Improving the scalability of an environmental modelling framework to allow for large-scale high-resolution geosimulations

Oliver Schmitz, Kor de Jong and Derek Karssenberg
Utrecht University
Department of Physical Geography
Utrecht, The Netherlands
{o.schmitz, k.dejong1, d.karssenberg}@uu.nl

Abstract

Keywords: spatio-temporal modelling, raster, shared memory multiprocessing, scalable software

The increasing amount of high-resolution spatio-temporal data is an asset to environmental modellers as it helps them to refine their simulation models. Incorporating big data sets in environmental models and analysing model results, however, comes at a cost: a high computational demand to perform model simulations. The heterogeneity of current computing hardware like multi-core CPUs, workstations with GPUs, or access to supercomputers potentially provides significant computing power to satisfy this demand. Making use of this power for simulation models, however, requires detailed knowledge of the underlying hardware and experience with designing parallel algorithms. In addition, these algorithms need to be implemented using an efficient system programming language and additional libraries such as OpenMP (e.g. Chapman et al., 2007) or OpenCL (Khronos Group, 2017). Domain scientists such as hydrologists or ecologists often lack this specific knowledge on parallel computing. Their emphasis is on exploratory building and analysis of simulation models. Consequently, models constructed by domain specialists mostly do not take full advantage of the available hardware. To support domain scientists in building efficient models, we propose to offer them a modelling framework with standard building blocks that they can freely combine into a model. The model building framework, as a result, needs to have built-in capabilities to make full usage of the available hardware.

Developing such an environmental modelling framework imposes several challenges on the software developers of such a framework. On the one hand, the modelling code needs to be plain and understandable for domain scientists such that straightforward model construction and modification is guaranteed. On the other hand, the execution of the constructed models needs to be runtime efficient. Without knowing the complexity of the problem provided by the modeller (i.e. the spatial and temporal extent, the number and sequence of operations in the model), several optimisation approaches are feasible. For example, optimisations can be performed on individual operations or the whole model (e.g. Huang et al., 2015), by subdomain decomposition (e.g. Shook et al., 2016), or tasks need to be generated for a well-balanced execution (e.g. Donato, 2017). Ideally, a modelling framework supports the best use of available hardware independent of the combination of model building blocks the domain scientists use.

Figure 1: Script to calculate the slope in PCRaster Python. By setting an environment variable the execution of slope will be either sequential or multithreaded, the script itself remains unchanged.

```
from pcraster import *

elevation = readmap("dem.map")
slope_map = slope(elevation)
report(slope_map, "slope.map")
```

We demonstrate our ongoing work on developing parallel algorithms and software for spatio-temporal modelling and demonstrate our work on PCRaster (Karssenberg et al., 2010) 1) an environmental software framework tailored to domain scientists, providing a wide range of spatio-temporal model building blocks based on the map algebra concept (Tomlin, 1990) and 2) the parallelisation of about 50 of these building blocks (i.e. local and focal) using a newly developed generic raster processing library. The Fern (2017) raster library is a highly generic software library written in C++ and provides shared-memory multiprocessing algorithms using standard C++11 threads. Its algorithms can be tailored to the configuration of a modelling framework. With manageable programming effort (e.g. matching data types between programming and domain language) we created a Python binding between the algorithm library and the PCRaster modelling framework. The resulting Python package can be used to execute models constructed with the modelling framework without having to make any changes to existing model code. An example implemented in PCRaster Python is given in Figure 1 showing a slope calculation from a digital elevation map. By default, the slope operation will be calculated sequentially. By simply setting the environment variable PCRASTER_NR_WORKER_THREADS to a number larger than one, the multithreaded slope operation will be executed.

We show results obtained from synthetic and geoscientific simulation models indicating significant runtime improvements by using the new parallel local and focal operations. These operations are included in our code repository (PCRaster, 2017) and will be included in the next release package. We also outline three further challenges in processing big data sets. First, the performance of remaining algorithms that interact with direct or distant neighbouring cells need to be improved. These algorithms, for example, calculate transport and accumulation of material in flow operations (based on the topography of a river catchment) and are often used in hydrological models. Second, using large model extents also increases the access time when reading inputs and writing model results to disk. We therefore also briefly discuss further potential improvements in enhancing disk I/O. Lastly, we plan to work on algorithms that divide the work across distributed memory systems, like a computer cluster. Several implementation strategies are feasible to reach distributed processing, such as building on MPI (e.g. Donato, 2017), MapReduce (e.g. Shi et al., 2015), or Apache Spark (e.g. GeoTrellis, 2017). We plan to use HPX (Kaiser et al., 2016), a C++ library and runtime system for parallel and distributed applications.

References

Barbara Chapman, Gabriele Jost, and Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*. The MIT Press, 2007.

David I. Donato. Simple, efficient allocation of modelling runs on heterogeneous clusters with MPI. *Environmental Modelling & Software*, 88:48–57, 2017. doi: 10.1016/j.envsoft.2016.11.003.

Fern. Fern source code repository, 2017. URL https://github.com/geoneric/fern. Accessed 20 April 2017.

GeoTrellis. GeoTrellis geographic data processing engine, 2017. URL https://geotrellis.io/. Accessed 20 April 2017.

Chien-Chin Huang, Qi Chen, Zhaoguo Wang, Russell Power, Jorge Ortiz, Jinyang Li, and Zhen Xiao. Spartan: A Distributed Array Framework with Smart Tiling. In *Proceedings of the 2015 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '15, pages 1–15, Berkeley, CA, USA, 2015.

Hartmut Kaiser, Bryce Adelstein-Lelbach, Thomas Heller, Agustín Bergé, John Biddiscombe, Anton Bikineev, Grant Mercer, Andreas Schäfer, Jeroen Habraken, Adrian Serio, Matthew Anderson, Martin Stumpf, Daniel Bourgeois, Patricia Grubel, Steven R. Brandt, Marcin Copik, Vinay Amatya, Kevin Huck, Lars Viklund, Zahra Khatami, Devang Bacharwar, Shuangyang Yang, Erik Schnetter, Bcorde5, Maciej Brodowicz, Bibek, atrantan, Lukas Troska, Zach Byerly, and Satyaki Upadhyay. hpx: HPX V0.9.99: A general purpose C++ runtime system for parallel and distributed applications of any scale, July 2016. URL https://doi.org/10.5281/zenodo.58027.

Derek Karssenberg, Oliver Schmitz, Peter Salamon, Kor de Jong, and Marc F. P. Bierkens. A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. *Environmental Modelling & Software*, 25(4):489–502, 2010. doi: 10.1016/j.envsoft.2009.10.004.

Khronos Group. OpenCL website, 2017. URL https://www.khronos.org/opencl/. Accessed 20 April 2017.

PCRaster. PCRaster source code repository, 2017. URL https://github.com/pcraster/pcraster. Accessed 20 April 2017.

Qiqi Shi, Hongzhi Wang, Dong Li, Xinfei Shi, Chen Ye, and Hong Gao. Maximal influence spread for social network based on mapreduce. In Hongzhi Wang, Haoliang Qi, Wanxiang Che, Zhaowen Qiu, Leilei Kong, Zhongyuan Han, Junyu Lin, and Zeguang Lu, editors, *Intelligent Computation in Big Data Era*. ICYCSEE 2015, volume 503, pages 128–136. Springer, Berlin, Heidelberg, 2015. doi: 10.1007/978-3-662-46248-5_16.

Eric Shook, Michael E. Hodgson, Shaowen Wang, Babak Behzad, Kiumars Soltani, April Hiscox, and Jayakrishnan Ajayakumar. Parallel cartographic modeling: a methodology for parallelizing spatial data processing. *International Journal of Geographical Information Science*, 30(12):2355–2376, 2016. doi: 10.1080/13658816.2016.1172714.

C. D. Tomlin. *Geographic Information Systems and Cartographic Modeling*. Prentice Hall, Englewood Cliffs, NJ, 1990.