag.ch (E. Rouholahnejad).

1364-8152/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2011.12.001

reasons such as time constraints, difficulties in parameterization, non-uniqueness (having more than one acceptable solution), uncertainties in the conceptual model, and model inputs. In this paper we address the problem of computation time and describe a system where an optimization algorithm, SUFI2, is used in a parallel processing scheme to run on Windows-based computers. The parallel processing scheme developed here utilizes the existing capabilities of the available systems and is ideal for performing hydrologic model calibration and uncertainty analysis.

Distributed computing is an extension of the object-oriented programming concept of abstraction (Sundaram, 2010). Abstraction removes the complex working details from visibility. All that is visible is an interface which receives inputs and provides outputs. How these outputs are computed is completely hidden. Grid and cloud computing and parallel processing belong to the family of distributed computing. Currently, two different ways are pursuit in parallelizing hydrological models. One is parallelization of the code itself to be run as parallel threads (Neal et al., 2010; Li et al., 2011), and another is to submit model runs with different parameters to different CPUs (or GPUs) on the same or different computers (Lecca et al., 2011; Singh et al., 2011; Kalyanapu et al., 2011). Jobs submitted to a network of computers are run on grid and cloud. Below is a brief definition of these systems and their advantages and disadvantages.

A *grid network* is a computer network consisting of a number of computers connected in a grid topology. Some advantages of grid computing are: there is no need to buy expensive symmetric multiprocessing (SMP) servers for applications that can be split up and distributed to less powerful servers; much more efficient use of idle resources is made; grid environments are much more modular and don't have single points of failure; and jobs can be executed in parallel, speeding up the performance (Fernandez-Quiruelas et al., 2011; Vassilios, 2008).

Some disadvantages of the grid are: memory intensive applications that can't take advantage of message passing interface (MPI) may not be able to take advantage of the grid system; a fast connection between computer resources (gigabit Ethernet at a minimum) is needed in intense MPI applications; some applications may need to be tweaked to take full advantage of a new model; licensing across many servers may make it prohibitively expensive for some applications; and political challenges associated with sharing resources across different administration domains may also prohibit the use of grids for many users (Taylor et al., 2004).

Cloud computing is Internet-based computing whereby shared resources, software, and information are provided to computers and other devices on demand, like the electricity grid. Similar to grid computing, some of the advantages offered by cloud computing are: lower computer costs, improved performance with less use of computer memory, virtually unlimited storage capacity, increased data reliability, universal document access, and easier group collaboration. There are also a number of disadvantages associated with cloud computing, which include: requiring a constant high-speed Internet connection, and not getting an instantaneous connection if the cloud servers are busy. It is the opinion of some experts that security may be a disadvantage of the cloud (Miller, 2009). Theoretically, data stored in the cloud should be safe because it is replicated across multiple machines. But in the case of missing data, there is no physical or local backup (unless data is methodically downloaded).

Parallel processing or *parallel computing* is the ability to carry out multiple operations or tasks simultaneously. In parallel processing more than one CPU or processor core is used to execute a program or multiple computational threads. Parallel processing makes programs run faster because there are more CPUs or cores running it. In practice, it is often difficult to divide a program in such a way that separate CPUs can execute different portions without interfering with each other. Older computers have just one CPU, but newer computers have multi-core processor chips and many CPUs. With single-CPU, or single-core computers, it is also possible to perform parallel processing by connecting the computers in a network. However, this type of parallel processing requires very sophisticated distributed processing software. The main advantages of the system we have developed is that: it does not have the disadvantages of the grid and cloud; but it has all the advantages of using a PC. These include: the user being in full control of the job being processed; the job can be stopped and restarted at any time; a PC is much simpler to use, not needing grid or cloud certificate and permission. For very large models, a powerful computer (e.g., 24–48 CPUs with >16 GB RAM) is, however, needed to take full advantage of parallel processing. With the advancement of new technologies, this is now available at a reasonable cost.

In the current paper we describe and test a parallel processing software that allows calibration of the hydrologic simulator Soil Water Assessment Tool (SWAT) (Arnold et al., 1998), which is linked to five different optimization schemes in the SWAT-CUP software package (Fig. 1). SWAT-CUP is a standalone program that links to SWAT's output text files set. Theoretically, any other model or optimization algorithm could be added to this package. Currently, however, the program SWAT is linked with the Generalized Likelihood Uncertainty Estimation (GLUE) (Beven and Binley, 1992), Sequential Uncertainty Fitting (SUFI2) (Abbaspour et al., 2004, 2007), Parameter Solution (ParaSol) (Van Griensven and Meixner, 2006), Markov chain Monte Carlo (MCMC) (e.g., Kuczera and Parent, 1998; Marshall et al., 2004; Vrugt et al., 2003), and Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995). The parallel processing currently only supports SUFI2, which lends itself easily to parallelization as the program runs with independent parameter sets. Support for PSO is under development.

The objective of this paper is to describe the SUFI2 parallel processing algorithm as implemented in the SWAT-CUP. We compare the computation time of parallel SUFI2 by applying it to a small, a medium, and a large size SWAT project using different computer systems. This study is designed to investigate parallel processing issues not to perform a meaningful calibration task. Finally, we summarize the performance of the parallelization and the gains in time obtained by parallel processing.

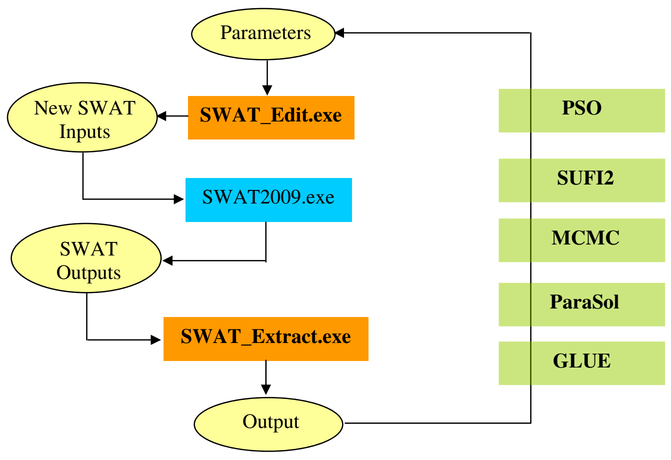

Fig. 1. Structure of the SWAT-CUP and its linkage to five optimization algorithms.

2. Materials and methods

2.1. The hydrologic simulator (SWAT)

SWAT is a process-based, river basin-scale, semi-distributed hydrologic model. The program has been developed to predict the impact of land management on water, sediment, and agricultural chemical yield in large, complex watersheds. SWAT is an integrated model including components such as weather, hydrology, soil, nutrients, pesticides, land management, bacteria and pathogens. The model is a computationally efficient simulator of hydrology and water quality at various scales which has been used in many international applications (Arnold and Allen, 1996; Abbaspour et al., 2007; Yang et al., 2007; Schuol et al., 2008a, 2008b). It includes procedures to describe how CO_2 concentration, precipitation, temperature, and humidity affect plant growth. It also simulates evapotranspiration, snow and runoff generation, and is used to investigate climate change impacts [Abbaspour et al., 2009; Eckhardt and Ulbrich, 2003; Fontaine et al., 2001].

SWAT is a continuous simulation model which operates on a daily time step. In SWAT the spatial heterogeneity of the watershed is taken into account, considering information from the Digitized Elevation Model, soils, and landuse GIS data. Spatial parameterization of the SWAT model is performed by dividing the watershed into sub-basins based on topography. These are further subdivided into hydrologic response units (HRU) based on soil and landuse characteristics. These data and related parameters are stored in text files. A high resolution project could easily result in thousands of input files, which would make it challenging to reconfigure or update model parameters. All SWAT's text input and output files reside in the TxtInOut directory. Initially, SWAT-CUP copies these files to a BACKUP directory, which remain unchanged throughout the calibration process.

2.2. SUFI2 optimization program

In Fig. 2 a schematic diagram of the coupling between SUFI2 and SWAT is illustrated. Initially, a Latin hypercube (McKay et al., 1979) procedure draws samples from the spaces defined by user-supplied parameter ranges. This is the pre-processing stage executed by a batch file *SUFI2_pre.bat*, which runs the program *SUFI2_LH_sample.exe*. The parameter sets thus sampled are independent and for this reason parallel runs could be executed. Theoretically, all samples could be run at once, hence an entire iteration would require only the time that it takes to make one model run.

After pre-processing, another batch file, *SUFI2_run.bat*, executes the *SWAT_Edit.exe* program, which copies a set of sampled parameters from *par_val.txt* in their appropriate locations in the SWAT input files. Model parameterization is based on the physical characteristics of the parameters as described in the formulation below:

x__<parname>.<ext>__<hydrogrp>__<soltext>__<landuse>__<subbsn>__<slope>

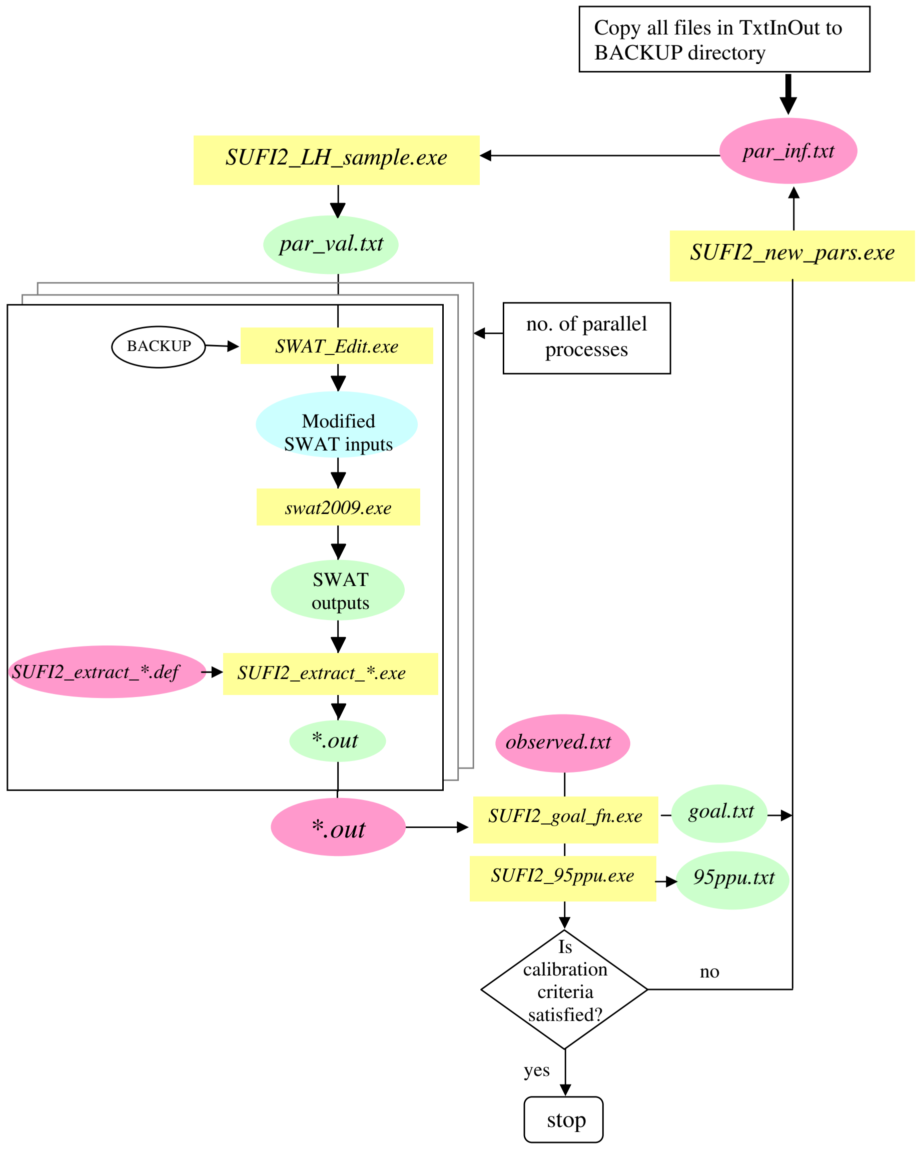

Fig. 2. Schematic coupling of SWAT, and SUFI2. The entire algorithm is run by three batch files: *SUFI2_pre.bat*, which runs the *SUFI2_LH_sample.exe*; *SUFI2_run.bat*, which runs *SWAT_Edit.exe*, *swat2009.exe*, and *SUFI2_extract_*.exe* files, and *SUFI2_post.bat*, which executes the *SUFI2_goal_fn.exe*, *SUFI2_95PPU.exe*, and *SUFI2_new_pars.exe* programs. The symbol * stands for rch, hru, and sub files.

Table 1
Examples of parameterization in SWAT-CUP[a].

Parameter identifiers	Description
r__SOL_K(1).sol____FSL__PAST__1-3	Example of a parameter in .sol file. K of layer 1 of sub-basin 1,2, and 3 with HRUs containing soil texture FSL and landuse PAST will be modified
v__HEAT_UNITS{rotation no,operation no}.mgt	Example a parameter in .mgt file. Management parameters are subject to operation/rotation. Here, heat unit is modified for a certain rotation and operation
v__PLTNFR(1){3}.CROP.DAT	Example of a parameter in crop.dat file. Nitrogen uptake parameter #1 for crop number 3 is modified
v__precipitation( ){1977001-1977361,1978001-1978365,1979003}.pcp	Example of data in a precipitation file. ( ) means all stations from day 1 to day 361 in 1977, and from day 1 to day 365 in 1978, and day 3 in 1979 are modified

[a] r__ indicates relative change, v__ indicates value change, SOL_K is soil hydraulic conductivity, HEAT_UNITS is the total heat units required for crops to reach maturity, PLTNFR is crop's nitrogen uptake parameter, precipitation is the mm of daily precipitation.

where x__ is a code to indicate the type of changes to be applied to a parameter, <parname> is a SWAT parameter name, <ext> is the SWAT file extension, <hydrogrp> is the soil hydrological group, <soltext> is soil texture, <landuse> is the name of the landuse category, <subbsn> is the sub-basin number, and <slope> is the slope of an HRU.

Any combination of the above factors can be used to describe a parameter. If the change is made globally, the identifiers <hydrogrp>, <soltext>, <landuse>, <subbsn>, and <slope> are omitted. Table 1 shows examples of parameterization in different SWAT files.

The encoding scheme allows the parameters to be kept regionally constant to modify a prior spatial pattern, or be changed globally. This gives the analyst greater freedom in depicting the spatial complexity of a distributed parameter. By using this flexibility, a calibration process can be started with a small number of parameters that only modify a given spatial pattern, with more complexity and regional resolution added in a stepwise learning process.

Next, the SWAT model is executed, and the outputs of interest are extracted from SWAT output files (*output.rch*, *output.hru*, *output.sub*). In the last step, post-processing begins, where *SUFI2_post.bat* executes a number of programs, which are briefly described below.

First, *SUFI2_goal_fn.exe* calculates the objective function. SUFI2 allows seven different functions including summation and multiplicative forms of mean square error, r^2, Chi square, Nash–Sutcliffe, weighted r^2, and ranked sum of square error aimed at fitting the frequency distributions. Each formulation may lead to a different result; hence, the final parameter ranges are always *conditioned* on the objective function used. The use of a "multi-objective" formulation (Duan, 2003; Gupta et al., 1998) where different variables are included in the objective function reduces the non-uniqueness problem. An option is now included in SWAT-CUP where variables from different SWAT output files can be included in the objective function. Furthermore, As SUFI2 is iterative; we also found advantages when different objective functions were used in different iterations.

Second, *SUFI2_95ppu.exe* is executed to calculate the 95% prediction uncertainty. SUFI2 describes parameter uncertainty by means of a multivariate uniform distribution in a parameter hypercube, while the output uncertainty is quantified by the 95% prediction uncertainty band (95PPU). SUFI2 maps uncertainties on the parameters in the hydrological model by bracketing all measurements in the 95PPU while minimizing the thickness of the uncertainty band (Fig. 3) (Abbaspour et al., 2007). Although disaggregation of the errors into their source components is quite desirable, de-convolution of the interacting errors both in the simulated and measured signals is quite difficult, particularly in nonlinear models.

Third, *SUFI2_new_pars.exe* runs, which calculates updated parameters for the next iteration. For this step, the users must define physically meaningful absolute minimum and maximum ranges for the parameters being optimized. There is no theoretical basis for excluding any one particular distribution. However, because of the lack of information, we assume that all parameters are uniformly distributed within a region bounded by minimum and maximum values. Because the absolute parameter ranges play a constraining role, they should be as large as possible, yet physically meaningful. Parameter ranges should be based on ranges measured in the watershed and the field under study when possible. SWAT-CUP reads a file called *Absolute_SWAT_Values.txt* where the absolute ranges of all parameters are listed. These ranges can be set by the user. When these data are lacking, then the suggested values from the SWAT database could be used. The absolute parameter ranges in the SWAT database are mostly based on professional judgment and the experimental values taken from the literature (Winchell et al., 2010). Updated parameters are then calculated as:

$$b'_{j,\min} = b_{j,\text{lower}} - \text{Max}\left(\frac{(b_{j,\text{lower}} - b_{j,\min})}{2}, \frac{(b_{j,\max} - b_{j,\text{upper}})}{2}\right) \quad j = 1, \dots, p \tag{1}$$

$$b'_{j,\max} = b_{j,\text{upper}} + \text{Max}\left(\frac{(b_{j,\text{lower}} - b_{j,\min})}{2}, \frac{(b_{j,\max} - b_{j,\text{upper}})}{2}\right) \quad j = 1, \dots, p \tag{2}$$

where b' indicate updated values, $b_{j,\text{lower}}$ and $b_{j,\text{upper}}$ are calculated using the best parameter values of the current iteration as well as the confidence intervals around them, $b_{j,\min}$, and $b_{j,\max}$ are the absolute parameter ranges, and p is the number of parameters. The above formulation, while producing narrower parameter ranges for the subsequent iteration; ensure that the updated parameter ranges are always centered on the best estimates of the current iteration. In the formulation of (1) and (2), the uncertainty in the sensitive parameters decrease faster than those of the insensitive parameters due to the inclusion of the confidence interval that is larger for less sensitive parameters.

After updating, the process repeats until a satisfactory value for the objective function or model evaluation parameters, *P-factor* and *R-factor*, are achieved. These indices are explained below.

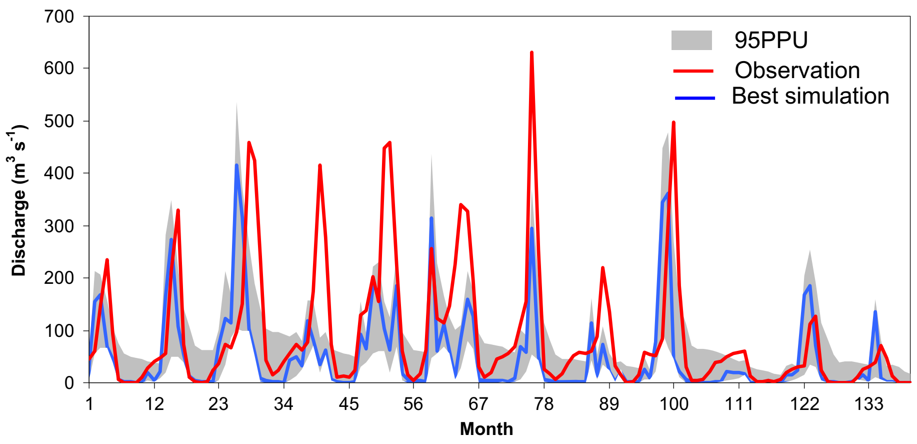

Fig. 3. The 95PPU graph in the SWAT-CUP interface. Also shown are the observed signal and the best simulation. The 95PPU is given for every variable considered in the objective function.

Table 2
Description of the 6 computer systems used to test the parallel Sufi2.

Server 1	**PC 1**
Processor (2 processors)	Processor
24 CPUs	8 CPUs
Processors: Intel(R) Xeon(R) CPU L5640@2.27 GHz (2 processor)	Process: Intel(R) Core(TM) i7 CPU 860@2.8 GHz
RAM = 24.0 GB	RAM = 16.0 GB
System type = 64-bit OS Windows 7	System type = 64-bit OS Windows 7
PC 2	**PC 3**
2 CPUs	2 CPUs
Processor: Intel(R) Core(TM) 2 Duo CPU E8600@3.33 GHz	Processor: Intel(R) Pentium(R) D CPU 3.01 GHz
RAM = 3.46 GB	RAM = 1.00 GB
System type = 32-bit OS Windows XP	System type = 32-bit OS Windows XP
Laptop 1	**Laptop 2**
2 CPUs	2 CPUs
Processor: Intel(R) Core(TM) 2Duo CPU T960000 @ 2.80 GHz	Processor: Intel(R) Core(TM) 2 Duo CPU P8700 @ 2.53 GHz
RAM = 3.0 GB	RAM = 4.0 GB
System type = 32-bit OS Windows XP	System type = 64-bit OS Windows 7

2.3. *Goodness of fit in SUFI2*

As the simulation result is expressed by the 95PPU band, it cannot be compared with observation signals using the traditional indices such as r^2, Nash–Sutcliffe (*NS*) or mean square error (*MSE*). For this reason we used two measures referred to as the *P-factor* and the *R-factor* (Abbaspour et al., 2004, 2007). The *P-factor* is the percentage of the measured data bracketed by the 95PPU. This index provides a measure of the model's ability to capture uncertainties. As all the "true" processes are reflected in the measurements, the degree to which the 95PPU does not bracket the measured data, the predictions are in error. Ideally, *P-factor* should have a value of 1, indicating 100% bracketing of the measured data, hence capturing or accounting for all the correct processes. The *R-factor*, on the other hand, is a measure of the quality of calibration and indicates the thickness of the 95PPU. It is calculated as the average distance between the upper and the lower 95PPU divided by the standard deviation of the observed data:

$$R - factor = \frac{\frac{1}{m}\sum_{i=1}^{m}\left(X_{s,97.5\%} - X_{s,2.5\%}\right)_i}{\sigma_{\text{obs}}} \tag{3}$$

where $X_{s,97.5\%}$ and $X_{s,2.5\%}$ represent the upper and lower boundary of the 95PPU for a simulated variable X_s, and σ_{obs} is the standard deviation of the measured data. *R-factor* should ideally be near zero, hence coinciding with the measured data.

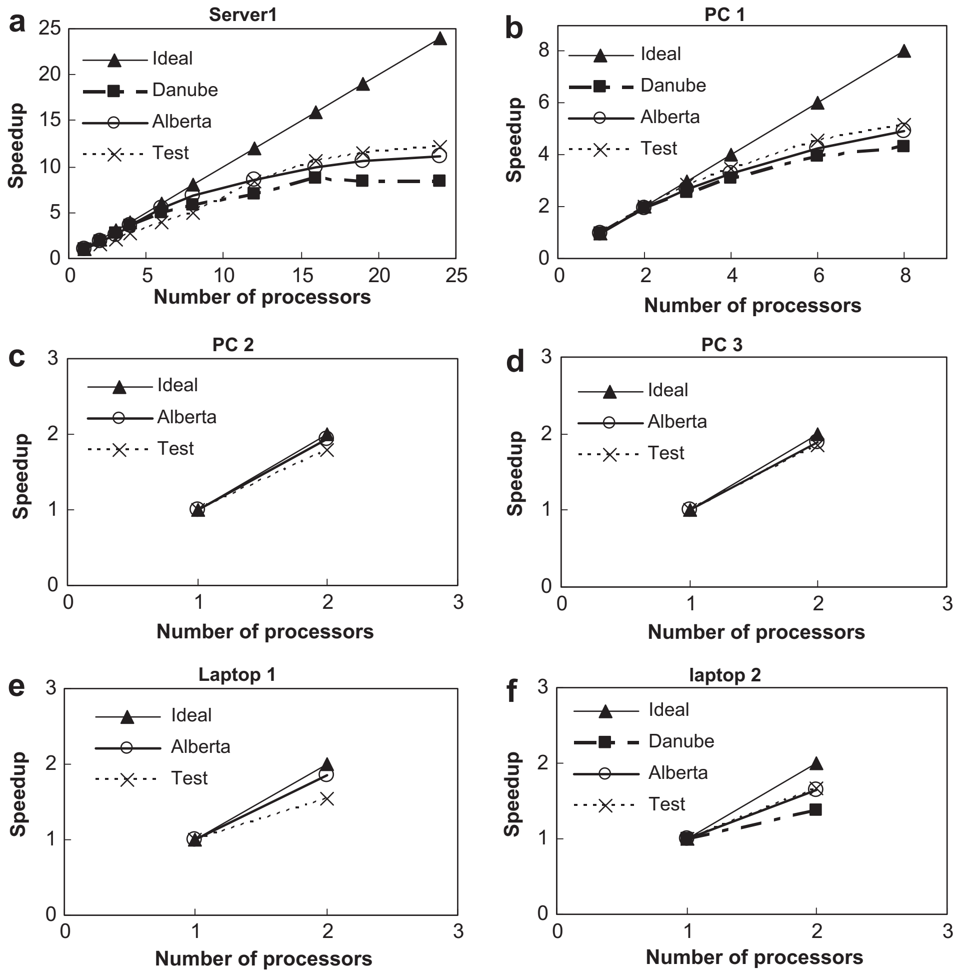

Fig. 4. The speedup achieved for different computer systems and SWAT projects. Number of processors on the horizontal axis indicates the number of parallel jobs submitted. The Figure shows that most projects could be run 10 times faster with about 16 processors. Note that PC 2, PC 3, and Laptop 1 could not handle the size of the Danube project for two parallel runs.

The goodness of calibration and prediction uncertainty is judged on the basis of the closeness of the *P-factor* to 1 (i.e., all observations bracketed by the prediction uncertainty) and the *R-factor* to 0 (i.e., measured and simulated values coinciding). The combination of these two indices together indicates the strength of model calibration and uncertainty assessment as these are intimately linked.

2.4. Parallel processing setup

The structure of parallel SUFI2 is also schematically shown in Fig. 2. Although the parallelization is mostly hidden from the user, different processors are involved. The program initially calculates the number of parallel processes that can be submitted to a system by optimizing the number of CPUs against the required RAM to run a certain project.

Although the SUFI2 program has the potential to execute model tasks in parallel, using the full processing capacity of a computer was challenging. There are several hardware components which affect file read and write speed including CPU, RAM, and the hard disk. It was important to minimize communication to the hard disk by utilizing the RAM. All attempts were made to speedup the runs while using less memory by changing the SUFI2 algorithm, primarily in the following two areas:

1 *Changes in the SWAT-edit.exe program*: SWAT-edit is a program that incorporates updated calibration parameters into SWAT input files. Initially, the files in the BACKUP directory are cached, and held static during the calibration process. All relative changes in the parameters are made with respect to their initial values. The number of SWAT input/output files can vary from tens to hundreds of thousands of files, depending on the project, but each file is only a few KB in size. We took advantage of this characteristic and changed SWAT_edit so that it loads a number of input files on the system's RAM, makes the necessary changes using the cached BACKUP files and writes them to the hard disc. Then it deletes them from the RAM to load the next set of files. This reduces memory usage by up to 90%.
2 *Changes in the SWAT-CUP*: SWAT-CUP functions were split to apply simultaneously on several nodes or parallel jobs. The number of nodes can be the same or fewer than the number of CPUs. Depending on the project size and the available RAM of the system, the program calculates the maximum number of jobs that can be submitted to the system. For example, to have 48 simulations on a machine with 8 processors, 8 nodes can be made, each conducting 6 simulations if the RAM allows. The parallel processing program does not allow the number of nodes to exceed a certain limit if there is a lack of memory. Hence, having a large system RAM ($\geq$16 GB) is an advantage.

The changes in the SWAT-CUP package include: a) changes in user interface to make the parallel option available to users, b) showing the status of each parallel node while the program is running, c) preventing system freezing by giving priorities to other jobs being simultaneously executed, and d) disallowing SWAT-CUP sub-processes to continue running in the background when the program is stopped for any reason. When the program is finished, or stopped unexpectedly, all systems resources are restored.

As parallel SUFI2 is being executed, the user sees the following tasks being performed:

1 *Collect and delete unused files.* These are cached files and files in the *Parallel Processing* directory left over from previous runs in case of program failure or crash. It is necessary to delete these files to clear the RAM and establish the conditions necessary for step 4 below. This step is mostly used when a parallel processing job has already been run in the same directory.
2 *Collect source files*. It counts the number of files in the project directory.
3 *Validate memory usage*. It calculates the number of parallel processes that can be run on the system. In estimating the number of parallel processes, we consider the available RAM of the system, the amount of system cache memory per file, reserve memory for the system's other operations, and SWAT-edit memory usage. SWAT_edit checks that total parallel processing memory

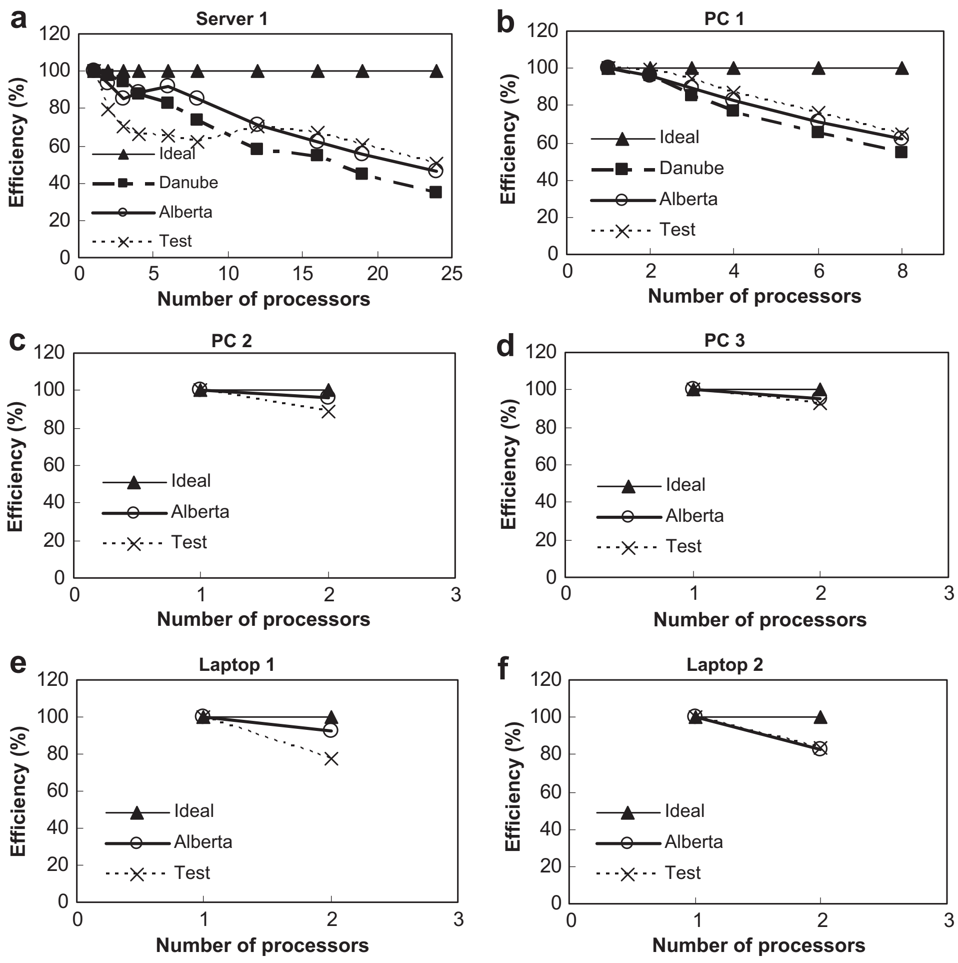

Fig. 5. Percentage efficiency calculated for different computer systems and SWAT projects. The decrease in efficiency is a function of the size of the project and the characteristics of the hard disk.

usage plus protected free memory for other operations on the system are less than the available memory. If this constraint is not fulfilled, the program reduces the maximum number of parallel processes.

4 *Synchronize files*. The program copies the necessary files to each parallel folder from the project folder (these files are pre-fetched so the copy operation is done quite fast) and makes them available for each node to read from the corresponding folder.
5 *Collect BACKUP files*. The program enumerates BACKUP files to make the condition ready for pre-fetching all the files in this directory to make them ready for faster read operation by parallel nodes. In this parallel version of SWAT-Edit there is no need to copy the BACKUP directory in every parallel process directory. All the parallel processors will read from the same BACKUP files.

After finishing the program set up, the simulations are divided between different processors and run in a synchronized manner along each other.

After the parallel runs are finished, parallel processing clean up starts to work by collecting output files from each parallel processing directory and concatenating them in the SUFI2.OUT of the main project directories. Then all unused files are deleted, and the RAM which was used during the parallel process, including the RAM which was used by caching the BACKUP files are released.

Now all is done with the parallel section. The rest is post-processing, which is done in the same way as the single calibration run by executing *SUFI2_post.exe*.

2.5. Case studies and computer systems

To demonstrate the functionality of parallel SUFI2, we evaluated the program with six benchmarks, including three hydrological models tested on six different systems. We used one server, three personal computers, and two laptops. Table 2 has a summary of the attributes of these systems. For hydrological models, we built large, medium, and small projects using the SWAT2009 program. The period of simulation for all runs was set to 5 years.

The large-size project is the Danube River Basin, which covers an area of 801,093 km^2. Danube is Europe's second longest river, flows for a distance of 2826 km and enters the Black Sea east of Izmail (Ukraine) and Tulcea (Romania). We simulate the Danube river basin using SWAT, where we divide the region into 1224 smaller sub-basins taking into account elevation, soil, landuse and climatic information. This resulted in 69,875 HRUs. Running the calibration program SUFI2, 48 simulations took approximately 2 days to run on the server without using the parallel option. The medium-size project is the Alberta project that covers an area of 661,185 km^2. The region is divided into 938 sub-basins for a total of 2689 HRUs. The small-size project (Test project) is a small test example in the SWAT program with only 4 sub-basins and 75 files. It should be mentioned that calibration of the hydrological models is not the focus of this study and we only tried to focus on the speed of the calibration process.

2.6. Performance measures

There are two performance measures that are commonly used to evaluate the performance of parallel computation: *speedup* and *efficiency* (Houstis et al., 1997; Mateos et al., 2010). Speedup for *n* parallel sessions is defined as the computed time of the task when only one processor is used to the computing time when *n* processors are used. The efficiency of a parallel system of *n* processor is defined as the ratio of actual speedup to ideal speedup, where the ideal speedup is equal to the number of processors.

3. Results and discussion

Parallel SUFI2 was tested on different computers using three different projects. Calibration speedup, designed system efficiency, and peak CPU usage of each system versus the number of processors is shown in Figs. 4–6, respectively.

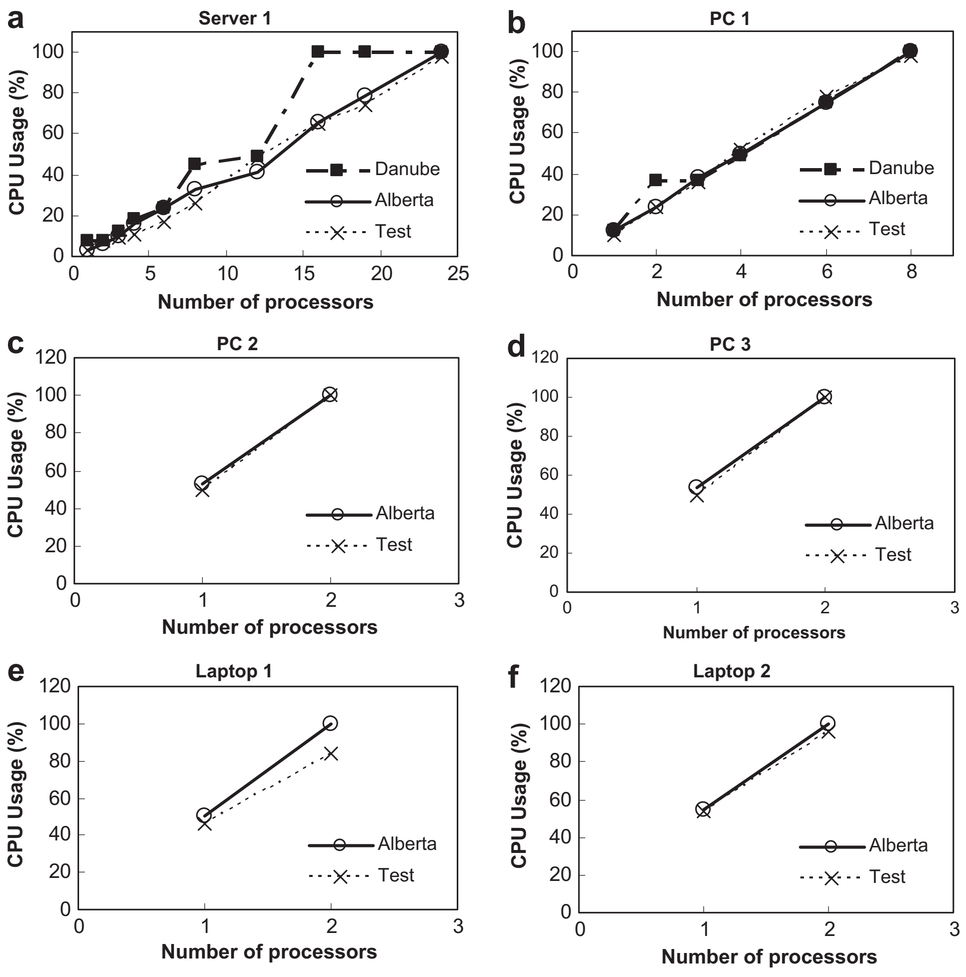

Fig. 6. Peak CPU usage in different computer systems and SWAT projects. As the number of parallel jobs increase, the efficiency in using CPU also increases.

Fig. 4a and b show the speedup of the Danube, the Alberta, and the Test project on machines with 24 and 8 processors, respectively. In the system with 24 CPUs, the speedup of parallel SUFI2 follows closely the ideal speedup up to 8 processors. As the number of the processors increases, the gap between the parallel SUFI2 and the ideal performance grows. As the number of jobs increases, the communication of each CPU with the hard disk increases. The loss of speed is, hence, mostly due to hard disk limitation. The use of Redundant Array of Independent Disks (RAID) or Solid State Drive (SSD) should improve the performance. If the number of parallel jobs is set to 24, then in the Danube project ($24 \times 69,875$) files are simultaneously read and written to the hard disk. For this reason, as the number of parallel processes increases, the speedup decreases for large projects. Hence, Danube shows a smaller speedup than the Alberta or Test project for 24 CPUs (Fig. 4a).

Fig. 4 (c,d,e,f) shows the speedup of running the same projects on 2 PCs with weaker capabilities than PC 1, and two laptops with only 2 CPUs. Because the machines only have 2 processors, we could only submit up to 2 parallel jobs. The speedup achieved using parallel SUFI2 is around 1.9 with 2 processors. This compares quite well to the ideal speedup, indicating that the running time was halved. The small deviations in different machines and projects have to do with the initial state of the computers. For the best result the computers should be restarted before each run to release the remaining occupied memories from previous computer use. It should be noted that the Danube project is missing from Fig. 4c,d,e. This is because the memory limitation, as the size of the project was too large for these machines to run two parallel processes.

Fig. 5 illustrates the efficiency of parallel SUFI2. In general, the efficiency of a parallel system is less than unity because of the system overhead such as the resolution of conflicting demands between shared resources, the communication time between processors, and the inability to keep every processor fully busy. For the ideal case, when the number of processors allocated to a particular task increases, a higher speedup (reduction in computing time) can usually be obtained. The efficiency therefore decreases for the above reasons and the fact that the processors cannot be fully utilized. For small requests, such as the Test case in this study, the overhead introduced by the initial model setup is not compensated because the number of processors is too small. This can be seen in Fig. 5a, which shows that for very small jobs the server becomes more efficient as the number of processors increase.

Fig. 6 shows the peak CPU usage. In Server 1 and PC 1 a non-parallelized program used little of the CPU capacities. As the number of parallel jobs increases, more CPUs are used. In each machine, as the number of parallel jobs equals the number of CPUs, 100% of the CPU is used.

A few recommendations should be followed when running SUFI2 in parallel mode. 1) Do not run multiple SWAT-CUP projects on a single system using the parallel process. 2) It is recommended to restart the computers before starting the parallel calibration project to access the full memory available on the computer although the program itself will clean up the unused files related to the SWAT-CUP project. As the number of parameters increase, SWAT_edit requires more RAM to cache the files. 3) No other memory consuming programs along the parallel SWAT-CUP should be run simultaneously. 4) The number of simulations should be (but not absolutely necessary) a multiple of the number of parallel processing jobs. 5) It is very important to switch off any antivirus program during the parallel program run or exclude the corresponding directory from the virus scan. The antivirus scan decreases the speed of parallel SUFI2 substantially.

4. Conclusion

In this paper, we presented parallel SUFI2, a framework that automatically and transparently parallelizes the SUFI2 optimization program for higher performance calibration purposes. Performance results with both small and large size projects show that parallel SUFI2 achieves good speedup and reasonable scalability in most cases.

Although the parallel SUFI2 is designed to be used on any system, larger time savings can be achieved with multiple CPUs and larger RAM memory. Note that the emphasis of this research was not on achieving the highest possible speedup and that our current implementation is an early proof-of-concept prototype that does not contain optimization or refinement. Computations based on GPU technology hold the promise of achieving greater speedups in execution of hydrologic models (Kalyanapu et al., 2011; Singh et al., 2011). However, we show that parallel SUFI2 is able to achieve reasonable speedup on real-world computation-intensive calibration applications, while significantly exceeding the performance of non-parallelized packages.

Acknowledgments

This project has been funded by the European Commission's Seventh Research Framework through the enviroGRIDS project (Grant Agreement n 226740).

References

Abbaspour, K.C., Faramarzi, M., Ghasemi, S.S., Yang, H., 2009. Assessing the impact of climate change on water resources in Iran. Water Resources Research 45, W10434. doi:10.1029/2008WR007615.

Abbaspour, K.C., Yang, J., Maximov, I., Siber, R., Bogner, K., Mieleitner, J., Zobrist, J., Srinivasan, R., 2007. Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT. Journal of Hydrology 333, 413–430.

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

Arnold, J.G., Srinivasan, R., Muttiah, R.S., Williams, J.R., 1998. Large area hydrologic modeling and assessment - part 1: model development. Journal of the American Water Resource Association 34 (1), 73–89.

Arnold, J.G., Allen, P.M., 1996. Estimating hydrologic budgets for three Illinois watersheds. Journal of Hydrology 176, 57–77.

Beven, K., Binley, A., 1992. The future of distributed models – model calibration and uncertainty prediction. Hydrological Processes 6 (3), 279–298.

Duan, Q., 2003. Global optimization for watershed model calibration. In: Duan, Q., Gupta, H.V., Sorooshian, S., Rousseau, A.N., Turcotte, R. (Eds.), Calibration of Watershed Models. AGU, Washington, DC, pp. 89–104.

Eckhardt, K., Ulbrich, U., 2003. Potential impacts of climate change on groundwater recharge and streamflow in a central European low mountain range. Journal of Hydrology 284, 244–252.

Fernandez-Quiruelas, V., Fernandez, J., Cofino, A.S., Fita, L., Gutierrez, J.M., 2011. Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model. Environmental Modelling and Software 26, 1057–1069.

Fontaine, T.A., Klassen, J.F., Cruickshank, T.S., Hotchkiss, R.H., 2001. Hydrological response to climate change in the Black Hills of South Dakota, USA. Hydrological Sciences Journal 46, 27–40.

Gupta, H.V., Sorooshian, S., Yapo, P.O., 1998. Toward improved calibration of hydrologic models: multiple and non-commensurable measures of information. Water Resources Research 34, 751–763.

Houstis, C., Kapidakis, S., Markatos, E.P., Gelenbe, E., 1997. Execution of computer-intensive applications into parallel machines. Information Sciences 97, 83–124.

Kalyanapu, A.J., Shankar, S., Pardyjak, E.R., Judi, D.R., Burian, S.J., 2011. Assessment of GPU computational enhancement to a 2D flood model. Environmental Modelling and Software 26, 1009–1016.

Kennedy, J., Eberhart, R., 1995. IEEE International Conference on Neural Networks (ICNN 95), Nov 27-Dec 01, 1995 UNIV W Australia, Perth, Australia, Source: 1995 IEEE International Conference on Neural Networks Proceedings 1–6, 1942–1948.

Kuczera, G., Parent, E., 1998. Monte Carlo assessment of parameter uncertainty in conceptual catchment models: the Metropolis algorithm. Journal of Hydrology 211 (1–4), 69–85.

Lecca, G., Petitdidier, M., Hluchy, L., Ivanovic, M., Kussul, N., Ray, N., Thieron, V., 2011. Grid computing technology for hydrological applications. Journal of Hydrology 403, 186–199.

Li, T., Wang, G., Chen, J., Wang, H., 2011. Dynamic parallelization of hydrological model simulation. Environmental Modelling and Software 26, 1736–1746.

Marshall, L., Nott, D., Sharma, A., 2004. A comparative study of Markov chain Monte Carlo methods for conceptual rainfall-runoff modeling. Water Resources Research 40, W02501. doi:10.1029/2003WR002378.

Mateos, C., Zunino, A., Campo, M., 2010. On the evaluation of gridification effort and runtime aspects of JGRIM applications. Future Generation Computer Systems 26, 797–819.

McKay, M.D., Beckman, R.J., Conover, W.J., 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21, 239–245.

Miller, M., 2009. Cloud Computing Pros and Cons for End Users. Article on Internet. http://www.informit.com/articles/article.aspx?p=1324280.

Neal, J.C., Fewtrell, T.J., Bates, P.D., Wright, N.G., 2010. A comparison of three parallelization methods for 2D flood inundation models. Environmental Modelling and Software 25, 398–411.

Schuol, J., Abbaspour, K.C., Sarinivasan, R., Yang, H., 2008a. Estimation of freshwater availability in the West African Sub-continent using the SWAT hydrologic model. Journal of Hydrology 352, 30–49.

Schuol, J., Abbaspour, K.C., Srinivasan, R., Yang, H., 2008b. Modelling blue and green water availability in Africa at monthly intervals and subbasin level. Water Resources Research 44, W07406. doi:10.1029/2007WR006609.

Singh, B., Pardyjak, E.R., Norgren, A., Willemsen, P., 2011. Accelerating urban fast response Lagrangian dispersion simulation using inexpensive graphics processor parallelism. Environmental Modelling and Software 26, 739–750.

Sundaram, K., 2010. Cloud Computing vs. Grid Computing. Article on Internet. http://www.brighthub.com/environment/green-computing/articles/68785.aspx.

Taylor, S.J.E., Popescu, G.V., Pullen, J.M., Turner, S.J., 2004. Panel on distributed simulation and the grid. Proceeding of 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS–RT 2004), Budapest, Hungary October 21-23, 2004. IEEE Comp Soc Tech Comm Parallel Process; IEEE Comp Soc Tech Comm Simulat; IEEE Comp Soc Tech Comm Comp Architecture; Pages: 144-149 DOI: 10.1109/DS-RT.2004.14.

Van Griensven, A., Meixner, T., 2006. Methods to quantify and identify the sources of uncertainty for river basin water quality models. Water Science and Technology 53 (1), 51–59.

Vassilios, P., 2008. Computing, Virtualization, Configuration Management. Article on Internet. http://it.toolbox.com/people/outervillage/.

Vrugt, J.A., Gupta, H.V., Bouten, W., Sorooshian, S., 2003. A shuffled complex evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. Water Resources Research 39 (8), 1201.

Winchell, M., Srinivasan, R., Di Luzio, M., Arnold, J., 2010. ArcSWAT Interface for SWAT2009, User's Guide. Blackland Research and Extension Center Texas AgriLife Research, Temple, Texas 76502, 495 pp.

Yang, J., Reichert, P., Abbaspour, K.C., Yang, H., 2007. Hydrological modelling of the Chaohe basin in China: statistical model formulation and Bayesian inference. Journal of Hydrology 340, 167–182.