Revista Brasileira de Geografia Física

ISSN:1984-2295

Homepage: www.ufpe.br/rbgfe

Paralelização do algoritmo SUFI2: Uma abordagem Windows

Rodrigo de Queiroga Miranda[1], Josiclêda Domiciano Galvíncio[2], Magna Soelma Beserra de Moura[3], Raghavan Srinivasan[4]

[1] Dr. em Desenvolvimento e Meio Ambiente, Pesquisador Pós-doc, Laboratório de Sensoriamento Remoto e Geoprocessamento, Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil, 50670901. (81) 2126-7375. rodrigo.qmiranda@gmail.com (autor correspondente). [2] Dra. em Recursos Naturais, Professora, Laboratório de Sensoriamento Remoto e Geoprocessamento, Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil, 50670901. (81) 2126-7375. josicleda@hotmail.com. [3] Dr. Pesquisadora, Embrapa Semiárido, BR 428, Km 152, Zona Rural - Caixa Postal 23, CEP 56302-970, Petrolina, Pernambuco. (87) 3866-3600. magna_upa@hotmail.com. [4] Professor, Spatial Sciences Laboratory, Texas A&M University, College Station, Texas, United States of America, 77845. +1 (979) 777-9822. r-srinivasan@tamu.edu.

Artigo recebido em 10/03/2017 e aceito em 15/11/2017

R E S U M O

A ferramenta *Soil and Water Assessment Tool* (SWAT) tem sido utilizada para avaliar mudanças do uso da terra nos recursos hídricos em todo o mundo, e como muitos modelos, o SWAT requer calibração. No entanto, o tempo de execução dessas calibrações pode ser bastante longo, reduzindo o tempo disponível para uma análise adequada. Este artigo apresenta uma abordagem Windows para calibrar o SWAT usando um computador *multinodal cluster*, composto por seis computadores com processadores i7 (3,2 GHz, 12 núcleos), 8 GB de RAM e 1 TB HDD cada. O único requisito para este tipo de cluster é ter processadores 64-bit. Nossos computadores foram configurados com o Windows Server HPC 2012 R2, um switch de rede 10/100 e cabos Ethernet comuns. Utilizamos o algoritmo SUFI2 que vem com o pacote SWAT-CUP para realizar calibrações com 100 simulações no nível de nó. As execuações de calibração foram configuradas da seguinte forma: 1-12 (intervalo de 1 processo) e 12-72 (intervalo de 12 processos), resultando em 17 execuções. Cada execução é repetida e os resultados são apresentados como o tempo médio das execuções, a fim de minimizar qualquer influência das flutuações dos recursos. Os resultados mostraram que o tempo de execução foi reduzido em quase a metade usando nove processos (15 min) em comparação com o controle de um nó (28 min). Observamos uma diminuição linear do tempo de execução de um a nove processos. Com processos adicionais, o tempo de execução foi de 23% e estabilizou-se em 80% do controle. Todas as amostras são divididas em cinco etapas: distribuição dos arquivos (2,24% de todo o tempo de processamento), organização das amostras (0,89%), execução do SWAT (47,59%), coleta dos resultados (46,51%) e limpeza (0,28%).

Palavras-chaves: alta performance, hidrologia.

Parallelization of the SUFI2 algorithm: A Windows HPC approach

A B S T R A C T

The Soil and Water Assessment Tool (SWAT) has been used for evaluating land use changes on water resources worldwide, and like many models, SWAT requires calibration. However, the execution time of these calibrations can be rather long, reducing the time available for proper analysis. This paper presents a Windows approach for calibrating SWAT using a multinodal cluster computer, composed of six computers with i7 processors (3.2 GHz; 12 cores), 8 GB RAM and 1 TB HDD each. The only requirement for this type of cluster is to have 64-bit processors. Our computers were setup with Windows Server HPC 2012 R2, a network switch 10/100, and regular Ethernet cables. We used the SUFI2 algorithm that comes with SWAT-CUP package to perform calibrations with 100 simulations at node level. Calibration runs were configured as follows: 1-12 (1 process interval), and 12-72 (12 processes interval), resulting in 17 runs. Each run was repeated three times, and results are presented as the mean execution time, in order to minimize any influence of resources fluctuations. Results showed that time of execution was reduced by almost half by using nine processes (15 min) in comparison with the one node control (28 min). We observed a linear decrease of execution time from one to nine processes. With additional processes, execution time increased about 23% and stabilized at 80% of the control. All processing is divided into five steps: distribute files (2.24% of all processing time), organize samples (0.89%), run SWAT (47.59%), collect results (46.51%) and cleanup (0.28%).

Keywords: high performance, hydrology, modelling, water resources.

Introduction

Many physical and biological processes are difficult to analyze due to their complexity or cost of monitoring. This is especially the case for large-scale environmental analyses. As a result, mathematical models have been used worldwide by researchers and government agencies to simulate these processes. There are two main types of models, and they have different objectives and applications: (*i*) conceptual or semantic models, which are built based only on the relationships between variables and seek to identify and explain a certain phenomenon and its implications on a real system. These models are sometimes essential for creating (*ii*) empirical or physical models, which seek to mimic the behavior of a phenomenon in a real system under physically observed or measured conditions. This type of model also allows the user to make predictions about the behavior of the system under different conditions.

In general, empirical models require a set of input data, $X = (X_1, X_2, \ldots, X_N)$, which produce an output value, $Y = f(x)$. When the value of Y approaches the value that would be obtained by an experiment under conditions X, the model is considered accurate. In addition, when Y represents a phenomenon in the real world, it usually exhibits temporal variation, $Y(t)$, spatial variation, $Y(x, y)$, or both, $Y(t, x, y)$, which considerably increases the complexity of analysis (McKay et al., 1979). Sometimes, the variabilities of X are large and randomly distributed in time or space, making it impossible to obtain accurate simulations without inserting some random variable. These models are classified as stochastic, and in them, Y can be explained by the distribution of its possible values, *e.g.* climate and geological models. Unlike deterministic models, where Y is exclusively determined by the values of X used in modeling, *e.g.* hydrological models.

Hydrological models mathematically simulate the dynamics of a natural system. In recent years, several hydrological models have been created, evolving from rational methods (Mulvany, 1850; Todini, 2007) to recent empirical distributed models (Beven and Kirkby, 1979; Ewen et al., 2000; Srinivasan and Arnold, 1994; Wigmosta et al., 1994). They are used mainly to help manage natural reserves, evaluate anthropic impacts, and plan water use, ensuring that the many existing human and natural demands are met. They also differ greatly in terms of complexity. Some of them are simple and allow only simulation of water balance parameters, *i.e.* surface runoff, percolation, lateral flow, shallow aquifer return flow, and evapotranspiration. Others are complex and are used to model all major physical processes associated with vegetation growth, soil water movement, and climate change at different spatial and temporal scales. But all of these complex models require calibration due to their use of coefficients, nested model components, and spatially heterogeneous inputs that are usually not measured systematically.

Calibration is the process by which the parameters of a model can be approximated to a realistic range for a set of local conditions. The calibration processing consist in determining the most sensitive parameters, and then attributing values to them by comparing modeled and observed data for a certain period. Calibration is usually performed iteratively over a set of possible values for the sensitive parameters. These sets of values are often created through mathematical combination given a predefined limit or interval. Currently, there are many algorithm that provides auto-calibration, for example: SUFI2 (Sequential Uncertainty Fitting 2; Abbaspour et al., 2004) and GLUE (Generalized Likelihood Uncertainty Estimation; Beven and Binley, 1992)..

SUFI2 is the second version of the SUFI (Sequential Uncertainty Fitting) algorithm developed by Abbaspour et al. (1997), and consists of an iterative calibration method that combines optimizations and uncertainty analysis of multiple parameters. This method shares many elements with the GLUE algorithm, but in contrast, SUFI2 aims to define ranges of values for each parameter, rather than specific values (Abbaspour et al., 2004), and SUFI2 is designed to improve accuracy and processing speed, featuring multi-site and multi-objective calibrations. However, despite the increasing speed and capacity of computers over the years, calibration processing time may still be excessive, and several approaches involving parallel processing have been proposed to reduce processing time..

Parallel processing or multiprocessing, which is understood as the simultaneous execution of multiple operations or tasks using computational resources at any level, have been successfully implemented using cloud (Yalew et al., 2013) and grid processing (Yalew et al., 2013), either through direct parallelization of source codes (Wu et al., 2013), or spawning multiple executables with different sets of parameters (Rouholahnejad et al., 2012). In this study, we describe a different approach to speed up calibration using SUFI2 and Microsoft® Windows HPC to spawn processes across a multinodal cluster computer.

Methodology

The model SWAT (Soil and Water Assessment Tool)

For this study we chose the model SWAT (Soil and Water Assessment Tool), because it is closely associated to SUFI2 in many publication. In fact, SUFI2 has been reported as the most efficient algorithm for calibrating SWAT (Nkonge et al., 2014; Wu et al., 2013; Yang et al., 2008). SWAT is one of the most widely used watershed-scale hydrological models, and its use by both public and private-sector users is expanding (Bressiani et al., 2015). It provides results at four temporal resolutions: yearly, monthly, daily and sub-daily. Current large-scale SWAT applications include the African (Awotwi et al., 2015; Bossa et al., 2012; Faramarzi et al., 2013; Pouyan Nejadhashemi A, 2011; Welderufael et al., 2013) and European continents (Koch et al., 2013; Malagò et al., 2015; Oeurng et al., 2011; Sellami et al., 2013; Taylor et al., 2016), and the United States of America (Bieger et al., 2015; Chien et al., 2013; G. W. Marek et al., 2016; Gary W. Marek et al., 2016).

SWAT is a semi-distributed model in which a watershed is divided into Hydrological Response Units (HRUs) that correspond to homogeneous areas with only one land use, slope and soil type. HRUs are represented in the model as percentages of sub-basin or watershed area, and may not be spatially explicit. SWAT has five main components: hydrology, soil, climate, vegetation and land management. The hydrological component is its main modulator, directly influencing plant growth, sediment, nutrient and pesticide movements. SWAT hydrology is based on the water balance equation:

$$SW_f = SW_i + \sum_{t=1}^{t}(P - Q_s - ET - W_s - Q_{gw}) \quad (1)$$

where SW_f and SW_i are the final and initial water in the contents of the soil, respectively (mm), t is time (days), P is precipitation (mm), Q_s is surface runoff (mm), ET is evapotranspiration (mm), W_s is percolation (mm) and Q_{gw} is base flux (mm). Detailed equations can be found in Arnold et al. (1998) and Arnold et al. (2010).

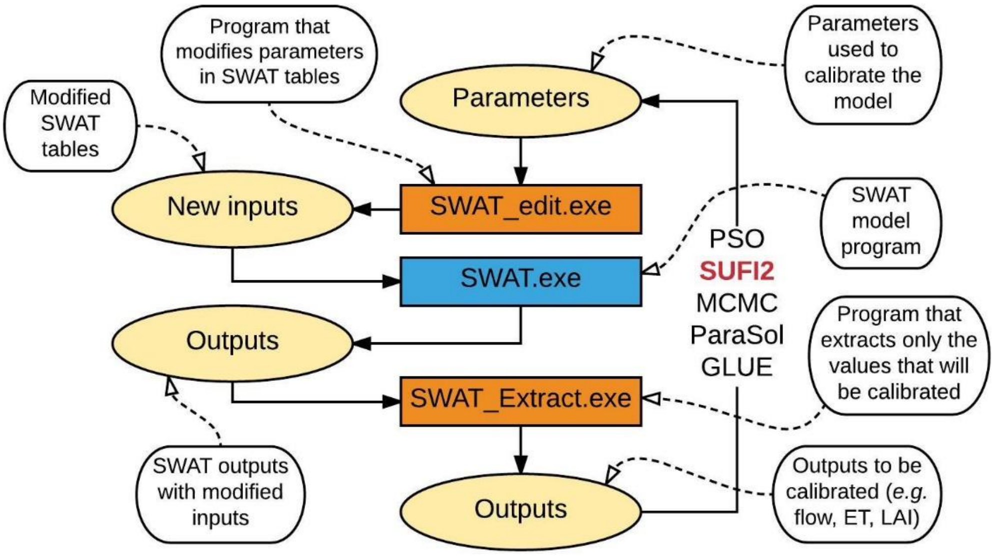

Figure 1. Adaptation of the SWAT-CUP structure published by Rouholahnejad et al. (2012).

SWAT-CUP (SWAT Calibration and Uncertainty Programs)

To facilitate SWAT calibration processes, we used the SWAT-CUP software (SWAT Calibration and Uncertainty Programs, Rouholahnejad et al., 2012), which has been used by several researchers to carry out calibrations, and is recommended by SWAT developers. SWAT-CUP uses several calibration algorithms in a single graphical interface to optimize SWAT inputs: SUFI2 (Sequential Uncertainty Fitting 2), GLUE (Generalized Likelihood Uncertainty Estimation), PSO (Particle Swarm Optimization; Kennedy and Eberhart, 2002), ParaSol (Parameter Solution; van Griensven and Meixner, 2007), e MCMC (Markov chain Monte Carlo; Gilks, 2005). SWAT-CUP processing consists of three steps performed iteratively until SWAT outputs are determined to be sufficiently accurate (Figure 1): (*i*) modify the SWAT inputs, (*ii*) run SWAT, and (*iii*) extract its outputs. More details can be found in Rouholahnejad et al. (2012).

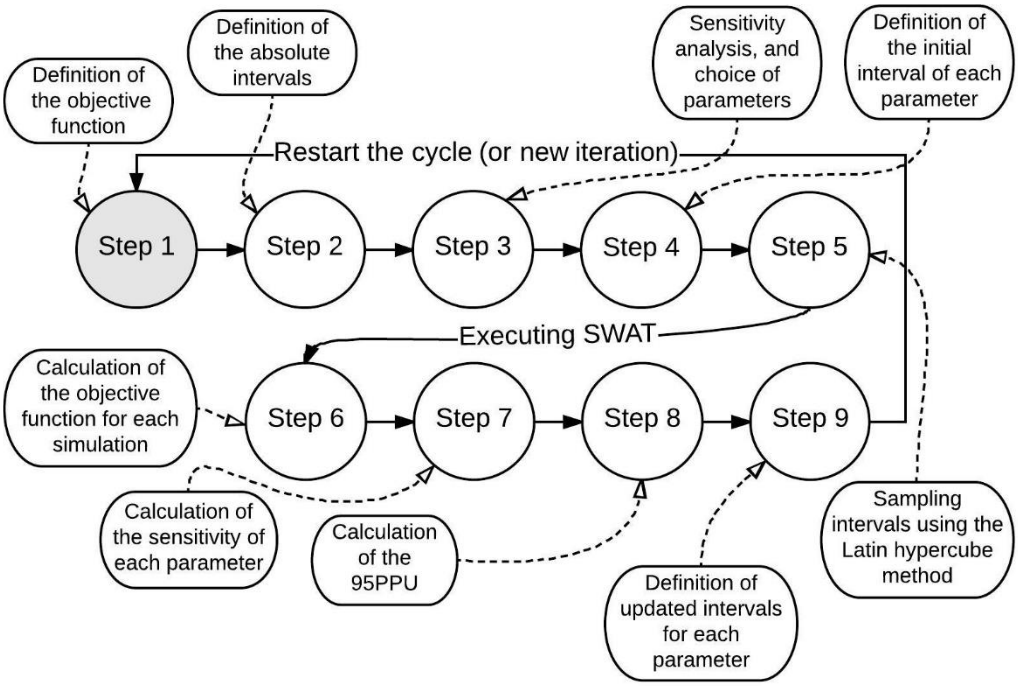

Figure 2. Diagram of the steps of SUFI2 (Sequential Uncertainty Fitting 2).

The algorithm SUFI2 (Sequential Uncertainty Fitting 2)

The operational details of SUFI2 was described by Abbaspour et al. (2004), and can be divided into 9 steps (Figure 2), which begin with the definition of the objective function to be used by the algorithm to evaluate the accuracy of a given set of parameters. An objective function is a category or group of mathematical indicators that ranges from a simple sum of values to complex indices, such as the Coefficient of determination (r^2), Chi-square (chi2), Nash-Sutcliffe (NS), Percent Bias (PBIAS) and Kling Gupta Efficiency (KGE). In SUFI2, one can use either a single function or multiple functions (multi-criteria calibration) with only one or many different variables (*e.g.* flow and nutrient) from one or more measurements points (multi-objective formulation). Also these variables or part (or compartment) of their temporal series can be assigned different weights that are taken into account in the calculation of the objective function. The second step is to determine realistic intervals (the possible max and min values) between which each parameter should vary. These intervals prevent the algorithm from tagging a parameter as calibrated with unrealistic values, *e.g.* BLAI (maximum Leaf Area Index) must be between 0.3 m^2/m^2 and 3.1 m^2/m^2 for a certain biome, or it is unreal. The third step involves the sensitivity analysis of all parameters related to the output variable used to calibrate the model, and this analysis allows not only to define which parameters will compose the calibration, but also helps the fourth step, which consists in choosing the maximum and minimum values between which each parameter must vary. The fourth step can sometimes be confused with the second step. Unlike the second step, this interval is not absolute and must vary with each iteration. For example, in a given study area, through field experiments, sensitivity analyses or remote measurements, we found that BLAI should vary between 2.0 and 2.5 m^2/m^2. This is our initial interval, which may move, narrow or sometimes expand, but always within the absolute range (*e.g.* from 0.3 to 3.1 m^2/m^2) that keeps it realistic. The fifth step consists in sampling n times (n is the number of iterations) the intervals defined on step four of all parameters chosen on step three. In order to maximize the speed and processing time of the algorithm SUFI2, the sampling is done using the Latin hypercube method, which ensures that all intervals are completely analyzed.

From now on, all evaluations and statistical analysis of simulations are performed, starting with the sixth step that calculates the objective function for each simulation. The seventh step then computes a series of equations that are applied to obtain the sensitivity of each parameter. First, (*i*) the sensitivity matrix and (*ii*) Hessian matrix are calculated as described in Abbaspour et al. (2004);

and then, (*iii*) an estimate of the lower limit of the covariance matrix of each parameter is obtained along with its (iv) standard deviation and (v) 95% confidence interval. Details can be found in Press et al. (1992). Finally, the sensitivity of each parameter is calculated by averaging all columns of the sensitivity matrix. In step eight, the 95PPUs of all parameters are computed. SUFI2 considers the uncertainty of predictions to be the 2.5º (q_u) and 97.5º (q_i) percentile of the cumulative distribution of simulated points. The optimized set of values is calculated for each parameter q using the percentage of observed data that lies within the 95PPU region. The mean distances between upper and lower 95PPU are determined by the following equation:

$$\bar{a} = \frac{1}{K}\sum_{l=1}^{K}(q_u - q_i)\, l \qquad (2)$$

where $\bar{a}$ is the parameter uncertainty, l is a sequential counter and K is the total number of observations for each parameter q. The best result is when 90% or more of observed values are within 95PPU region, r^2 is greater than or equal to 0.8, and $\bar{a}$ is close to zero.

Normally, in the first iterations, $\bar{a}$ is high, but as iterations continue, ā approaches zero, and the percentage of observed values within the 95PPU range increases. This happens because at each iteration the values defined in the fourth step are updated, producing smaller intervals for achieving a better model accuracy. The new interval determined in the ninth step only increases when the best values are close to the limits of the previous interval, but never exceeding the values defined in the second step.

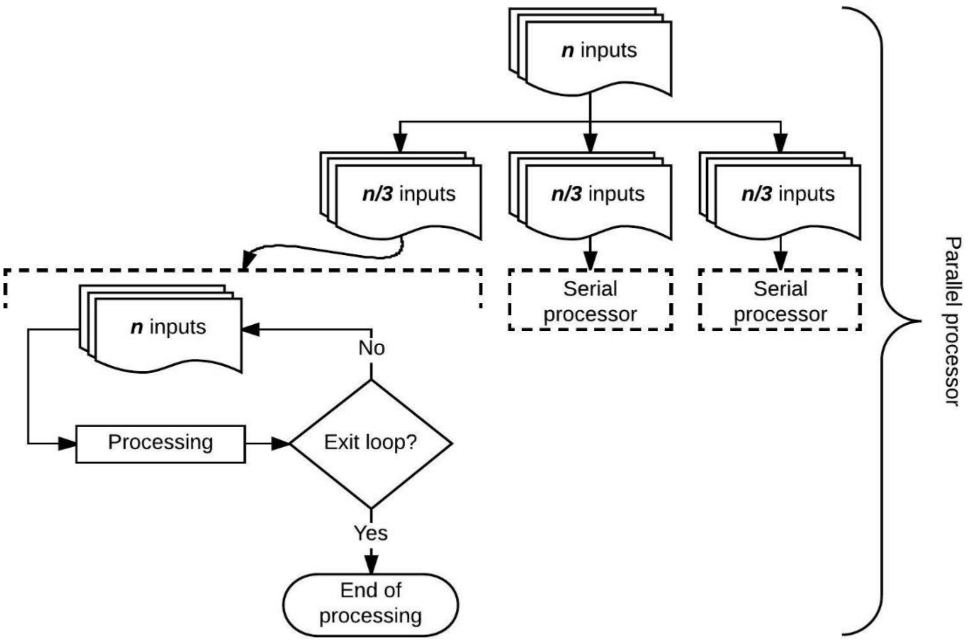

Figure 3. Scheme of the parallelization of a computer code.

Microsoft® HPC Pack

The Microsoft® HPC Pack is designed for high-end applications that require numerous computers to be clustered together by multiple network cables to achieve scalable high processing power. More precisely, the HPC Pack is a collection of tools that allows creating and managing nodes from a multinodal cluster computer. It includes management tools, a job scheduler, and the ability to join workstation nodes and unmanaged servers to a cluster. Its latest release is a free download available on the Microsoft Download Center (https://www.microsoft.com/en-us/download/). These tools manage the complex processing that actually occurs in computers. HPC Pack allows to interact starting and controling processes or tasks that can leverage part or all resources available in a cluster computer; and that can happen through two ways: (*i*) a friendly interface (HPC Manager) and (*ii*) and series of command-line applications that can be automated using the programming languages Poweshell and Batch.

In order to achieve simplicity, in this study we used Batch, which is a scripting language that can be executed by a command-line interpreter available in the Microsoft® DOS and Command Prompt in Windows operational systems, which is essentially a legacy environment carried forward that copies many DOS commands. A Batch file consists of a series of commands stored in ACSII text files that allows a user to accomplish specific tasks such as automating repetitive commands and decision making through looping and conditional branching. The HPC Pack was used before for calibrating the SWAT model, but within a grid environment. Humphrey et al. (2012) utilized the Pack to access cloud resources from Microsoft® Azure.

Parallelization scheme

From all of the procedures related to SWAT modelling, the one that is most time consuming is calibration, because it is an iterative process in which the model is executed repeatedly until a satisfactory results are obtained; and in this point, parallel processing can make a substantial difference in the total processing time. A well-known way to insert parallelization into a programming code is by identifying the iteration points, and then remapping the processing flux. In a serialized processing, a program or function is repeated several times with different inputs until there are no more inputs available to process. When parallel processing is implemented, the inputs are divided into groups of inputs, and each group is serially processed at the same time. That is, if the number of groups is equal to the number of inputs (one input per group), the processing becomes 100% parallelized, and all processes are to end simultaneously; On the other hand, if the number of groups is equal to one, the processing can be considered fully serialized (Figure 3).

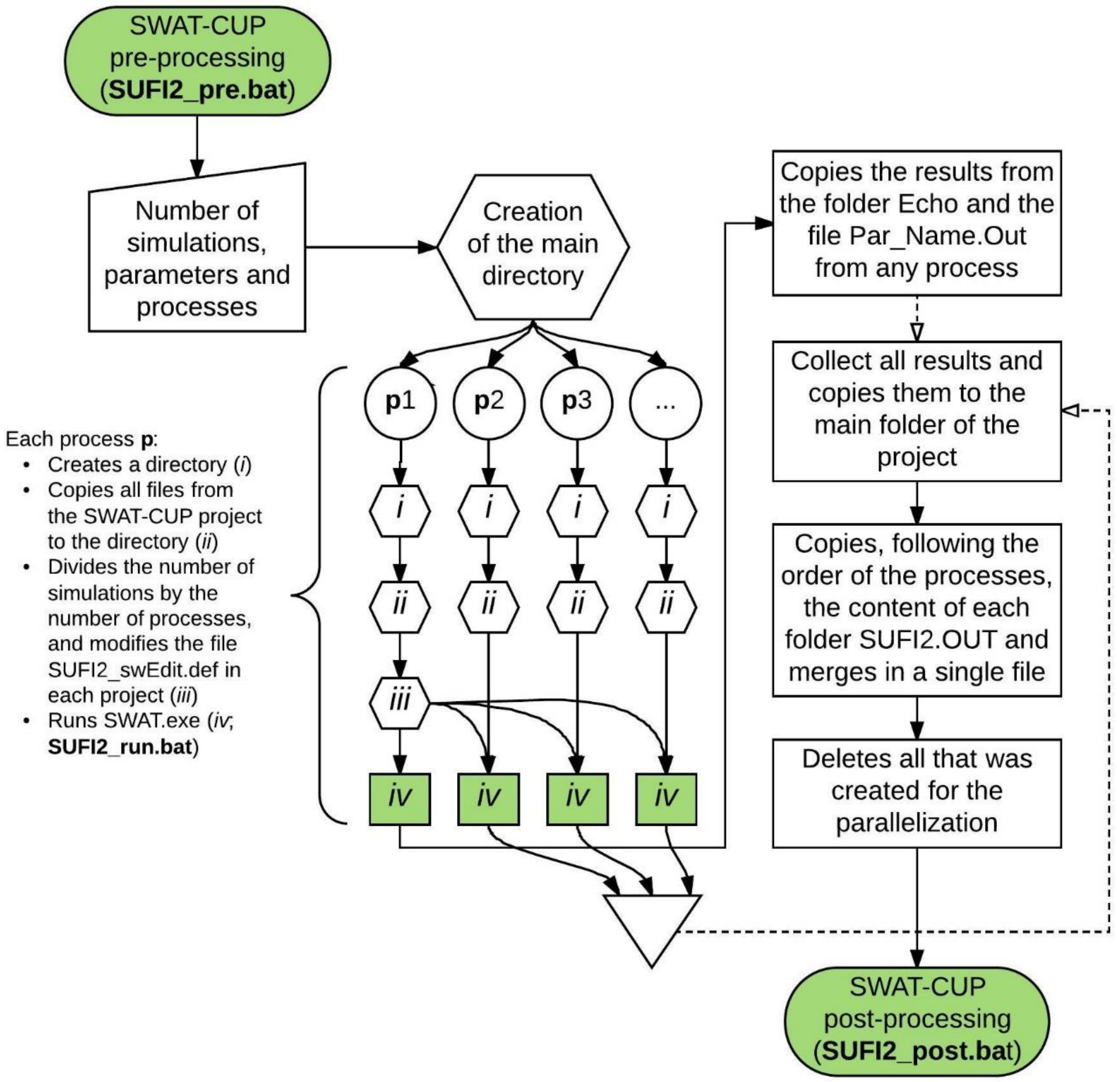

Figure 4. Scheme of the parallelization of SUFI2 (Sequential Uncertainty Fitting 2).

In SWAT-CUP, three files are used to run SUFI2: SUFI2_pre.bat, SUFI2_run.bat, SUFI2_post.bat. The SUFI2_pre.bat divides the intervals of the parameters in order to optimize calibration; SUFI2_run.bat runs SWAT.exe one or more times, saving the desired results of each iteration in text files; and SUFI2_post.bat performs all statistics. For parallel processing, we modified SUFI2_run.bat. Instead for running sufi2_execute.exe, it runs our custom Batch file as illustrated in Figure 4. All processing was divided into five steps: distribute files, organize samples, run SWAT, collect results and cleanup. The first step will make n copies of the SWAT-CUP project depending on how many processes will be running. The folders 'ParallelProcessing', 'Iterations' and the main directory were avoided, since we do not need these folders to run SWAT. The second step consists in dividing the samples to prevent two processes from working on the same simulation. The Figure 5 illustrates the algorithm that we created for this purpose. It treats the simulations as a vector of simulations, $X = (S_1, S_2, S_3, \ldots, S_N)$, where S_N is a simulation N, and divides it equally, distributing any remainder across the groups. For example, if X is composed by 10 simulations, and those must be divided into 4 groups: $10/4 = 2\ R\ 2$, the remainder is distributed from the first group to the last. In this case, 2 will be added to the first two groups as in equation 3:

$$f(X) = \begin{matrix} 3: (S_1, S_2) + (S_3) \\ 3: (S_4, S_5) + (S_6) \\ 2: (S_7, S_8) \\ 2: (S_9, S_{10}) \end{matrix} \qquad (3)$$

To specify which simulations each processes will run, our Batch file modifies the file SUFI2_swEdit.def of the SWAT projects. This file was first created for this purpose, and is the most important SUFI2's feature that allows its parallelization. After that, Batch file starts SWAT in all projects, and once it is finished, all results must be copied to the original project. Some of them need to be processed first, and that is accomplished by the following code, where %procs% is the number of processes that we used and %file% is the file that is in all projects with different results from different simulations and will be merged into one. At the end, the Batch file deletes every folder and file created for the parallelization process.

```
- Input the vector's length;
- Input the number of groups;

- Initializes all necessary variables.
Initialize D as vector's length / number of groups;
Initialize R as vector's length % number of groups;
Initialize L as FALSE;
Initialize MAX as 0;
Initialize MIN as 0;

- Creates a loop that for i = 1, until i is equal to
- vector's length, keeps adding 1 to i in each iteration.
FOR (i = 1, i is equal to vector's length, i + D) DO
    - L serves as a "first iteration" marker
    - to avoid that the first MIN is not 1.
    IF L is TRUE THEN
        Define MIN as MAX + 1;
    ELSE
        Define MIN as i;
    - While there is a positive remainder
    - keeps substracting 1 from it in each
    - iteration until there is no remainder left.
    IF R is not 0 THEN
        Define L as TRUE;
        Subtract 1 from R;
        - It is here where the remainder is
        - added to the group. Always 1 of the
        - total remainder per group.
        Define MAX as MIN + D;
    ELSE
        - In case all the remainder is distributed
        - across the groups, we do not want
        - to keep adding 1 to the group, so we
        - remove it at all the subsequent iterations.
        Define MAX as MIN + D - 1;
    - Defines the max value as the vector's length
    - to avoid a group that will make the sum of
    - all groups to be greater than the vector's length.
    IF MAX >= vector's length THEN
        Define MAX as vector's length;
```

Figure 5. Pseudocode to get the groups of simulations in function of the number of simulations (vector's length) and number of groups (number of processes to be spawn).

Code 1. Batchfile code that collects and writes all of the results in a single file.

```
for /l %%n in (1,1,%procs%) do (
   for /f "tokens=*" %%l in (main_directory\worker%%n\SUFI2.OUT\%file%) do (
      echo %%l >> SUFI2.OUT\%file%
   )
)
```

Case study and computer configuration

Besides software modifications, for a parallel processing to occur, one must have specific computational resources that support this type of computation. There are three main architectures used for parallelize processing: grid, cloud, and processor cores on a single computer (Figure 6). In this study, we used a grid architecture composed by six computers with Intel® i7 X980 processors (3.33 GHz; 12 cores), 8 GB RAM (DDR3; 1,333 MHz), 1 TB HDD (7,200 RPM) and 2 PCI-E Gigabit Ethernet Controller (10/100/1000) in each one, although the only requirement for this type of cluster is to have 64-bit processors. Our computers were set up with Windows Server HPC 2012 R2, a network switch 10/100, and regular Ethernet cables (CAT-5). We used the sample project of SUFI2 algorithm that comes with the SWAT-CUP package to perform calibrations with 100 simulations at node level. Calibration tests were configured as follow: 1-12 (1 process interval), and 12-72 (12 processes interval), resulting in 17 essays. Each test was repeated three times and results are presented as the mean execution time, in order to minimize any influence of resource fluctuations.

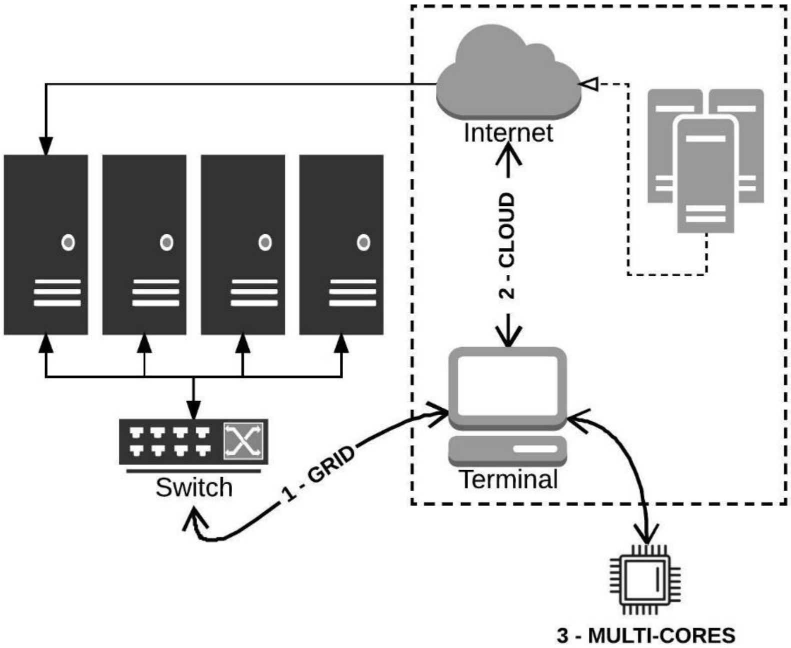

Figure 6. Main architectures used to parallelize a computer processing.

Results and discussion

Results showed that time of analysis was cut almost in half using nine processes (15 min) in relation to the control node (28 min). We observed a linear decrease of execution time from one to nine processes, and then it increased about 23% and stabilized at 80% of the control. The five processing steps required different processing times: distribute files (2.24% of all processing time), organize samples (0.89%), run SWAT (47.59%), collect results (46.51%) and cleanup (0.28%). The SWAT execution step required the most time, and depending on the available computer hardware, it can be rather long, making it difficult to perform a proper calibration (Rouholahnejad et al., 2012). Our results showed that parallel processing of SUFI2 accelerated this step substantially. Collecting SWAT outputs used almost as much time as running SWAT because this step was not parallelizable; it ran in series.Files in directory SUFI2.OUT cannot be written asynchronically because SWAT needs them to be in the correct order to correctly read and process them. Reading and writing operations (Input/Output; I/O) in Batch are restricted in speed and format flexibility; thus, this step may be drastically improved by implementing a faster language like Python, C or Fortran; or installing hardware upgrades.

Conclusion

In this study, we describe how to parallelize the widely used SUFI2 algorithm on Microsoft® Windows to reduce algorithm execution time. Using a sample project, we derived a simple Batch file that is able to distribute processing across multiple machines. Our study used grid architecture technologies and hardware for model auto-calibrating purposes. Experimental results showed a significant improvement of performance in computation time. The major bottleneck of this method was the I/O processing speed of Batch in merging the different model results. Although we focused on a simple programming language and a common platform, we believe that this research can be used to make high performance computing applications more intuitive and easy to use.

Acknowledgements

We thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazilian Coordination for the Improvement of Higher Level Personnel) and FACEPE (Fundação de Amparo à Ciência Tecnologia do Estado de Pernambuco) for funding this study through the projects PVE A103/2013 and APQ 0646-9.25/16 respectively. The CNPq for Edital Universal 2016, PQ Scholarship of the second author and SUPER ptoject. We thank C. Allan Jones for his editorial assistance.

References

Abbaspour, K.C., Johnson, C. a., van Genuchten, M.T., 2004. Estimating Uncertain Flow and Transport Parameters Using a Sequential Uncertainty Fitting Procedure. Vadose Zo. J. 3, 1340. doi:10.2136/vzj2004.1340

Abbaspour, K.C., van Genuchten, M.T., Schulin, R., Schläppi, E., 1997. A sequential uncertainty domain inverse procedure for estimating subsurface flow and transport parameters. Water Resour. Res. 33, 1879–1892. doi:10.1029/97WR01230

Arnold, J.G., Gassman, P.W., White, M.J., 2010. New Developments in the SWAT Ecohydrology Model, in: 21st Century Watershed Technology: Improving Water Quality and Environment. ASABE, St. Joseph, USA, pp. 21–24.

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

Awotwi, A., Yeboah, F., Kumi, M., 2015. Assessing the impact of land cover changes on water balance components of White Volta Basin in West Africa. Water Environ. J. 29, 259–267. doi:10.1111/wej.12100

Beven, K., Binley, A., 1992. The future of distributed models: Model calibration and uncertainty prediction. Hydrol. Process. 6, 279–298. doi:10.1002/hyp.3360060305

Beven, K.J., Kirkby, M.J., 1979. A physically based, variable contributing area model of basin hydrology / Un modèle à base physique de zone d'appel variable de l'hydrologie du bassin versant. Hydrol. Sci. Bull. 24, 43–69. doi:10.1080/02626667909491834

Bieger, K., Rathjens, H., Allen, P.M., Arnold, J.G., 2015. Development and Evaluation of Bankfull Hydraulic Geometry Relationships for the Physiographic Regions of the United States. JAWRA J. Am. Water Resour. Assoc. 51, 842–858. doi:10.1111/jawr.12282

Bossa, A.Y., Diekkrüger, B., Igué, A.M., Gaiser, T., 2012. Analyzing the effects of different soil databases on modeling of hydrological processes and sediment yield in Benin (West Africa). Geoderma 173–174, 61–74. doi:10.1016/j.geoderma.2012.01.012

Bressiani, D. de A., Gassman, P.W., Fernandes, J.G., Garbossa, L.H.P., Srinivasan, R., Bonumá, N.B., Mendiondo, E.M., 2015. Review of Soil and Water Assessment Tool (SWAT) applications in Brazil: Challenges and prospects. Int. J. Agric. Biol. Eng. 8, 1–27. doi:10.3965/j.ijabe.20150803.1765

Chien, H., Yeh, P.J.-F., Knouft, J.H., 2013. Modeling the potential impacts of climate change on streamflow in agricultural watersheds of the Midwestern United States. J. Hydrol. 491, 73–88. doi:10.1016/j.jhydrol.2013.03.026

Ewen, J., Parkin, G., O'Connell, P.E., 2000. SHETRAN: Distributed River Basin Flow and Transport Modeling System. J. Hydrol. Eng. 5, 250–258. doi:10.1061/(ASCE)1084-0699(2000)5:3(250)

Faramarzi, M., Abbaspour, K.C., Ashraf Vaghefi, S., Farzaneh, M.R., Zehnder, A.J.B., Srinivasan, R., Yang, H., 2013. Modeling impacts of climate change on freshwater availability in Africa. J. Hydrol. 480, 85–101. doi:10.1016/j.jhydrol.2012.12.016

Gilks, W.R., 2005. Markov Chain Monte Carlo, in: Encyclopedia of Biostatistics. John Wiley & Sons, Ltd, Chichester, UK. doi:10.1002/0470011815.b2a14021

Humphrey, M., Beekwilder, N., Goodall, J.L., Ercan, M.B., 2012. Calibration of watershed models using cloud computing, in: 2012 IEEE 8th International Conference on E-Science. IEEE, Chicago, USA, pp. 1–8. doi:10.1109/eScience.2012.6404420

Kennedy, J., Eberhart, R., 2002. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks. IEEE, pp. 1942–1948. doi:10.1109/ICNN.1995.488968

Koch, S., Bauwe, A., Lennartz, B., 2013. Application of the SWAT Model for a Tile-Drained Lowland Catchment in North-Eastern Germany on Subbasin Scale. Water Resour. Manag. 27, 791–805. doi:10.1007/s11269-012-0215-x

Malagò, A., Pagliero, L., Bouraoui, F., Franchini, M., 2015. Comparing calibrated parameter sets of the SWAT model for the Scandinavian and Iberian peninsulas. Hydrol. Sci. J. 60, 1–19. doi:10.1080/02626667.2014.978332

Marek, G.W., Gowda, P.H., Evett, S.R., Baumhardt, R.L., Brauer, D.K., Howell, T.A., Marek, T.H., Srinivasan, R., 2016. Estimating Evapotranspiration for Dryland Cropping Systems in the Semiarid Texas High Plains Using SWAT. JAWRA J. Am. Water Resour. Assoc. 52, 298–314. doi:10.1111/1752-1688.12383

Marek, G.W., Gowda, P.H., Marek, T.H., Porter, D.O., Baumhardt, R.L., Brauer, D.K., 2016. Modeling long-term water use of irrigated cropping rotations in the Texas High Plains using SWAT. Irrig. Sci. 1–13. doi:10.1007/s00271-016-0524-6

McKay, M.D., Beckman, R.J., Conover, W.J., 1979. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics 21, 239–245. doi:10.1080/00401706.1979.10489755

Mulvany, T.J., 1850. On the use of self registering rain and flood gauges. Trans. Inst. Civ. Eng. 4, 1–8.

Nkonge, L.K., Sang, J.K., Gathenya, J.M., Home, P.G., 2014. Comparison of two Calibration-uncertainty Methods for Soil and Water Assessment Tool in Stream Flow Modeling. J. Sustain. Res. Eng. 1, 40–44.

Oeurng, C., Sauvage, S., Sánchez-Pérez, J.-M., 2011. Assessment of hydrology, sediment and particulate organic carbon yield in a large agricultural catchment using the SWAT model. J. Hydrol. 401, 145–153. doi:10.1016/j.jhydrol.2011.02.017

Pouyan Nejadhashemi A, I.M., 2011. Evaluation of Swat Performance on a Mountainous Watershed in Tropical Africa. J. Waste Water Treat. Anal. s3, 1–7. doi:10.4172/2157-7587.S14-001

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes in C (2nd ed.): The Art of Scientific Computing, 2nd ed. Cambridge University Press, New York, USA.

Rouholahnejad, E., Abbaspour, K.C., Vejdani, M., Srinivasan, R., Schulin, R., Lehmann, A., 2012. A parallelization framework for calibration of hydrological models. Environ. Model. Softw. 31, 28–36. doi:10.1016/j.envsoft.2011.12.001

Sellami, H., La Jeunesse, I., Benabdallah, S., Baghdadi, N., Vanclooster, M., 2013. Uncertainty analysis in model parameters regionalization: a case study involving the SWAT model in Mediterranean catchments (Southern France). Hydrol. Earth Syst. Sci. Discuss. 10, 4951–5011. doi:10.5194/hessd-10-4951-2013

Srinivasan, R., Arnold, J.G., 1994. Integration of the basin-scale water quality model with GIS. J. Am. Water Resour. Assoc. 30, 453–462. doi:10.1111/j.1752-1688.1994.tb03304.x

Taylor, S.D., He, Y., Hiscock, K.M., 2016. Modelling the impacts of agricultural management practices on river water quality in Eastern England. J. Environ. Manage. 180, 147–163. doi:10.1016/j.jenvman.2016.05.002

Todini, E., 2007. Hydrological catchment modelling: past, present and future. Hydrol. Earth Syst. Sci. 11, 468–482. doi:10.5194/hess-11-468-2007

van Griensven, A., Meixner, T., 2007. A global and efficient multi-objective auto-calibration and uncertainty estimation method for water quality catchment models. J. Hydroinformatics 9, 277. doi:10.2166/hydro.2007.104

Welderufael, W.A., Woyessa, Y.E., Edossa, D.C., 2013. Impact of rainwater harvesting on water resources of the modder river basin, central region of South Africa. Agric. Water Manag. 116, 218–227. doi:10.1016/j.agwat.2012.07.012

Wigmosta, M.S., Vail, L.W., Lettenmaier, D.P., 1994. A distributed hydrology-vegetation model for complex terrain. Water Resour. Res. 30, 1665–1679. doi:10.1029/94WR00436

Wu, Y., Li, T., Sun, L., Chen, J., 2013. Parallelization of a hydrological model using the message passing interface. Environ. Model. Softw. 43, 124–132. doi:10.1016/j.envsoft.2013.02.002

Yalew, S., van Griensven, A., Ray, N., Kokoszkiewicz, L., Betrie, G.D., 2013. Distributed computation of large scale SWAT models on the Grid. Environ. Model. Softw. 41, 223–230. doi:10.1016/j.envsoft.2012.08.002

Yang, J., Reichert, P., Abbaspour, K.C., Xia, J., Yang, H., 2008. Comparing uncertainty analysis techniques for a SWAT application to the Chaohe Basin in China. J. Hydrol. 358, 1–23. doi:10.1016/j.jhydrol.2008.05.012