Computational Infrastructure of SoilGrids 2.0

Luís Sousa, Laura Poggio, Gwen Dawes, Bas Kempen, Rik van Den Bosch

Computational infrastructure of SoilGrids 2.0

Luís M. de Sousa[1][0000−0002−5851−2071], Laura Poggio[1][0000−0003−1892−0764], Gwen Dawes[2][0000−0002−1657−0986], Bas Kempen[1][0000−0003−0959−9358], and Rik van den Bosch[1]

[1] ISRIC - World Soil Information, Wageningen, NL luis.desousa@isric.org
[2] FB-IT, Wageningen University and Research, NL

Abstract. SoilGrids maps soil properties for the entire globe at medium spatial resolution (250 metres cell side) using state-of-the-art machine learning methods. The expanding pool of input data and the increasing computational demands of predictive models required a prediction framework that could deal with large data. This article describes the mechanisms set in place for a geo-spatially parallelised prediction system for soil properties. The features provided by GRASS GIS – *mapset* and *region* – are used to limit predictions to a specific geographic area, enabling parallelisation. The Slurm job scheduler is used to deploy predictions in a high-performance computing cluster. The framework presented can be seamlessly applied to most other geo-spatial process requiring parallelisation. This framework can also be employed with a different job scheduler, GRASS GIS being the main requirement and engine.

Keywords: Digital Soil Mapping · High-Performance Computing · GRASS GIS.

1 Introduction

Soil is key in the realisation of a number of UN Sustainable Development Goals by providing a variety of goods and services. Soil information is fundamental for a large range of global applications, including assessments of soil and land degradation, sustainable land management and environmental conservation. It is important to provide free, consistent, easily accessible, quality-controlled and standardised soil information. Spatial soil information is often available as maps of soil properties, e.g. pH, carbon content, texture information. Soil is a 3D body and the maps should describe the landscape (i.e. horizontal) variability as well as the vertical (i.e. along the soil depth) variability. Often, such maps are produced using a Digital Soil Mapping (DSM) approach [7], creating a statistical model between the properties measured at known locations and environmental covariates describing the soil forming factors [9]. In recent years machine learning methods [4] were used to develop such models. Once the model is calibrated and evaluated, it is used to predict soil properties at non visited locations.

SoilGrids is based on a DSM system that produces geo-spatial soil information fulfilling two main goals: 1) be a source of consistent soil information to

support global modelling, and 2) provide complementary information to support regional and national soil information products in data deprived areas. The main target user groups are policy makers and land management services in countries that lack the means to produce such information, as well as scientists developing other models and tools requiring soil data as input.

SoilGrids is of a series of maps for different soil properties following the specifications of the GlobalSoilMap project [1]. The production of a soil property map consists of different steps: the fitting of a model, the evaluation of that model and finally the prediction for each raster cell at the global scale with a spatial resolution of 250 metres. Six depth intervals are currently considered and prediction uncertainty is quantified. The total computation time required for a single soil property exceeds 1 500 CPU-hours. Parallelisation is therefore fundamental, requiring up to hundreds of CPUs.

This article outlines the computational infrastructure (hardware and software) employed to compute the latest SoilGrids products in a high-performance computing (HPC) cluster, focusing in particular on the parallelisation and the resource management for global DSM modelling.

2 General Framework

SoilGrids requires an intensive computational workflow, including different steps and integrating different software. SoilGrids is entirely based on open source software. The inputs to SoilGrids currently comprise observations from 250 000 soil profiles and a selection from over 400 global raster characterising soil forming factors: morphology (e.g. elevation, landform), vegetation information (e.g. NDVI and other indices), climate (e.g. precipitation, land surface temperature) and human factors (e.g. land use/cover).

Gathering and harmonising soil profile observations is one of the core workstreams performed by ISRIC, with the World Soil Information System (WoSIS) [2] being its most visible product. These observations are maintained in a relational database hosted by PostgreSQL [10], from which they are directly sourced for modelling. Environmental covariates are ingested into GRASS GIS [6, 11], thus being automatically normalised to a unique raster cell matrix (of 250 metres cells). Previous SoilGrids releases were computed on the millenary Marinus of Tyre map projection [16], a popular projection in environmental modelling. However, this projection expands the surface area of the globe by about 60%. With the increasing size of inputs and outputs in SoilGrids, this overhead urged the switch to an equal-area map projection. The Homolosine projection [5] was selected, since among those projections supported by open source software, is the one that best preserves the shapes of lands masses [17]. The size of each output raster was thus reduced in over 10^{12} cells.

The general framework of SoilGrids is described in Figure 1. The main steps are:

1. creation of a regression matrix that overlays soil profile data with environmental covariates.

2. fitting of the regression model. Currently this step is based on state-of-the-art machine learning methods, able to produce a measure of the uncertainty of the predictions, namely Quantile Regression Forests (QRF) [8].
3. prediction with the model at global scale.

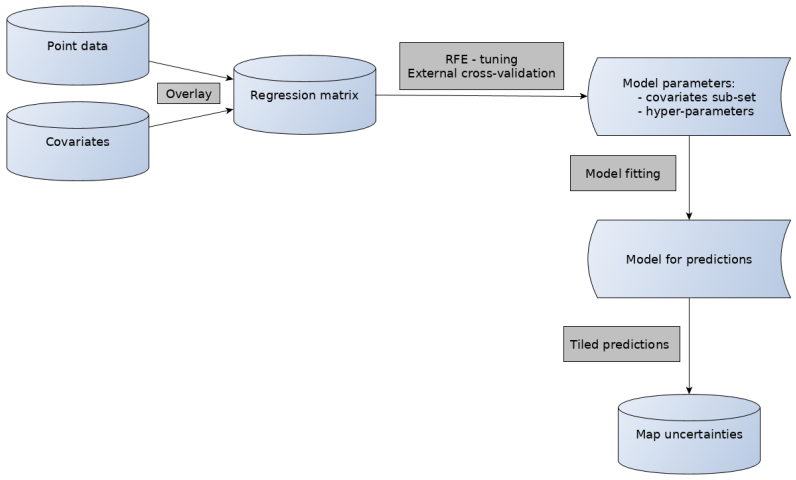

Fig. 1: General framework of SoilGrids.

3 HPC infrastructure

SoilGrids is currently computed on Anunna, an HPC cluster managed by Wageningen University and Research (WUR). Anunna provides more than 2 000 CPUs in an heterogeneous set of compute nodes. The majority of compute nodes provide 12 GB of RAM per CPU. Storage is managed by the Lustre distributed file system [15].

The Slurm Workload Manager [20] is used to manage the cluster workload. Slurm offers a large range of flexibility. The user may restrict computation to a specific type of node, require an exact number of CPUs to use in parallel, set the number of CPUs required by each process, set a computation time limit, request a specific amount of memory and declare the exact software packages to load at computation time. These settings are defined in a configuration file (the Slurm file), where the user sets the programmes to run and their parameters.

4 Parallelisation of global scale geo-spatial computations

SoilGrids requires the integration of a number of software. The two key components are GRASS GIS [6, 11] and R [13]. GRASS GIS is used to store the input

data and as the engine to control the parallelisation of the predictions. R is used for the statistical modelling, i.e. to calibrate, fit and compute the predictions using the QRF model.

The GRASS GIS *mapset* and *region* features are used to set up parallelisation. *Location* is a directory which contains GRASS GIS *mapsets*, which are its sub-directories. All data in one *location* must refer to the same spatial reference system (SRS).*Mapsets* contain the actual data, they are a tool for organising maps in a transparent way and provide isolation between different tasks to prevent data loss. GRASS GIS is always connected to one particular *mapset*. *Mapsets* are used to store maps related to a project, a specific task, issue or subregion. Besides the geo-spatial data, a *mapset* holds the resolution and extent of the current computational region. In this version of SoilGrids, the SRS used for the GRASS *location* is the Homolosine projection applied to the WGS84 datum.

Prior to prediction, a global tessellation is created dynamically using the GRASS module `r.tile`, dividing land masses into square tiles of a given side (Figure 2). Predictions are then executed independently, and in parallel, within each of these tiles.

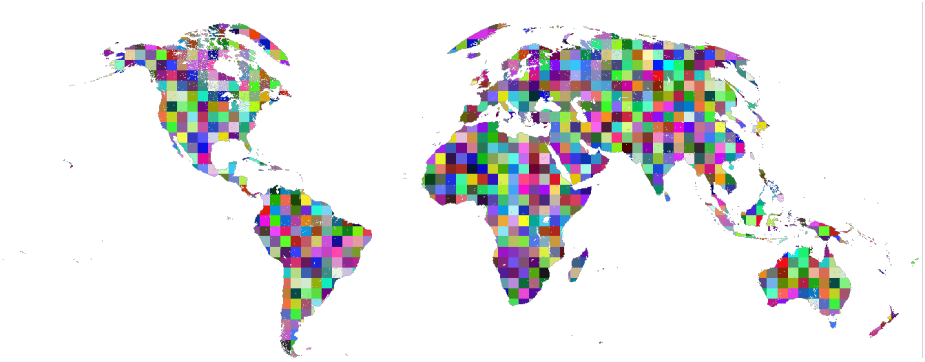

Fig. 2: Land masses tessellated with tiles of 450 km in side.

A Slurm file is used to start up each individual prediction process. The prediction process receives as argument the identifier of one of the tiles in the tessellation. The process then creates a temporary *mapset*, setting its *region* to the extent of the tile it is tasked to process. The temporary *mapset* works as a geo-spatial sand box for the prediction process. The prediction process loads data from GRASS within the extent of the tile, as set by the GRASS *region*. The prediction process is controlled by R software linked to GRASS with the `rgrass7` package [3]. The result is saved to disk as a GeoTIFF file and the temporary *mapset* deleted (Figure 3).

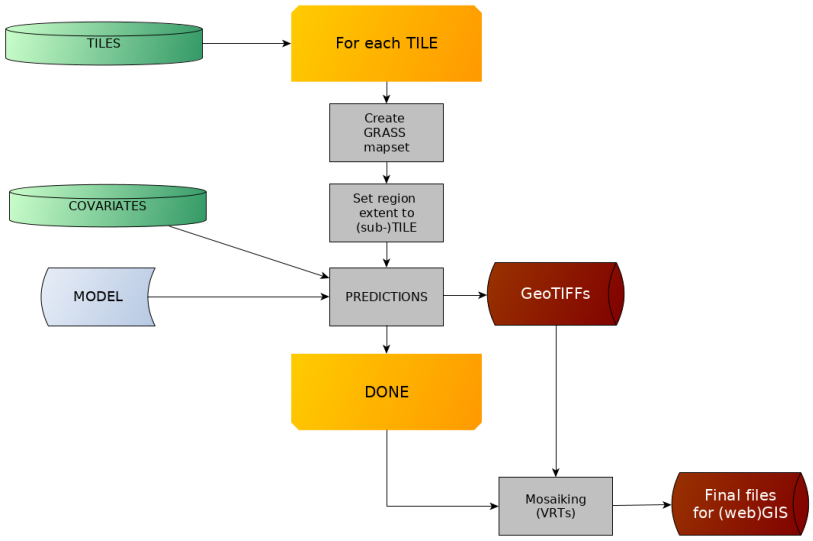

Fig. 3: Predictions occur in parallel in each of these tiles.

5 Resources management with (sub-)tiling

Prediction models developed with the R language may require a large amount of RAM, i.e. tens of GBs of RAM to execute predictions for a single tile under 1 MB in size. The obvious strategy to tackle memory constraints would be to reduce the size of the tiles used to set up the temporary GRASS *mapsets*. However, this can soon result in tens of thousands of GeoTIFF files. Such large numbers can be excessive for some file systems, and complicate the functioning of command line tools [19].

In the case of SoilGrids, a different strategy was applied to address resource constraints, i.e. sub-tiling. The prediction process sub-divides the tile in equal-size sub-tiles. Within the temporary GRASS *mapset*, the prediction process successively sets the *region* to each of these sub-tiles and invokes the general prediction model. The output computed within each sub-tile is finally saved into a folder named after the prediction tile. The end result is a collection of folders, one per tile, each containing as many GeoTIFF files as the number of sub-tiles contained in the tile.

6 Assemblage of prediction files

The sub-tilling strategy can result in a very large number of GeoTIFF files. For example, a three-by-three sub-tiling matrix applied to tiles of $200x200$ km results in over 50 000 GeoTIFF files covering the globe's land masses. All these files must then be aggregated to produce a single asset, easily manageable by end

users. The Virtual Raster Tiles (VRT) format [12] introduced by the Geospatial Data Abstraction Library (GDAL) is a suitable solution, as it provides a simple and lightweight way to mosaic geo-spatial data. A VRT is in essence a XML file that patches together various contiguous rasters in the same SRS into a mosaic. As the GDAL library is used as I/O driver by most GIS programmes this format is widely supported.

GDAL provides a specific tool for the of creation of VRT mosaics: `gdalbuildvrt`. It can take as input a file with the list of rasters to include in the output VRT. Another GDAL tool, `gdaladdo`, can afterwards be employed to create overviews for the VRT. These overviews are stored in a companion file to the original VRT. The end user needs only to point a GIS programme to the VRT file to load the full raster mosaic. With the overviews created, access is fast and fluid at different map scales.

The VRT format is also useful to simplify the re-projection of large rasters. Another GDAL tool – `gdalwarp` – can be applied directly to a VRT file, creating a second VRT file encoding the parameters of the specified output SRS. No transformations are conducted on the rasters themselves, thus a swift operation.

7 Reproducibility and Portability

The parallelisation scheme described in this article is directly portable to any system using Slurm where GRASS GIS and R can be installed. It is also applicable to any other system relying on a similar scheduling mechanism, such as Son of Grid Engine or Mesos [14]. In essence, any system able to spawn processes passing a tile identifier as parameter can be used to reproduce this set up. This solution can also be easily adapted to run on single machines with a large number of CPUs and managed by the `GNU parallel` tool [18].

While GRASS GIS is a key component in the approach described wherewith, alternative software can be used to similar ends. Certain raster formats, like GeoTIFF, store rasters in blocks of constant size. It is therefore possible to parallelise spatial computation on the basis of such blocks. GDAL in particular provides an API that facilitates the retrieval of these blocks. However, some pre-processing might be necessary to guarantee that all rasters stack up, using blocks of equal size and spatial extent. The *mapset* and *region* features in GRASS are unique among open source software and perform this stacking of input rasters in a seamless way.

The parallelisation approach reported in this article can be applied to other types of modelling where results can be easily parallelised in space. The tiling mechanism is fully dynamic and independent of the underlying SRS. The size of both the main map tiles and its sub-tiles are parameters to the tiling routine, and expressed in number of raster cells, not map units. It is therefore straightforward to apply with different SRSs and computation extents. The main limitation of this approach is with problems that can not be easily parallelised in space, such as hydrological processes requiring continuity or interactions between catchments.

References

1. Arrouays, D., Grundy, M.G., Hartemink, A.E., Hempel, J.W., Heuvelink, G.B., Hong, S.Y., Lagacherie, P., Lelyk, G., McBratney, A.B., McKenzie, N.J., d.L. Mendonca-Santos, M., Minasny, B., Montanarella, L., Odeh, I.O., Sanchez, P.A., Thompson, J.A., Zhang, G.L.: Chapter three - globalsoilmap: Toward a fine-resolution global grid of soil properties. Advances in Agronomy, vol. 125, pp. 93 – 134. Academic Press (2014). https://doi.org/http://dx.doi.org/10.1016/B978-0-12-800137-0.00003-0
2. Batjes, N.H., Ribeiro, E., van Oostrum, A.: Standardised soil profile data to support global mapping and modelling (wosis snapshot 2019). Earth System Science Data Discussions **2019**, 1–46 (2019). https://doi.org/10.5194/essd-2019-164, https://www.earth-syst-sci-data-discuss.net/essd-2019-164/
3. Bivand, R.: rgrass7: Interface Between GRASS 7 Geographical Information System and R (2018), https://CRAN.R-project.org/package=rgrass7, r package version 0.1-12
4. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
5. Goode, J.P.: The homolosine projection: a new device for portraying the earth's surface entire. Annals of the Association of American Geographers **15**(3), 119–125 (1925)
6. GRASS Development Team: Geographic Resources Analysis Support System (GRASS GIS) Software, version 7.6.0 (2019), http://www.grass.osgeo.org
7. McBratney, A., Santos, M., Minasny, B.: On digital soil mapping. Geoderma **117**, 3–52 (2003). https://doi.org/10.1016/S0016-7061(03)00223-4
8. Meinshausen, N.: Quantile regression forests. Journal of Machine Learning Research (2007)
9. Minasny, B., McBratney, A.: Digital soil mapping: A brief history and some lessons. Geoderma **264, Part B**, 301 – 311 (2016), soil mapping, classification, and modelling: history and future directions
10. Momjian, B.: PostgreSQL: Introduction and concepts. Addison-Wesley, New York (2001)
11. Neteler, M., Mitasova, H.: Open source GIS: a GRASS GIS approach, vol. 689. Springer Science & Business Media (2013)
12. OSGeo Foundation: VRT GDAL Virtual Format. https://gdal.org/drivers/raster/vrt.html (2019), accessed: 2019-10-09
13. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2019), http://www.R-project.org/, ISBN 3-900051-07-0
14. Reuther, A., Byun, C., Arcand, W., Bestor, D., Bergeron, B., Hubbell, M., Jones, M., Michaleas, P., Prout, A., Rosa, A., et al.: Scheduler technologies in support of high performance data analysis. In: 2016 IEEE High Performance Extreme Computing Conference (HPEC). pp. 1–6. IEEE (2016)
15. Schwan, P., et al.: Lustre: Building a file system for 1000-node clusters. In: Proceedings of the 2003 Linux symposium. vol. 2003, pp. 380–386 (2003)
16. Snyder, J.P.: Flattening the earth: two thousand years of map projections. University of Chicago Press (1997)
17. de Sousa, L.M., Poggio, L., Kempen, B.: Comparison of FOSS4G Supported Equal-Area Projections Using Discrete Distortion Indicatrices. ISPRS International Journal of Geo-Information **8**(8), 351 (2019)

18. Tange, O.: GNU Parallel 2018. Ole Tange (Apr 2018). https://doi.org/10.5281/zenodo.1146014, https://doi.org/10.5281/zenodo.1146014
19. Thomas Koenig: `sysconf` manual page, in *Linux Programmer's Manual*. http://man7.org/linux/man-pages/man3/sysconf.3.html (2019), accessed: 2019-10-15
20. Yoo, A.B., Jette, M.A., Grondona, M.: Slurm: Simple linux utility for resource management. In: Workshop on Job Scheduling Strategies for Parallel Processing. pp. 44–60. Springer (2003)