*International Environmental Modelling and Software Society (iEMSs)*
*2012 International Congress on Environmental Modelling and Software*
*Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany*
*R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)*
*http://www.iemss.org/society/index.php/iemss-2012-proceedings*

# Parallel implementation to support large spatial simulations

**R. Wieland** [a], **W. Mirschel**[a] **and D. Deumlich**[a]

[a]*ZALF,Eberswalder Str. 84,15374 Muencheberg,Germany, (rwieland@zalf.de)*

**Abstract:** The paper introduces different approaches to parallelization based on OpenMPI and OpenMP applied to the Universal Soil Loss Equation (USLE). The USLE was used as a proxy for similar models from the “impact assessment toolbox”. The simulation of impact assessment takes into account climate change and changes in management. Even such a simple model as the USLE can lead to a time-consuming simulation when applied to a large region and when including stochastic data. The paper discusses the pros and cons of the implemented parallelization techniques. The key technology is to divide the simulation into two parts: a binary part implemented in C++ and an interpreter part which controls the parallel simulation written in Python. Python and its modules were also used to pre- and post-process the simulation.

***Keywords:*** Cluster computing, parallel simulation, Python, pypar, OpenMPI, OpenMP, climate change, impact assessment

## 1 INTRODUCTION

The German Renewable Energy Act, introduced in 2000, has changed agricultural production dramatically. Crops traditionally produced for agriculture now compete with crops grown for energy production. These changes in land use have led to an amendment to the act. An impact assessment group was set up at the Leibniz Center for Agricultural Landscape Research (ZALF) to support decision-makers with scientific expertise. This impact assessment group implemented a set of models to simulate different scenarios according to different land use distribution. Impact assessment has to cope with the problem that the investigation, and therefore the models, are available only for selected points, although the results ought to be valid for a larger region. A methodical overview of impact assessment is given in Payraudeau and van der Werf [2005]. The impact caused by land use change interacts with the impact of climate change. This paper was initiated by the discussion on current climate change and the adaptations (agro-management, land use) agriculture may introduce due to climate change [Wenkel et al., 2011].

The aim of this paper is to show how models can be implemented using a computing cluster to cope with large data sets. The erosion model, one of the impact assessment models, was selected to demonstrate the method. All other models use the same parallelization technique. The Federal State of Brandenburg, Germany ($29,479 km^2$ with a spatial resolution of 1 ha) was selected as the investigation area. All of the map information required for spatial simulations should also have a resolution of 1 ha. The map is organized as a matrix of 2339 x 2448 grid cells. Not all cells are filled. The soil map, for example, has 4,046,492 no-data cells and 1,679,380 cells with data. For the whole Federal State of Brandenburg, even a simple model such as the Universal Soil Loss Equation [Wischmeier and Smith, 1978] used to calculate long-term average amounts of eroded soil requires a long simulation time to iterate over all grid cells and 30 years.

Parallelization using a cluster computer with many cores can reduce the simulation time from many days to a matter of hours, as shown below. The paper introduces in a parallel implementation based on the programming language Python using the module Pypar to control the parallel simulation.

## 2 METHOD

### 2.1 Description of the erosion model

Impact assessment has to cope with two main influences:

- one or more land use maps for each scenario
- climate data for thirty years to support the influence of climate.

The land use maps are the result of a simulation of a two-dimensional probability distribution $p(soil, crop)$. This distribution was estimated using the "mean natural yield" for each crop for all soil types of the investigation area and using an assumption on market prices and costs (for fertilizer, seed, machines, etc.) per crop unit according to the scenarios.

Climate change is a long-term trend, and the weather is a realization within this trend. Global climate models expect the temperature to rise by about 2K by 2050 and precipitation to decrease in the coming decades, especially in the vegetation period. Reliable estimations should be based on long-term simulations (at least 30 years, including different climate realizations for each year with a daily resolution), and should be applied to large regions such as the Federal State of Brandenburg, Germany.

Soil erosion risk assessment is based on the globally applied Universal Soil Loss Equation (USLE) [Wischmeier and Smith, 1978]. The USLE calculates the annual amount of eroded soil A $t \times (ha \times a)^{-1}$ by multiplying five different factors: the R-factor (rainfall runoff erosivity), the K-factor (soil erodibility), the LS-factor (slope length and slope-angle affected erosion), the C-factor (cover management induced soil coverage by crops) and the P-factor (protective management measures against soil erosion). The equation is written as:

$$A = LS \times K \times R \times C \times P \tag{1}$$

The LS-factor covers the effects of topography and hydrology on soil loss; here, the slope steepness is more sensitive than the slope length. The LS-factor can be calculated from the digital elevation model (here DM 25) from which the LS-map was derived. The soil erodibility factor (K) is the soil loss rate per erosion index unit for a specified soil as measured on a standard plot, defined as a 22.1 m length of uniform 9% slope in continuous clean-tilled fallow. The K-factor was derived from a soil map (from the medium scale land classification "MMK" map). In simulations over 30 years, the LS- and K-factors are constant for every grid cell within the region. R = f(rain(t)) is the rainfall and runoff factor (rain erosivity factor), calculated according to Wischmeier and Smith (1978). The extreme rainstorm events required for the calculation are taken from climate scenario STAR2 [Gerstengarbe et al., 2003], a statistical-based regionalisation approach taking into account a temperature increase of 2K. Since the daily resolution of the climate scenario data contains only the daily precipitation amount and no information about the rain intensity, an assumption concerning rain intensity $(I)$ was made. From side of the energy of rain, a rain event is insignificant to erosion if the daily precipitation amount is $(rainThreshold < 10mm)$. To calculate the intensity of rain events with a daily precipitation amount $\geq 10mm$, an event duration of 2 hours $(T_N = 2h)$ was assumed. The calculation of the R-factor is based on the energy $(E)$ of each extreme rain storm event with $rainThreshold > 10mm$.

C is the crop cover and management factor, which is the ratio of soil loss from an area with specified cover and management to soil loss from an identical area in tilled continuous fallow. The C-factor is dynamic over the year, and depends on crop type and agro-management. Knowing the initial land use map, the crop rotation and the crop growth stage, depending on coverage, the C-factor values during the vegetation period are taken from a table for C-factors of crops typical for North-East Germany, as provided in Steidl et al. [1999]. To take into account the influence of the soil tillage method on erosion, in the table users can choose between conventional soil tillage, reduced soil tillage and no tillage. Due to the multiplication of all factors in Equation 1, the C-factor needs to be calculated only on days with a rain event significant to soil erosion. P is the support practice factor. This factor is the ratio of soil loss with a support practice, such as contouring, strip cropping and terracing, to soil loss with straight-row farming up and down the slope. Dependent on cropping practises typical for subregions, the P-factor can take values from (0,1). For the Federal State of Brandenburg, the support practice factor is assumed to be constant.

### 2.2 Parallelization techniques

A traditional approach is to use massive parallel computes, as described in Pirozzi and Zicarelli [2000]. This approach competes with the use of computer clusters, which are cheap and easy to use. For example, a computer cluster was used to find optimum parameters for a multi-scale landscape analysis [Wieland et al., 2011]. Today, computer clusters are replaced by even cheaper hardware using a Graphics Processing Unit (GPU). In addition to computer hardware, different software options exist for parallel processing. It is very common to use Message Passing Interface (MPI)[1], which implements inter process communication between independently running processes. Alternatively, a thread-based parallelization approach based on Open Multi-Processing (OpenMP)[2] can be deployed. Use of OpenMP requires the addition of pragmas in the source code. This often avoids the software having to be redesigned. A comparison of different parallelization approaches is given in [Neal et al., 2010].

Before parallelization using N cores can be made, the conditions for the simulation should be specified precisely. STAR2 climate data includes measured values for 1951-2006 and for 2007-2060 100 independent equally probable simulated values for each year. This data is stored in a mysql database on a different server to the simulation machine. The data should be loaded into the RAM for all stations of the investigation region before commencing the real simulation. The maps are grids with $5.73 * 10^6$ grid cells for all inputs (DEM, MMK, crop distribution, distribution of climate stations) which should also be loaded before the simulation starts.

According to Amdahls law [Smiatek, 2008], both loading times are the sequential part (1-P) that cannot be reduced by parallelization[3]. Another important condition for parallelization is the communication effort o(P) which is needed for information exchange

[1] http://www.open-mpi.org/
[2] http://openmp.org/wp
[3] One way to make it faster is to distribute the database to different servers.

between parallel parts:

$$S = \frac{1}{(1-P) + P/N + o(P)} \leq \frac{1}{1-P} \tag{2}$$

In the simulation of the USLE model for water erosion, there is no need for communication between the processes because each simulated grid cell has no influence on its neighbors. This simplifies parallelization enormously. On the other hand, the data reading in the sequential part can be a bottleneck.

A general approach for parallelization is to split the region into smaller rectangular regions with or without communication at the borders. This approach is often used in CFD-modeling, as in Popinet [2003]. For the climate change impact assessment, the simulation can also be split into yearly time slices. This is especially easy because “for loop” in the sequential program can be replaced by multiple processes. The sequential part of the parallel implementation has to provide the weather and spatial data. In the parallel part, a process calculates soil erosion for one year and stores the result before it dies. This is organized in a “bucket” structure, see Fig. 1: the first process reads the data and then sends the starting message to the second process, and so on. This ensures that only one process has access to the database and the hard disk at the time.

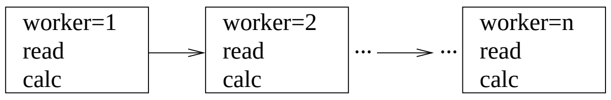

Figure 1: Bucket structure for process control

For parallel implementation in ZALF, a small cluster computer for parallel computing can be used. It is a SUN AMD86-based computer consisting of two parts (cluster1, cluster2), see Fig. 2. This enables the two parts to be used independently or together. The cluster is used not only to save simulation time but also to solve problems requiring a large memory. Users can access memory from the SAN and gain access to their data via SAMBA.

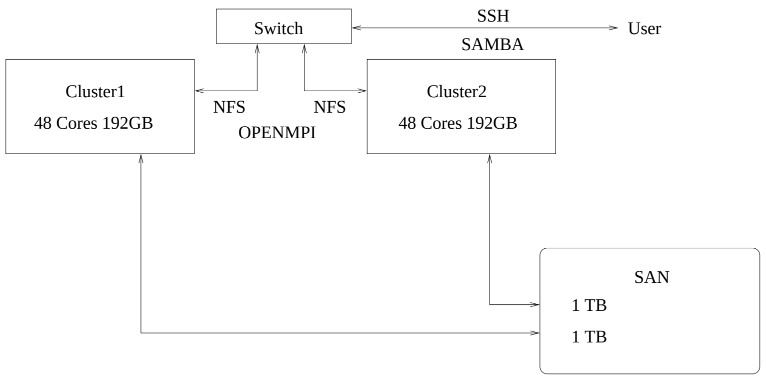

Figure 2: Cluster structure and connection

The cluster is controlled by the operation system Linux (Debian 6.0). OpenMPI and OpenMP are installed as software.

### 2.3 Implementation based on OpenMPI using Python, Swig and Pypar

All basic software for impact assessment is written in C++. Using OpenMPI directly would lead to reimplementation of the C++ code due to the use of message passing to control the processes. To avoid this reimplementation, all C++ code was wrapped in Python modules using the wrapper SWIG[4] in the first stage of development. After wrapping, all classes and their methods of the original C++ code are now available in Python. At this stage of development, the code can be checked in sequential order using small test examples. Matplotlib[5], a module for scientific visualization which comes with Python, is used to analyze and verify the models.

In the second stage of development, Pypar[6] is used to start the parallel processes and control the simulation. Pypar provides an easy-to-learn command set for parallel computing, based on the underlying OpenMPI. It is important to note that Pypar controls the simulation; the simulated code is still the C++ code and runs with native speed.

The main advantage of OpenMPI implementation is that all processes use their own memory and run independently. There is no shared memory between processes, which simplifies the software development. On the other hand, each process needs all the data, which must be loaded from the database or hard disk. This sequential loading can increase the calculation time. Inter process communication often uses a network connection, meaning that substantial communication can be the bottleneck of an OpenMPI implementation[7].

### 2.4 Implementation based on OpenMP

OpenMP is an application programming interface that supports shared memory multiprocessing on SMP computers. This allows data to be loaded and used in all threads. In general, communication is fast because it is based on the system bus of the SMP computer. Modellers have to analyze the source code to find the most time-consuming parts. After code analysis, they can add pragmas to the source code (which compilers will ignore in sequential applications) control parallelization. Although this is often easy, there are a number of traps. Modellers are responsible for the correct code. The following code fragment demonstrates this:

Listing 1: OpenMP

```
...
#pragma omp parallel for private(j)
for(i=0; i<c_grid->nrows; i++){
  for(j=0; j<c_grid->ncols; j++){
    if((int)c_grid->feld[i][j]!=c_grid->nodata)
      c_grid->feld[i][j]=table.get(doy,(int)plant->feld[i][j]);
  }
}
...
```

[4]http://www.swig.org/
[5]http://matplotlib.sourceforge.net/
[6]http://code.google.com/p/pypar/
[7]SMP computer communicates over the system bus which reduces the communication effort.

The pragma tells compilers to split the loop (for i=0; ...) into threads using a private j for all threads. Without the private(j), each thread would use the common j for all threads, which can be dangerous when threads do not run synchronously. It is difficult to find errors due to forgotten private(j). The private(j) is not the only trap. Others, such as race conditions due to shared variables, are more difficult to identify. Modellers have to bear in mind that all threads use the same memory space!

## 3 Results

The simulation creates 30 maps (years: 2021-2050) containing the mean soil erosion for each year and the same for its standard deviation. These maps are used to calculate statistics, often used for impact assessment or to extract parts of the region with high erosion risk.

In terms of parallelization, the computing time and speed-up are more interesting. A single run for 30 years, including the loading time from the database and the loading of the spatial data, takes 271 minutes, 59 s. The loading time alone takes 15.8 s. The expected speed-up according Equation 1 using N=30 processors (30 for calculation and one for control) is:

$$S = \frac{271 * 60 + 59}{30 * 15.8 + (271 * 60 + 59 - 15.8)/30} \approx 15.8 \quad (3)$$

The measured speed-up was: $\frac{16379}{1066} \approx 15.4$. The difference is caused by the time Open-MPI requires to initialize the parallel environment. A speed-up of 15.4 means that the computation time was reduced from $T_S \approx 4h33m$ to $T_P = 17m46$.

The OpenMP implementation induced a speed-up for a single year of $\frac{357.7s}{35.1s} \approx 10$ measured on the same computer. This is not bad compared to the small changes of the source code.

## 4 Discussion

Even such a simple model as the USLE can be time-consuming when the model region is large and a 30-year time period is used for climate data. Parallelization can help to reduce the calculation time significantly. The method introduced to wrap the time-consuming calculation, written in C++ to a python module and to control the parallel simulation using pypar [Dalcin et al., 2011], has the following advantages:

- no need to adapt the C++ source code
- full access to all classes and their methods
- the python script is easy to understand and maintain
- the script runs with nearly native speed of the wrapped C++ code
- visualization techniques (matplotlib, mayavi2) can be used to analyze the simulation during run time.

The implementation is split into a sequential and a parallel part. The sequential part (database access and map loading) is often the bottleneck of the parallel implementation.

In the USLE model, the sequential part was more efficient when each process loaded only the climate data for the simulation year and not for the whole simulation period. This would necessitate an adaptation of the underlying C++ source code and would influence other models of the impact assessment tool. The estimated time gain would be $31 * 15.8s - 31 * 6s = 303.8s \approx 5m$ for one simulation run with a fixed land use map for a simulation time of 30 year.

In the parallel part, each process is started and runs for the i-th year before it dies. The number of events in the i-th years depends on the climate data, and varies from year to year. For a larger simulation including different land use modes, each process could inform the master process of its end, and the master process could start a new process to ensure load balancing. In our simulation, this can save approximately 10-20% of the parallel simulation time.

Data exchange between processes, which is often necessary, is based on the message passing of OpenMPI. The transfer rate is very high due to the memory to memory copy of the SMP cluster architecture used. The number of messages can be the limiting factor. MPI has to provide additional control information before sending a message; the transfer medium has to carry this information and the receiver has to extract it. This leads to a latency before information can be used. For a larger message, the additional control information is short compared to the message, which makes the transfer of larger messages more efficient. The transfer should therefore be organized using data vectors instead of single data transfer.

The use of OpenMP implementation was very easy for all impact assessment models, including the USLE model. Compared to this small effort, the speed-up 10 is remarkable. Due to memory sharing, the OpenMP implementation can further reduce loading time and the calculation. A more elaborate software approach based on OpenMP, however, will require more effort in the analysis and adaptation of the source code.

## 5 Conclusion

The described parallelization technique was able to reduce simulation time significantly. This is true for both OpenMPI and OpenMP. The advantage of the key technology based on OpenMPI and pypar is that a well- designed source code can be used without making any changes. In terms of impact assessment, all models can be simulated using different landuse maps for at least 30 years of climate data within one week. This short simulation time allows stakeholders to react to current political discussions, such as the amendment of the German Renewable Energy Act.

As an alternative thread-based parallelization technique, OpenMP can reduce the simulation time, but is based on a preceding analysis of the source code. For impact assessment models such as the USLE model, OpenMP can speed up simulation by 10 times. This is a good value compared to the small change generated by the source code.

There are limitations to both techniques for speeding up simulation. But one aim of this paper was to show how easily parallelization could be implemented and which speed was possible with this simple implementation.

## 6 Acknowledgments

This contribution was supported by the German Federal Ministry of Food, Agriculture and Consumer Protection, and the Ministry of Infrastructure and Agriculture of the Federal

State of Brandenburg (Germany).

## REFERENCES

Dalcin, L., R. Paz, P. Kler, and A. Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, xx:in print, 2011.

Gerstengarbe, F.-W., F. Badeck, F. Hattermann, V. Krysanova, W. Lahmer, P. Lasch, M. Stock, F. Suckow, F. Wechsung, and P. Werner. *PIK-Report No. 83*, chapter STUDIE ZUR KLIMATISCHEN ENTWICKLUNG IM LAND BRANDENBURG BIS 2055 UND DEREN AUSWIRKUNGEN AUF DEN WASSERHAUSHALT, DIE FORST- UND LANDWIRTSCHAFT SOWIE DIE ABLEITUNG ERSTER PERSPEKTIVEN, page 79pp. PIK Potsdam, 2003.

Neal, J., T. Fewtrell, P. Bates, N. Wright, and G. Smiatek. A comparison of three parallelisation methods for 2d flood inundation models. *Environmental Modelling*, 25:398–411, 2010.

Payraudeau, S. and H. van der Werf. Environmental impact assessment for a farming region: a review of methods. *Agriculture, Ecosystems and Environment*, 107:2–19, 2005.

Pirozzi, M. and M. Zicarelli. Environmental modeling on massively parallel computers. *Environmental Modelling*, 15:489–496, 2000.

Popinet, S. A tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, 190:572–600, 2003.

Smiatek, G. Parallelization of a grid-oriented model on the example of a biogenic volatile organic compounds emission model. *Environmental Modelling*, 23:1468–1473, 2008.

Steidl, J., J. Quast, S. Fritsche, M. Behrens, D. Deumlich, L. Voelker, M. Mioduszewski, and I. Kajewski. *Diffuse entries in rivers of the Oder Basin.*, page 132pp. (DVWK), Bonn, 1999.

Wenkel, K.-O., W. Mirschel, M. Berg, R. Wieland, and B. Koestner. Experience from the use of the interactive model- and gis-based information and decision support system landcare-dss for the development of economic effective application strategies of agriculture to climate change. In *HAICTA 2011 - 5th International Conference on Information Technologies in Agriculture, Food and Environment*, pages 129–140, 2011.

Wieland, R., C. Dalchow, M. Sommer, and F. K. Multi-scale landscape analysis (msla) - a method to identify correlation of relief with ecological point data. *Ecological Informatics*, 6:164–169, 2011.

Wischmeier, W. and D. Smith. *Predicting Rainfall Erosion Losses: A Guide to Conservation Planning.*, chapter xxx, page 58. USDA/Science and Education Administration, 1978.