Environmental Modelling & Software 41 (2013) 223–230

Contents lists available at SciVerse ScienceDirect

Environmental Modelling & Software

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

Distributed computation of large scale SWAT models on the Grid

S. Yalew [a,*], A. van Griensven [a], N. Ray [c,d], L. Kokoszkiewicz [b], G.D. Betrie [a]

[a] UNESCO-IHE Institute for Water Education, Department of Hydroinformatics and Knowledge Management, PO Box 3015, 2601 DA, Delft, The Netherlands
[b] CERN – European Organization for Nuclear Research, Route de Meyrin 385, Genève, Switzerland
[c] University of Geneva, Climate Change and Climate Impacts, EnviroSPACE laboratory, Battelle, Building D, 7 route de Drize, CH-1227 Carouge, Switzerland
[d] United Nations Environment Programme, Division of Early Warning and Assessment, Global Resource Information Database – Europe, International Environment House, 11 Chemin des Anémones, CH-1219 Châtelaine, Switzerland

ARTICLE INFO

Article history:
Received 2 March 2011
Received in revised form
31 July 2012
Accepted 6 August 2012
Available online 3 September 2012

Keywords:
Distributed computing
Grid computing
Hydrological models
SWAT
Gridification

ABSTRACT

The increasing interest in larger spatial and temporal scale models and high resolution input data processing comes at a price of higher computational demand. This price is evidently even higher when common modeling routines such as calibration and uncertainty analysis are involved. Likewise, methods and techniques for reducing computation time in large scale socio-environmental modeling software is growing. Recent advancements in distributed computing such as Grid infrastructure have provided further opportunity to this effort. In the interest of gaining computational efficiency, we developed generic tools and techniques for enabling the Soil and Water Assessment Tool (SWAT) model application to run on the EGEE (Enabling Grids for E-science projects in Europe) Grid. Various program components/scripts were written to split a large scale hydrological model of the Soil and Water Assessment Tool (SWAT), to submit the split models to the Grid, and to collect and merge results into single output format. A three-step procedure was applied to take advantage of the Grid. Firstly, a python script was run in order to split the SWAT model into several sub-models. Then, individual sub-models were submitted in parallel for execution on the Grid. Finally, the outputs of the sub-basins were collected and the reach routing process was performed with another script executing a modified SWAT program. We conducted experimental simulations with multiple temporal and spatial scale hydrological models on the Grid infrastructure. Results showed that, in spite of computing overheads, parallel computation of socio-environmental models on the Grid is beneficial for model applications especially with large spatial and temporal scales. In the end, we conclude by recommending methods for further reducing computational overheads while running large scale model applications on the Grid.

Software availability

The codes and tools developed in this research are freely available through the GNU general public license (*http://www.gnu.org/copyleft/gpl.html*) for the SWAT user community as well as for the general public. They can be accessed through the following ftp link: *ftp://ftp.ihe.nl/SWAT-GRID/* with user name and password combinations of *ftpguest* and *fi3tsb3l*, respectively.

1. Introduction

The availability of increasingly higher resolution input data for socio-environmental models is certain to be computationally demanding, especially for large spatial and temporal scale models. However, the effort on devising methods or techniques of reducing computation time in large scale socio-environmental modeling software is growing likewise (Bryan, 2012; Fernández-Quiruelas et al., 2011; Goodall et al., 2011; Jeffery, 2007; Mineter et al., 2003; Sulis, 2009). The endeavor being carried out by the EU/FP7 funded 'EnviroGRIDS@Black Sea Basin' project (hereafter, enviroGRIDS; http://www.envirogrids.net) attempts to enable large-scale environmental and hydrological models to execute and deliver results at near real-time. The project aims at building capacities in the Black Sea region to use new international standards to gather, store, distribute, analyze, visualize and disseminate crucial information on past, present and future states of this region based on Grid infrastructures.

Grid computing originated in the 1990s as a metaphor for making computer power as easy to access as an electric power grid (Berman et al., 2003; Foster, 2003; Foster and Kesselman, 2004). As such, a computing Grid is a distributed system that supports a virtual research environment across different institutions (Chen

* Corresponding author. Tel.: +31 681199214.
E-mail addresses: seleshiget@gmail.com, s.yalew@unesco-ihe.org (S. Yalew).

1364-8152/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.envsoft.2012.08.002

et al., 2009; Schwiegelshohn et al., 2010). Another, task-oriented, definition of computing Grid is that it is a group of loosely coupled computers acting in concert to execute very large tasks. Grid computing has advanced rapidly to take an appeal by major institutes and projects worldwide. In recent years, Grids have emerged as wide-scale distributed infrastructures that support the sharing of geographically distributed, heterogeneous computing and storage resources (Tsouloupas and Dikaiakos, 2007). Scientific areas like particle physics, bioinformatics, nanotechnology, meteorology, and flow mechanics are few among research disciplines benefitting from such infrastructures to run computationally demanding workflows (Fernández-Quiruelas et al., 2011; Huang and Chang, 2003; Wegener et al., 2009).

Coordinated by CERN (the European Organization for Nuclear Research) and funded by the European Commission through a series of projects called EGEE (Enabling Grids for E-sciencE projects I, II and III), the Worldwide LHC Computing Grid (WLCG) is currently the largest multi-science computing Grid. As of this writing, the WLCG links about 260 resource centers in 55 countries, with more than 150,000 CPU cores, 28 PB of disk storage, 38 PB of tape storage, more than 14,000 registered users, and more than 300,000 jobs per day. The WLCG initially focused on two well-defined application areas, Particle Physics and Life Sciences mainly due to the fact that these communities were already Grid aware and ready to deploy challenging real applications at the beginning of the project. A range of applications is currently supported on the WLCG, and hydrological and environmental models are no exception (Fernández-Quiruelas et al., 2011; Lagouvardos et al., 2010; Lecca et al., 2011).

The WLCG, like other Grid infrastructures, has its own middleware and access protocols. It demands the software to be run on its environment to be developed in specified (parallelized) formats for better computational efficiency on the Grid. Whereas new software tools might be developed with these requirements in mind, legacy scientific codes or existing applications (the majority of which are not grid-aware yet) often require major code restructuring before they can run on the Grid. The process of porting existing code to the Grid is termed as "Gridification" by the Grid user community. A "gridified" application can run on interoperable Grids. A gridified application is aware of communication protocols for submitting, starting and finishing jobs on the Grid.

For example, a gridified data analysis application will be able to:

- Obtain necessary authentication credentials to open files it needs.
- Query a catalog to determine where the files are and which grid resources are able to do the analysis.
- Submit requests to the grid: asks to extract data, initiate computations, and provide results.
- Monitor progress of the various computations and data transfers, notify the user when analysis is complete, and detecting and responding to failures (collective services), and
- Gather all results and possibly merge output files to obtain the final sets of result files.

Although institutional computing clusters can be used in certain occasions to tackle computational challenges, for instance (Whittaker, 2004), access to these clusters is generally very limited for the user community. For widespread access to a very large computational pool of computers, Grid computing is a foreseeable distributed computing alternative for the scientific community. This study discusses development of generic methods, tools and techniques for parallelization of large scale hydrological models in general, and SWAT models in particular, on a Grid infrastructure. The study demonstrates experimental simulations using multiple spatial and temporal scale models on the WLCG infrastructure. With the objective of achieving a near-real-time computation on large scale spatially distributed SWAT models, the study presents formulation and test results of various approaches.

2. Materials and methods

2.1. The Soil and Water Assessment Tool (SWAT)

The Soil and Water Assessment Tool (SWAT)(Arnold et al., 1998; Neitsch et al., 2005) is an increasingly popular watershed modeling software. It is a physically-based model that can simulate water quality and quantity at a watershed scale. The spatial scales of the problems analyzed with this modeling software have increased to a remarkably large degree over time. The whole of continental USA (Arnold et al., 1999) and the whole of Africa (Schuol et al., 2008) were modeled on separate occasions using SWAT with commendable results. However, as model extents and input data resolutions continue to grow, computation time becomes a bottle-neck especially for repetitive model simulations.

SWAT can compute continuously on a daily or sub-daily time step. The model is primarily designed to predict impacts of management on water, sediment, and agricultural chemical yield in gauged and un-gauged basins (Arnold et al., 1998). The major modules in the model include hydrology, erosion/sedimentation, plant growth, nutrients, pesticides, land management, stream routing, and pond/reservoir routing. Climate input files (precipitation, maximum and minimum temperature, relative humidity, wind speed, solar radiation) are inputs on a daily temporal resolution, although recent versions of the model allow hourly input files.

A watershed modeled using SWAT is partitioned into different required and/or optional objects of subunits such as sub-basins, reaches/main channel segments, impoundments/reservoirs on the main channel network and point sources. As such, a watershed is subdivided into sub-basins which are, thus, the first level of the subdivision. Sub-basins are defined by geographical positions in the watershed and are spatially related to one another (Neitsch et al., 2011). All sub-basins drain into river networks where water is routed from upstream to downstream reaches. The land area in a sub-basin may be divided into HRUs (Hydrologic Response Units). A sub-basin must contain at least one HRU and a main channel or reach. HRUs are portions of a sub-basin that possess unique land use, management, or soil attributes (Neitsch et al., 2011). SWAT setup provides options to choose HRU distributions in a watershed or sub-basin. Theoretically, one or as many as unlimited number of HRUs may be populated in a single sub-basin with unique land use/soil and management combinations. Unlike in the case of sub-basins, no spatial relationship or interaction can be specified among HRUs. Sediment, chemicals or nutrient loadings from each HRU are computed independently and then summed up to determine the total load from a sub-basin. (Note that defining more HRUs per sub-basin implies a higher spatial resolution.)

A watershed model should also incorporate one reach or main channel associated with each sub-basin. This channel carries loadings from the sub-basin or outflow from the upstream reach segments into the downstream network of the watershed in the associated reach segment. An impoundment such as a pond or a wetland may also be defined in a sub-basin.

2.2. Parallelization methods

The parallelization of SWAT models for Grid execution is analyzed using various methods and approaches. The general methodology of this study can simply be stated as: divide–compute–merge. In other words, it follows a general procedure of dividing or splitting a large scale and/or high resolution model into several small scale model components and computing them all in parallel on the Grid. After each component finishes executing, outputs from each of them are collected and merged. Results are presented similar to that of the original, undivided model.

Before proceeding with the actual work of splitting and merging, though, various methodological questions should be addressed: a) On what level (e.g. sub-basin, HRU or reach level) can one split SWAT models for optimal efficiency on the Grid. b) Do we have to split all sub-basin specific information into independent sub-basin/HRU files or can we still maintain some common input files for all the sub-basins as they are in the original model? c) From which files do we retrieve relevant information for parallel computations of such sub-basin level splits? d) What is the advantage or disadvantage of splitting or maintaining those common input files? e) Which parts of the SWAT computing code modules are irrelevant/relevant during independent sub-basin computations?

Within a SWAT watershed model, sub-basin processes are computed independently from each other; and, within a sub-basin, HRUs are computed independently from each other and from other sub-basins as well. They are independent enough as a basis for parallelization of SWAT models on the Grid. However, an HRU might be too small to serve as a unit for splitting a big model into sub-components. All HRUs within a sub-basin share the same weather input files (such as precipitation, temperature, humidity, solar radiation, and wind speed) which can be the largest model input files in SWAT. An HRU level splitting and merging, thus, would require the split of these big input files, a practice certain to be computationally costly. In

this study, HRU level parallelization was left out for speculative reasons that it may increase the simulation overhead during splitting, merging and communication on the Grid. On the other hand, the computation of a reach routing/loading depends on outputs of the upstream reach and also of the associated sub-basin. In other terms, downstream reaches incorporate loads from upstream reaches and also from sub-basin processes associated with them, and thus are dependent on them. This kind of dependency is sequential by design, a downstream reach may not proceed computing until it gets initial conditions from the upstream. For the reason that reach routing processes are highly interdependent in a SWAT watershed model, they are disregarded as a basis of parallelization unit. Splitting at sub-basin level, thus, seems the more viable option because sub-basins in SWAT are neither too interdependent nor too small to serve as a unit of computation.

SWAT takes a number of required and optional input data about physical (elevation, land cover, soil), weather (rainfall, air temperature, solar radiation, wind speed, relative humidity), hydrological (stream flow, sediment transport, nutrient loads), and point and non-point sources of pollution in a watershed. It accepts model input data in three categories or levels of detail: watershed, sub-basin, and HRU. These input data are stored in a number of files and databases. Some files carry HRU level data (e.g. *.hru, *.mgt, *.gw, *.sol, *.chm, etc files), while some others carry data related to sub-basins (e.g. *.sub, *.wgn, *.rte, *.swq, etc) and still others carry watershed level data (e.g. the configuration (fig.fig), the master watershed (file.cio), precipitation (*.pcp), temperature (*.tmp), etc files). For parallel simulation of the sub-basin processes, relevant data (for instance, whether a sub-basin is upstream or downstream from the 'fig.fig' file; number of simulation years, number and name of HRU files associated with the sub-basin from the master watershed 'file.cio' file, etc) are gathered for each sub-basin. Then, new configuration file is created for each sub-basin in a unique directory where other sub-basin relevant files are copied into. The watershed configuration file in Fig. 1(a) is split according to Fig. 2(a) for each sub-basin in the first approach (Approach I) and according to Fig. 2(b) in the second approach (Approach II). The first approach (Approach I) is that only the sub-basin processes are run in parallel without the reach routing process, whereas the second approach computes upstream reaches together with sub-basin processes. Unlike the first approach, the second approach also executes merging of upstream sub-basin results right after they finish computing rather than wait until every sub-basin (including the ones in the downstream) finishes computing. Approach I follows the principle of *split, compute all sub-basin processes, then start routing the reaches*. On the other hand Approach II follows the principle of *split, compute all sub-basin processes and, at the same time, route and merge upstream reaches/sub-basins, and then, route and merge the downstream ones*.

For the SWAT model splitting process in this study, file duplication was preferred to extreme file split. Short and computationally lightweight files are split to each sub-basin whereas long record and heavy files are copied to each sub-basin directory for processing. Since multiple file access and file processing (for example for files such as precipitation and temperature which can be too heavy/long and too many) take more time at a later stage when each sub-basin wants to use one, these files are split for each sub-basin. The downside of duplicating files to each sub-basin is storage space consumption. However, with Petabytes of space on the Grid, the demand for storage can be argued to be of smaller concern compared to the demand for speedup.

Watershed configuration in SWAT models is stored in a file called "fig.fig". This file consists of a number of "command lines" for each time step to be computed in sequence. Among the most important command lines are "sub-basin" that computes sub-basin processes, "route" that computes processes in a reach and "add" that sums outputs from previous command lines. A typical structure of the 'fig.fig' file is given in Fig. 1a, where the first block represent lines for the computations of the sub-basins (7 in total), followed by a block with several "route" and "add" commands. The latter represents the river network.

In the original SWAT code, the sequence of computation is as in Fig. 3(a), all sub-basins, and all HRUs within sub-basins, are computed first, followed by main channel networks/reaches. These computations are iterated for every day of every year for the stated simulation period. Two approaches/configurations of parallelization are devised in this study. In a first parallelization configuration (Approach I), all sub-basins are computed in parallel followed by all reaches in series ("Splitter1" in Fig. 3b). In a second configuration (Approach II) ("Splitter2" in Fig. 3c), all sub-basins and HRUs within sub-basins are computed first. Then, reaches of upstream sub-basins are computed and merged in parallel to each other after their associated sub-basins are computed, followed by the ones that depend on upstream reaches. Approach I, therefore, is concerned only with parallelization of sub-basin processes without affecting the sequential design of reach executions whereas Approach II executes upstream reaches (reaches which do not depend on outputs of other reaches) in parallel to each other with the associated sub-basin processes and merge the results of which right after they finish executing.

2.3. Computing environment

Computing environment refers to the hardware, operating system and software tools involving the parallelization experiment. Whereas the codes and tools are tested only on the platforms mentioned here, they might as well be used in other computing environments.

2.3.1. Hardware and operating system

Non-parallel runs were tested on AMD Phenom 9600B quad-core 2.3 Ghz machine with 4 GB operating memory. The testing machines used for these computations are run on Ubuntu 10.04 operating system whereas the Grid nodes use distributions of Scientific Linux SLC4 and SLC5.

2.3.2. Software

In total, 6 existing as well as newly developed software programs were used:

(1) **SWAT2005**: is the original SWAT model code and it is used for computing the sub-basins (as well as for routing of upstream reaches in the second configuration).

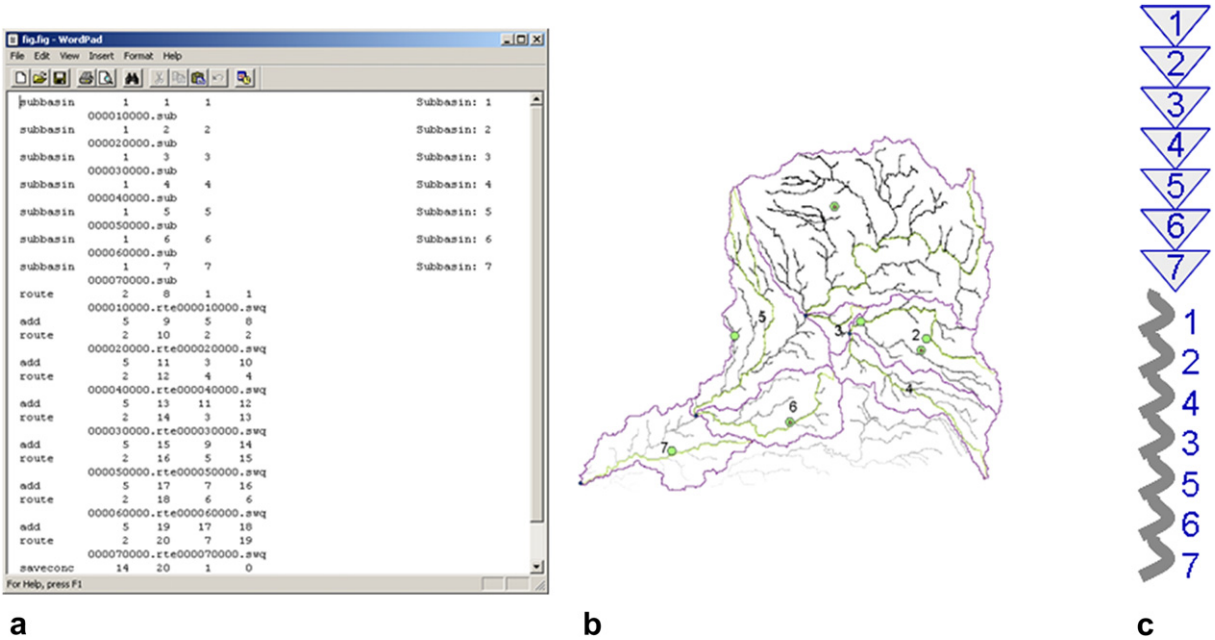

Fig. 1. The "fig.fig" file (a) for the SWAT model of the Nzoia catchment (Kenya) (b) and the corresponding conceptual schematization for this file (c). Triangles represent a sub-basin and the arcs represent reaches (c).

```
subbasin        1     1     1
          000010000.sub
save            9     1     1     0
          watout1.dat
finish          0
```

a

```
subbasin        1     1     1
          000010000.sub
route           2     2     1     1
          000010000.rte000010000.swq
save            9     2     1     0
          watout1.dat
finish          0
```

b

Fig. 2. (a) The new configuration file for a single sub-basin without the routing and (b) with the routing. Note that the parallel routing configuration, Fig. 2(b), applies only for upstream sub-basins/reaches.

(2) **SWATgrid**: is an adapted version of the SWAT model that reads output files from the sub-basins run on the Grid, and executes the add/route commands to compute the reach loadings. The difference between the original SWAT code and this adapted version is that the adapted version is modified to block/disable sub-basin process modules/computations and executes only the routing of stream-networks.

(3) **Splitter1** and **Splitter2**: are python/java programs written for splitting generic SWAT models into sub-basin scale model components. A number of files need to be adapted for the model splitting purpose:
- fig.fig: this file is reduced to one command line for sub-basin computation and one to save outputs to a file (Fig. 2a) in case of **Splitter1**. In the case of **Splitter2**, it is reduced to one command line, one routing line, and one to save outputs to a file (Fig. 2b).
- Precipitation (*.pcp), temperature (*.tmp) and other weather files are split to contain data specific only to the associated sub-basin.
- Input files that describe or represent a sub-basin, including the SWAT executable program **SWAT2005**, are copied to a directory created for each sub-basin.

(4) **SWATmerger.exe**: is a (python/java) program written for merging sub-basin computation results. It first initiates the SWATgrid program to compute the reach routing. Major tasks of this program are to collect outputs from individual sub-basin computations returned from each Grid node and to organize them as if they were executed from a single machine.

(5) **Ganga**: is an existing python script (Moscicki et al., 2009) that serves as a user interface for specifying and submitting jobs to a grid; it connects to a Grid, submits files to Grid machines and copies results back to the PC of the user.

(6) **Diane**: is an existing lightweight distributed framework for parallel scientific applications in master–worker model (Moscicki, 2003). It is used for an automatic control and scheduling of computations on a set of distributed worker nodes. It boosts execution efficiency, reduces work overhead through automatic failure management, and integrates local and Grid resources. See Fig. 4 for sequences of executions of the various tools/codes.

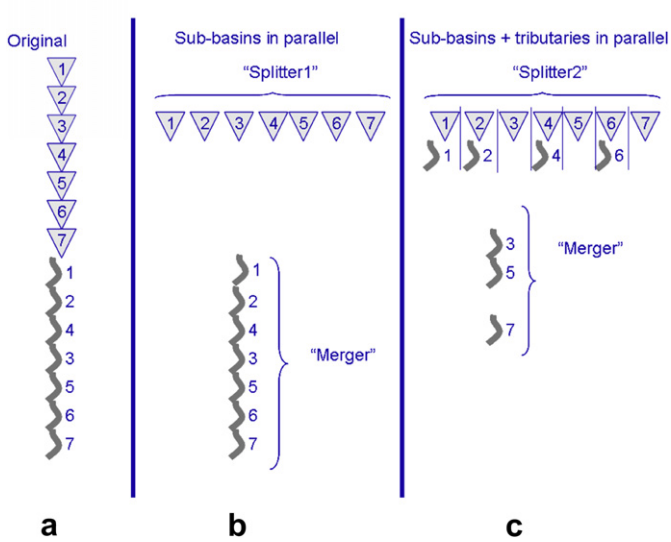

Fig. 3. The original configuration "sequence" (a), splitting methods "Splitter1" (b) and "Splitter2" (c).

3. Test models: setup and job submission

Three separate SWAT models were used for demonstration of the parallelization process: the Nzoia catchment model in Kenya, the Lake Balaton catchment model in Central Europe (Hungary) and the trans-boundary Danube catchment model in the Black Sea Basin (Fig. 5). The Nzoia catchment model, presented in Fig. 1, includes 7 sub-basins and contains data to simulate 43 years. The Lake Balaton catchment model contains 204 sub-basins with a simulation period of 16 years whereas the Danube River Basin model contains 423 sub-basins.

For the sake of conducting experiment with differences in simulation periods, the Danube model was built with 5, 16 and 38 years of simulation periods, selected at random. Furthermore, simulations of single and multiple HRUs per sub-basin were experimented with each model. After these models where setup, each of them where submitted to and managed on the Grid infrastructure through the ganga and DIANE software tools.

Job submission using ganga involves scripts for specifying properties of the job, such as whether job is an executable or not, whether it is supposed to run locally or on a grid, and on which back-end (infrastructure) it is supposed to be run, etc. Once these properties are stated as shown in Fig. 6, one more simple command [j.submit()] will be able to send the specified job to the EGEE/LCG grid back-end.

The same way to the use of ganga, a piece of script must be written to instruct the DIANE framework to manage jobs submitted on the EGEE/LCG grid back-end. This script involves the description to DIANE of what will be run, that is, executable or not, the number of jobs, and the number of worker agents to carry out the computation management. A sample DIANE script is shown in Fig. 7. The same script (saved as 'swat.sh' in this case) can be run on the DIANE command line interface using the bash input executable variable:

$ diane-run swat.sh

The above command triggers DIANE master to check-connection, prepare and copy input files to storage resources and wait for another command from ganga to specify the job submitter and the number of DIANE worker agents which the next command will do:

>ganga LocalSubmitter.py –diane-worker-number=4

This command will submit jobs to the Grid using ganga. The four (4) worker agents, in this case, of DIANE will automatically split the sub-basins submitted as jobs between each other and manage their execution. The whole process, e.g., whether a job succeeds in executing or fails to do so, can be monitored from the ganga interface.

4. Results and discussion

First, each of the models were setup with single HRU per sub-basin and tested computation time both with Approach I (where

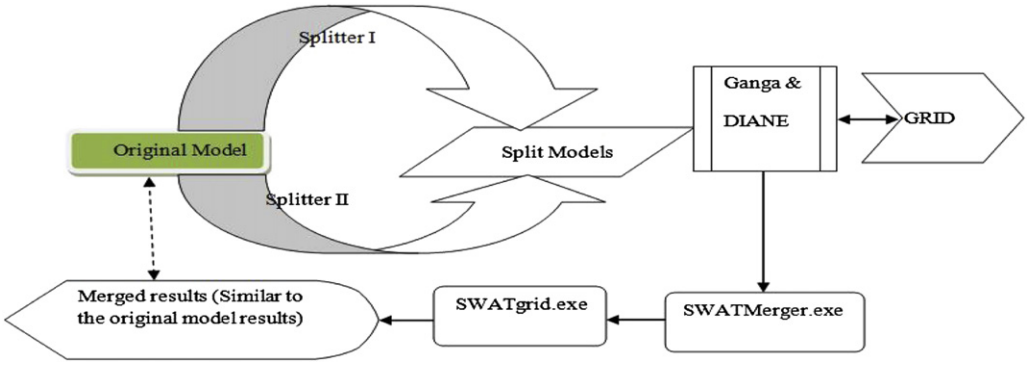

Fig. 4. Computation sequence of the various software tools.

Fig. 5. The Danube River basin.

all sub-basins are computed in parallel followed by all reaches in series) and with Approach II (where all sub-basins and only upstream reaches are computed in parallel, followed by downstream reaches). As shown in Table 1, the Nzoia catchment shows a speedup ratio of 2.96 in computation (which means that this execution setup is 2.96 faster than the original model if it was run on a personal computer). The model for the Balaton Lake basin gained a speedup of 1.4, whereas that of the Danube basin model (with 5 years of simulation period) achieved a speedup of 1.04.

Review of results in Table 1 show that Approach I performed worse, with the exception of the Nzoia catchment model, than the original model setting which is run on a personal computer (PC). This is mainly attributed to the large computing overhead involved in this approach. Approach II registers gains in computing for all the three models, due to the reduction in overhead (and in spite of increases in splitting and sub-basin execution time). Overhead, here, is considered to be the wasted time from the moment the model starts splitting to the time the last part of the model is returned from the Grid.

On a second experiment, the three models were setup again with multiple HRUs per sub-basin, other settings being the same. As demonstrated by test results in Table 2, the Nzoia catchment

```
In[1]:j=Job()
In[2]:j.name="SWAT"
In[3]:j.application=Executable()
In[4]:j.application.exe=File('runswat.sh')
In[5]:j.inputsandbox=[File('txtinout.tgz'),File('swat2009')]
In[6]:j.outputsandbox=['txtinout.tgz']
In[7]:j.backend=LCG()
```

Fig. 6. A job specification script on ganga.

```
# tell DIANE that we are just running executables
# the ExecutableApplication module is a standard DIANE test application

from diane_test_applications import ExecutableApplication as application

# the run function is called when the master is started

# input.data stands for run parameters

default_master = """
endPoint = giop:tcp::22246
"""
def run(input, config):
      d = input.data.task_defaults
      d.input_files = ['swat.sh','txtinout.tgz']
      d.output_files = ['txtinout.tgz']
      d.executable = 'swat.sh'

      # tasks which differ by arguments with the executable

      for i in range(1, 8):
            t = input.data.newTask()
            t.args = [str(i)]
```

Fig. 7. Job specification script for DIANE.

model gained a speedup of 6.2. The Lake Balaton and the Danube basin models achieved a speedup of 4.9 and 4.5, respectively. In this scenario, both Approach I and Approach II performed better than the original model setting for all the models, with Approach II giving the largest gain in computation for all the models.

A final experiment was conducted by varying simulation years for the Danube basin model. Three different simulation periods (5 years, 16 years, and 38 years) where selected at random and the model was setup and tested for each case with both single as well as multiple HRUs per sub-basin. Because of its apparent computing advantage from the previous experiments, this time the models were tested using only Approach II. Results (Table 3) show that the Danube basin model gained a speedup of 1.04 (also shown in Table 2) for 5 years simulation period, a speedup of 3.3 for the 16 years simulation period and a speedup of 4.7 for the 38 years simulation period when run with single HRU per sub-basin. When multiple HRUs per sub-basin are used, the same model shows speedups of 4.6, 6.8, and 7.7 for the 5 year, 16 year and 38 year simulation periods, respectively (Table 4).

The experimental results for nearly all cases show that computing on the Grid results in considerable gain in computation time. Approach II, which follows the principle of – *split, execute*–

Table 1
Computation times for the three models with single HRU per sub-basin.

Model	# of sub-basin	Sim. period (yrs)	HRUs per sub-basin	Computation time (s)											Spd (b)[a]
				Orig. model	Approach I[a]					Approach II[a]					
					Spl	SbE	Mr	OvH	Tot	Spl	SbE	Mr	OvH	Tot	
Nzoia	7	43	Single	32	1.2	1.4	2.2	6	10.8	1.4	1.6	1.8	4	10.8	2.96 (App II)
Balaton	204	16	Single	1355 (22.6 min)	62.7	1.2	2.1	2860 (47.7 min)	2926 (48.8 min)	63.1	1.4	1.3	902 (15 min)	968 (16.1 min)	1.4 (App II)
Danube	423	5	Single	1335 (22.25 min)	341 (5.7 min)	1	4	2781 (46.4 min)	3127 (52.1 min)	343 (5.72 min)	1.3	1.4	935 (15.6 min)	1283 (21.4 min)	1.04 (App II)

[a] Spl = Splitting, SbE = Sub-basin Execution, Mr = Merging, OvH = Overhead, Tot = Total time, Spd(b) = the better speedup of the two approaches, min = minutes, App II = Approach II.

Table 2
Computation times for the three models with multiple HRUs per sub-basin.

Model	# of sub-basin	Sim. period (yrs)	HRUs per sub-basin	Computation time (s)											Spd (b)[a]
				Orig. model	Approach I[a]					Approach II[a]					
					Spl	SbE	Mr	OvH	Tot	Spl	SbE	Mr	OvH	Tot	
Nzoia	7	43	Mult	2671 (44.5 min)	19.3	322	58	130.7	530 (8.85 min)	26	356	28	20	430 (7.1 min)	6.2 (App II)
Balaton	204	16	Mult	8388 (2.33 h)	398 (6.6 min)	311 (5.2 min)	87	2874 (47.9 min)	3670 (1.02 h)	403 (6.7 min)	347 (5.8 min)	19	958 (16 min)	1727 (28.8 min)	4.9 (App II)
Danube	423	5	Mult	8244 (2.29 h)	414 (6.9 min)	387 (6.45 min)	109	2995 (49.9 min)	3905 (1.08 h)	431 (7.2 min)	399 (6.65 min)	13	972 (16.2 min)	1815 (30.3 min)	4.5 (App II)

[a] Spl = splitting, SbE = sub-basin execution, Mr = merging, OvH = overhead, Tot = total time, Mult = multiple HRUs per sub-basin, Spd(b) = the better speedup of the two approaches, min = minutes, h = hours, App II = Approach II.

Table 3
Single HRU computation times for the Danube basin model using Approach II.

Model	# of sub-basin	Sim. period (yrs)	HRUs per sub-basin	Computation time (s)						Speedup
				Orig. model	Approach II[a]					
					Spl	SbE	Mr	OvH	Tot	
Danube	423	5	Single	1335 (22.3 min)	343 (5.72 min)	1.3	1.4	935 (15.6 min)	1283 (21.4 min)	1.04
		16	Single	11,141 (3.09 h)	364 (6.1 min)	4	8	2956 (49.3 min)	3332 (55.5 min)	3.3
		38	Single	15,532 (4.3 h)	388 (6.6 min)	5	16	2888 (48.2 min)	3297 (54.2 min)	4.7

[a] Spl = splitting, SbE = sub-basin execution, Mr = merging, OvH = overhead, Tot = total time, Mult = multiple HRUs per sub-basin, min = minutes, h = hours.

Table 4
Multiple HRU Computation times for the Danube basin model using Approach II.

Model	# of sub-basin	Sim. period (yrs)	HRUs per sub-basin	Computation time (s)						Speedup
				Orig. model	Approach II[a]					
					Spl	SbE	Mr	OvH	Tot	
Danube	423	5	Mult	8244 (2.29 h)	431 (7.2 min)	399 (6.65 min)	13	942 (15.7 min)	1785 (29.75 min)	4.6
		16	Mult	18,720 (5.2 h)	1092 (18.2 min)	678 (11.3 min)	19	960 (16 min)	2749 (43.7 min)	6.8
		38	Mult	28,512 (7.92 h)	1782 (29.7 min)	948 (15.8 min)	24	954 (15.9 min)	3708 (1.03 h)	7.7

[a] Spl = splitting, SbE = sub-basin execution, Mr = merging, OvH = overhead, Tot = total time, Mult = multiple HRUs per sub-basin, min = minutes, h = hours.

route–merge upstream, and then *route–merge* downstream – has a clear computing advantage over Approach I mainly due to considerable reduction in computing overhead. This is true even when splitting and average sub-basin execution times are higher in this approach.

In spite of reducing overhead, Approach II increases splitting and average sub-basin execution times. This is due to the fact that while Approach I does not split routing related files and commands, Approach II does for every sub-basin. This increases splitting time in Approach II for all the cases, however by a nominal margin. The same procedure also increases average sub-basin computation time because upstream sub-basins have to execute routing commands in addition to other sub-basin processes carried out in Approach I. Merging time is reduced in Approach II because of the fact that upstream sub-basins have already been merged on a one-step procedure of *execute-and-merge*. What remains at the end of all sub-basin execution, therefore, is merging the downstream sub-basin results, which takes less time than merging all sub-basin results.

Overall, lighter and smaller jobs (such as Nzoia, single HRU per sub-basin) seem to get a higher priority on the Grid computing nodes. On such cases, we saw that the average Grid computing overhead falls to few seconds irrelevant of whichever of the two approaches are used. A solid gain in computing is demonstrated when larger (in model extent or simulation length) and high resolution (e.g. multiple HRU per sub-basin) models are run on the Grid. Since model splitting is a one-time procedure, the experimental speedup results demonstrated so far can get even better with some modeling procedures such as sensitivity and uncertainty analysis efforts on the Grid which often require repetitive model simulations.

5. Conclusion

The 'gridification' of the Soil and Water Assessment Tool demonstrates the use of emerging Grid technologies and infrastructures for hydrological and environmental modeling purposes. Experimental results show a remarkable improvement of performance in computation time. Large scale and time intensive hydrological/environmental models with higher resolution (e.g. multiple HRUs per sub-basin) and longer simulation periods make the most of the Grid computing infrastructure. In addition to reducing computation time, computing on the Grid also relieves computer memory problems that often originate from running large scale SWAT models on a personal computer. The major bottleneck to even further gain in computation time is the overhead during model splitting, communication between various computing nodes, and merging of the different model result components. However, various techniques and approaches followed in this study (e.g. Approach II) to Grid job submission, routing and model merging prove that computing overhead on the Grid can be considerably reduced to an acceptable degree. The authors believe that this research, as well as the tools developed through it, can be used as a basis for the progress towards the use of existing as well as emerging high performance computing technologies and infrastructures such as the Grid to time-intensive socio-environmental models.

Acknowledgments

The authors would like to acknowledge the European Commission "Seventh Framework Programme" that funded the enviroGRIDS project (Grant Agreement n° 227640). In addition, heartily thank you goes to the editor and the anonymous reviewers of this manuscript for their meticulous analysis and feedback during the peer-review process.

References

Arnold, J.G., Srinivasan, R., Muttiah, R.S., Williams, J., 1998. Large area hydrologic modeling and assessment part I: model development. JAWRA Journal of the American Water Resources Association 34, 73–89.
Arnold, J.G., Srinivasan, R., Muttiah, R.S., Allen, P.M., 1999. Continental scale simulation of the hydrologic balance1. Journal of the American Water Resources Association 35, 1037–1051.
Berman, F., Fox, G., Hey, A.J.G., 2003. Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc..
Bryan, B.A., 2012. High-performance computing tools for the integrated assessment and modelling of social–ecological systems. Environmental Modelling & Software. http://dx.doi.org/10.1016/j.envsoft.2012.02.006.
Chen, A., Di, L., Wei, Y., Bai, Y., Liu, Y., 2009. Use of grid computing for modeling virtual geospatial products. International Journal of Geographical Information Science 23, 581–604.
Fernández-Quiruelas, V., Fernández, J., Cofiṇo, A., Fita, L., Gutiérrez, J.M., 2011. Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model. Environmental Modelling & Software 26 (9), 1057–1069.
Foster, I., Kesselman, C., 2004. The Grid: Blueprint for a New Computing Infrastructure Morgan Kaufmann.
Foster, I., 2003. The Grid: a New Infrastructure for 21st Century Science. Wiley Online Library, pp. 51–63.

Goodall, J.L., Robinson, B.F., Castronova, A.M., 2011. Modeling water resource systems using a service-oriented computing paradigm. Environmental Modelling & Software 26 (5), 573–582.
Huang, G., Chang, N., 2003. The perspectives of environmental informatics and systems analysis. Journal of Environmental Informatics 1, 1–7.
Jeffery, K.G., 2007. Next generation GRIDs for environmental science. Environmental Modelling & Software 22, 281–287.
Lagouvardos, K., Floros, E., Kotroni, V., 2010. A grid-enabled regional-scale ensemble forecasting system in the mediterranean area. Journal of Grid Computing 8, 181–197.
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 (1–2), 186–199.
Mineter, M.J., Jarvis, C., Dowers, S., 2003. From stand-alone programs towards grid-aware services and components: a case study in agricultural modelling with interpolated climate data. Environmental Modelling & Software 18, 379–391.
Moscicki, J., Brochu, F., Ebke, J., Egede, U., Elmsheuser, J., Harrison, K., Jones, R., Lee, H., Liko, D., Maier, A., 2009. Ganga: a tool for computational-task management and easy access to grid resources. Computer Physics Communications 180, 2303–2316.
Moscicki, J.T., 2003. Diane-Distributed Analysis Environment for Grid-Enabled Simulation and Analysis of Physics Data, vol. 3. IEEE, pp. 1617–1620.
Neitsch, S., Arnold, J., Kiniry, J., Williams, J., 2005. Soil and Water Assessment Tool Theoretical Documentation. Version 2005. Temple, TX. USDA Agricultural Research Service and Texas A&M Blackland Research Center.
Neitsch, S.L., Arnold, J.G., Kiniry, J., Williams, J., 2011. SWAT Model User's Manual. Texas A&M University, Texas.
Schuol, J., Abbaspour, K.C., Yang, H., Srinivasan, R., Zehnder, A.J.B., 2008. Modeling blue and green water availability in Africa. Water Resources Research 44, W07406.
Schwiegelshohn, U., Badia, R.M., Bubak, M., Danelutto, M., Dustdar, S., Gagliardi, F., Geiger, A., Hluchy, L., Kranzlmüller, D., Laure, E., 2010. Perspectives on grid computing. Future Generation Computer Systems 26, 1104–1115.
Sulis, A., 2009. GRID computing approach for multireservoir operating rules with uncertainty. Environmental Modelling & Software 24, 859–864.
Tsouloupas, G., Dikaiakos, M.D., 2007. GridBench: a tool for the interactive performance exploration of grid infrastructures. Journal of Parallel and Distributed Computing 67, 1029–1045.
Wegener, D., Sengstag, T., Sfakianakis, S., Rüping, S., Assi, A., 2009. GridR: an R-based tool for scientific data analysis in grid environments. Future Generation Computer Systems 25, 481–488.
Whittaker, G., 2004. Use of a Beowulf cluster for estimation of risk using SWAT. Agronomy Journal 96 (5), 1495–1497.