Geophysical Research Abstracts
Vol. 12, EGU2010-9894, 2010
EGU General Assembly 2010

Concepts to accelerate water balance model computation

Oliver Gronz (1,2), Markus Casper (1), and Peter Gemmar (2)

(1) University of Trier, Department of Physical Geography, Trier, Germany (gronz@uni-trier.de, +49 651 201 3976), (2) University of Applied Sciences Trier, Institute for Innovative Informatics Applications, Department of Computer Science, Trier, Germany

Computation time of water balance models has decreased with the increasing performance of CPUs within the last decades. Often, these advantages have been used to enhance the models, e. g. by enlarging spatial resolution or by using smaller simulation time steps. During the last few years, CPU development tended to focus on strong multi core concepts rather than "simply being generally faster". Additionally, computer clusters or even computer clouds have become much more commonly available. All these facts again extend our degrees of freedom in simulating water balance models – if the models are able to efficiently use the computer infrastructure. In the following, we present concepts to optimize especially repeated runs and we generally discuss concepts of parallel computing opportunities.

Surveyed model
In our examinations, we focused on the water balance model LARSIM. In this model, the catchment is subdivided into elements, each of which representing a certain section of a river and its contributory area. Each element is again subdivided into single compartments of homogeneous land use. During the simulation, the relevant hydrological processes are simulated individually for each compartment. The simulated runoff of all compartments leads into the river channel of the corresponding element. Finally, channel routing is simulated for all elements.

Optimizing repeated runs
During a typical simulation, several input files have to be read before simulation starts: the model structure, the initial model state and meteorological input files. Furthermore, some calculations have to be solved, like interpolating meteorological values. Thus, e. g. the application of Monte Carlo methods will typically use the following algorithm: 1) choose parameters, 2) set parameters in control files, 3) run model, 4) save result, 5) repeat from step 1. Obviously, the third step always includes the previously mentioned steps of reading and preprocessing. Consequently, the model can be optimized for repeated runs as follows: 1) read input files, 2) preprocess input files, 3) store initial state internally, 4) choose and set parameters, 5) read initial state internally, 6) simulate water balance, 7) save result, 8) repeat from step 4. Using this approach, the preprocessing is not performed several times without any changes, which saves time and input processes.

Parallelization
Resulting from the subdivision of the catchment, the following nested loops are computed during repeated simulation runs: for all parameter sets: for all elements: for all compartments: simulate hydrological processes; and again for all elements: simulate channel routing. Some of these computations are obviously independent from each other, e. g. one realization with a specific parameter set does not influence a second realization using a different parameter set; hydrological processes in different compartments and elements do also not interact and can thus be computed parallel, too. One feasible solution in MATLAB is to spread different realizations to different nodes, compute different elements parallel using the Parallel Computing Toolbox and use MATLAB's vector arithmetic operations to compute all compartments efficiently. Finally, the channel routing remains as a problem to be solved element by element, as the discharge of upriver elements is used in downriver elements. But catchments are usually branched and not a line of subsequent elements. And different branches do not influence each other up to their confluence. Thus, we use binary trees to represent the connectivity of elements: the outlet element is the root, its predecessors are the root's children etc. Now, all nodes having the same depth are in

different branches and can be simulated parallel. The refined algorithm is: for all depths (beginning with tree's height): for all nodes of this depth: simulate channel routing. The inner loop can be computed parallel.

Results

The previously described concepts were implemented and tested using MATLAB, its Parallel Computing Toolbox, and a Windows HPC computer cluster (4 nodes, total of 16 cores). The concrete benefit strongly depends on the task to be solved: for single runs using not optimized input data files, the MATLAB version is even slower than its FORTRAN relative on single core computers with high performance hard disks. For repeated runs on several nodes with several cores, the computation time is 20 up to 50 times shorter with significantly reduced requirements concerning local hard disks.