GeoSkelSL: A Python High-Level DSL for Parallel Computing in Geosciences

Kevin Bourgeois[1,2], Sophie Robert[1], Sébastien Limet[1], and Victor Essayan[2]

[1] Univ.Orléans, INSA Centre Val de Loire, LIFO EA4022, Orléans, France
(kevin.bourgeois, sophie.robert, sebastien.limet)@univ-orleans.fr
[2] Géo-Hyd (Antea Group), Olivet, France
(kevin.bourgeois, victor.essayan)@anteagroup.com

Keywords: GIS, DSL, Implicit parallelism, Performance

Abstract. This paper presents GeoSkelSL a Domain Specific Language (DSL) dedicated to Geosciences that helps non experts in computer science to write their own parallel programs. This DSL is embedded in Python language which is widely used in Geosciences. The program written by the user is derived to an efficient C++/MPI parallel program using implicit parallel patterns. The tools associated to the DSL also generate scripts that allow the user to automatically compile and run the resulting program on the targeted computer.

1 Introduction

In the last decades, in almost every sciences, the amount of data to process have grown dramatically because of the technological progress that improved the resolution of the measuring tools and also because of the emergence of new technologies such as sensor networks. These progress allow scientists to refine their theoretical models to make them more precise which increases the need of computation power to compute them. Geosciences are particularly affected by these trends. Indeed, geosciences cover an area at the edge of the computer science and the earth sciences and aim at gathering, storing, processing and delivering geographic information. The global warming and the pollution or water resource problems became global concerns which reinforce the need for geosciences. For example, many countries impose regulations that command to collect and analyze some environmental data to provide indicators to the citizens.

On the other hand, during the same period, parallel computers became widespread since nowadays almost all computers have several processing units. It is now quite easy to any scientist to access a powerful parallel computer like a cluster. However, writing programs that efficiently exploit the computing power of such computers is not an easy task and requires high expertise in computer science. Unfortunately, very few people have great skills in both their scientific domain and computer science.

A first solution, form a team composed of geoscientists and computer-scientists expert in parallel programming, may be difficult to implement because it could

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83

take a long time to make understand them each others. It may be a problem when programs become too complicated and the scientist loose the expertise on his codes. This could be also financially too expensive for small scientific units.

A second solution, provide tools to help non computer-scientists to produce efficient parallel program, is more attractive since it tends to make the geoscientist independent from the technicalities of parallel programming which allows him to master his programs. However, providing such tools needs to resolve three main issues, first, providing an easy-to-use programming language that is accepted by the scientist, then, being able to derive an efficient parallel program from the one written by the user and providing tools to easily run the generated programs on parallel machines.

According to their domain, the geoscientists are familiar with Python as GIS like ArcGIS or QGIS use it as script language. Therefore, the proposed programming language should be close to -or embedded in- Python.

Using implicit parallel patterns allow to hide some points of the parallelism. But using libraries as ScaLAPACK remains difficult because, programs are dependent of the provided routines which makes them difficult to evolve. The use of parallel patterns address this last issue however, they are usually based on a low-level language so they are hard to use for non computer scientists.

To provide a user-friendly tool that abstracts scientists from all the worries associated with parallelization, we the Geoskel framework. It relies on parallel patterns associated to a *Domain-Specific Language* (DSL) embedded in Python language. Thus, it is split into three layers: from a high-level programming language used to write algorithms, to a low-level data system to efficiently manage data on disks, with an implicit parallelism pattern layer in-between.

The GeoSkel heart is the implicit parallel patterns aimed to cover the recurring computations on rasters in geosciences [1]. This layer is written in C++ using template classes and the MPI parallel library. GeoSkelSL is the DSL top-level layer. It is a DSL embedded in Python allowing geoscientists to have a classical view of their GIS and to write their programs in a sequential way knowing the available predefined parallel patterns. This program is either executable in a Python context or can be transformed in a parallel program which can be automatically launched on a cluster. GeoSkelSL is the main contribution of the paper and is detailed Section 2 as well as the parallel program generation.

At the low-level layer, the datasets are stored in a distributed file system and the main objective of this layer in GeoSkel is to provide an efficient way to select the data required by the program and to distribute them according to the parallel execution.

The main tools used by geoscientists rely on Geographical Information Systems (GIS), like ArcGIS, QGIS [11] or GrassGIS [6]

These GISs were designed from sequential algorithms and focused on dealing with data of different formats as well as providing relevant user interfaces. They were not originally designed to be used on huge data sets. For several years now, some projects extend them with tools to perform parallel computations on large

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83

volume of data. SAGA-GIS has a module library and proposes parallel versions of some classical algorithms in hydrology.

Thanks to the possibility to add external modules, QGIS and ArcGIS allow to include some optimized parallel computations. For example, TauDEM [10] a tool set to analyze a Digital Elevation Model and based on parallel processing can be used both by QGIS and ArcGIS.

All these systems strongly depend on external contributions for parallel processing and for geoscientists who are not specialists of parallel computing, it is very difficult to implement a new algorithm that is not included in the used toolkit.

Implicit parallelism aims at helping non specialists to write parallel programs. The user writes the program in a sequential way and the transformation into a parallel program is realized thanks to automatic tools which tends to hide the technicalities of data distribution, data communication or task division.

One approach to propose implicit parallelism is to provide parallel containers that are automatically distributed among the nodes and that can be handled as sequential containers. The Standard Adaptive Parallel Library is a C++ library similar and compatible with the STL [8]. The user builds programs using View and pAlgorithms. The View is equivalent to the iterator in the STL library, it is used to access the pContainers, which are parallel data structures. The pAlgorithms are the parallel equivalent to the algorithms of the STL.

The second approach to provide implicit parallelism is to write the program in a specific programming language. In this context, DSL [2] can be used. In general, the DSL's role is to hide low-level tools to users while allowing them to develop their applications by mean of the features provided by these tools. For example, in geosciences, the DSL [5] aims to specify visualization of large scale geospatial datasets. Instead of writing the visualization code using low-level API as Google Maps or Javascript libraries a DSL is proposed to describe the application as data properties, data treatments and visualization settings. Then the DSL source code is transformed into the appropriate visualization code.

In the implicit parallelism context, many DSLs are proposed either to be suitable for a scientific domain or for a class of parallel algorithms. In the machine learning domain, OptiML [9] is an implicit parallel DSL built on Scala [7] which design allows to take advantage of heterogeneous hardware.

The approaches described in [4] propose a DSL to write programs based on classical parallel patterns. In [4], the DSL is composed of simple expressions to describe the operation sequences. the DSL code is transformed into an executable program using rewriting rules and refactoring code which introduces parallel patterns.

The DSLs presented above are not suitable for our purpose since none of them is both dedicated to geosciences and provides implicit parallelism.

Hence, we describe GeoSkelSL a DSL embedded in Python and detail the DSL source code transformation into parallel patterns in the C++ layer.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83

2 GeoSkelSL

In this section, we present the main contribution of the paper i.e. the DSL called GeoSkelSL. It is the top layer of GeoSkel framework. The first objective of this high-level programming language is to be accessible to geoscientists to allow them to write their own programs in an usual way. The DSL source code will be transformed into a parallel C++/MPI program that will be compile to be run on the targeted computer. Therefore, the original program needs to contain the useful information for this transformation which must be transparent for the user.

First, the main GeoSkelSL features are described. Then, the whole derivation process that transforms the DSL source code to a parallel program is detailed.

2.1 GeoSkelSL Features

GeoSkel is the interface with users. For its design, we have worked with geoscientists to understand their needs and expectations. This allowed us to extract the main features wanted in the DSL. They can be summarized in three points. The DSL must rely on a well-known language in geosciences, promote the expressiveness of the data they deal with, and hide the technical concerns related to the code parallelization. Our DSL is based on Python. It is a very common language in the geoscience domain and it is used in a lot of GIS tools such as ArcGIS, QGIS or GrassGIS. Therefore, Python is a well-known language to geoscientists, making it a good language to build a DSL on it. Moreover, embedding GeoSkelSL in Python language favors its use in GIS tools.

The Data. Our DSL deals with rasters which are matrices describing terrains with metadata as the projection used, the coordinates, the resolution, the color interpretation. It can have multiple bands and each of them can have its own datatype. A raster can depict the elevation of a terrain, soil pollutant concentrations at various depths, air pollutant concentrations. The rasters are used in many applications as watershed delineation, pollutant spreading and heat diffusion.

GeoSkelSL provides a `Mesh` class to handle rasters. Therefore a mesh is used to store each band of a raster. This class is also extended with basic operators corresponding to classical computations on a mesh as the addition or the subtraction. Thanks to a bracket operator for the cell access, a mesh is also usable as a standard 2D matrix. Regarding the metadata, GeoSkelSL uses GDAL [3] and its Python API to be able to read them. Therefore, `load_data` and `write_data` are the only functions necessary for the data management. They support the reading and the writing of the metadata information on the raster.

The Computation. The computation on rasters are based on the implicit parallel patterns implemented as a C++ library in our framework. These patterns allow to separate the writing of the computation to realize on the raster and

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83

the parallelization. According to the chosen pattern the user only needs to write some functions in Python to describe the computation to apply on the raster cells in a sequential way. For example, applying a median filter on a raster consists in using the stencil pattern. Thus, in GeoSkelSL the main program consists in few lines to describe how to load the data, which pattern to use and finally where the result is saved.

2.2 The program derivation

The program derivation consists in using the GeoSkelSL source code to concretely instantiate the parallel patterns. The main issues are the data types, the data distribution and the data dependencies that will define the exchanges necessary to the computation. The derivation process is based on a partial execution of the GeoSkelSL source code in order to guess the parameters of the parallelization. The steps illustrated Figure 1 allow to build the C++ main program with the relevant parameters to instantiate correctly the parallel pattern and to launch its execution on a cluster.

As GeoSkelSL is based on Python, the user does not explicitly type variables or functions. However, the targeted program is a C++ program where all variables are statically typed. The derivation process has to type each Python variable. Choosing the right type is very important regarding performance. For example, the run-time of a program can be much longer using `double` rather than `float`.

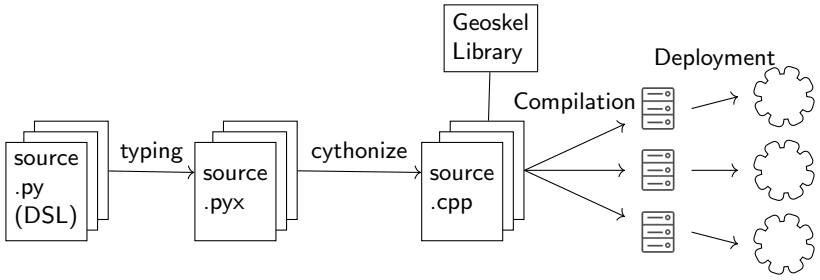

Fig. 1: Derivation process from the DSL source code to the parallel program.

Python has only two types to store numbers: `int` for integer and `float` to approximate real numbers. In C++, the programmer can control the precision of both integer and float numbers using different types like `short`, `int` or `long` for integers or `float` and `double` for real numbers. During the derivation from Python to C++, all variables and functions must be statically typed with the smallest equivalent in C++ for best performance. To do the type guessing the user must give at least the type of the incoming mesh. This can be done either using a parameter of the `load_data` function, or it can be guess by reading the metadata of the raster performing a partial execution of `load_data`. Each

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83

variable assignment of the Python program is then executed once to determine its type. NumPy is used to determine the smallest type needed in C++. NumPy is a Python library for scientific computing. NumPy types are very similar to C++ types and the execution of a variable assignment returns the smallest corresponding type.

The data distribution on the cluster nodes is based on a round-robin distribution which consists in splitting the mesh into sub-meshes in various shapes and sizes. For example, for the stencil pattern, a distribution per block of lines is chosen and the size of the block is defined according to the mesh size. The size of the raster is again guessed thanks to the partial execution of `load_data`.

The data exchanges depend on the data dependencies in the computation functions. Then, the size of the ghosts needs to be defined. In our patterns the computation of a raster cell can be dependent of a set of neighboring cells. As the data are distributed on different nodes the neighborhood of cells on edges is not local. These ghost cells need to be sent before the computation. The size of the ghosts can be guessed from the Python functions written by the user. Indeed, from the mesh accesses the neighborhood size can be defined as a distance with the studied cell. When multiple parallel pattern are called, all meshes share the larger ghost size found. This neighborhood size is necessary to the `load_mesh` function in the C++ main program.

The derivation from Python to C++ is based on the Cython library. This is a language very close to Python but it supports a subset of C/C++, like the typing, the variable declarations or the function calls. A Cython code (.pyx) can produce external modules than can be used in a standard C++ program. This feature is very useful to integrate the Python functions written by the user and which are the parameters of the parallel pattern. After the partial GeoSkelSL source code execution, a Cython typed code is generated with annotations to add the parameters related to the data distribution and the ghost size. The functions written by the user are also translated in Cython code. Then we use Cython associated to predefined rules to transform the original main program into a new main program in C++.

The programs are likely to be written on the desktop computer of geoscientists. the program derivation above generates a C++ program on this computer. It remains several steps that could be very tricky for non computer scientists, namely the compilation of this C++ program and the execution on a cluster. To overcome these issues, at the end of the derivation process, a script is generated. It automatically pushes and compiles C++ sources to the desired cluster thanks to a configuration file and finally launches the program.

3 Conclusion and Future work

In this paper, we introduced GeoSkelSL, an efficient and simple DSL intended for geoscientists. GeoSkelSL is embedded into Python as it is a widely used language in geosciences that can be easily integrated into popular GIS such as ArcGIS or QGIS. GeoSkelSL requires no skills in parallel programming and produces

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83

efficient and scalable programs. GeoSkelSL is the top level of the GeoSkel framework. It is used to derive the C++ program from the original Python program written by the user.

In the future, we plan to extend GeoSkelSL to handle vector data which are very common data structures used in geosciences. Vector data are made off geometric features such as points, lines or polygons representing roads, rivers, forests etc. Usually, vector data is not as huge as raster data, therefore it is possible to broadcast such data instead of split them as the rasters. However, new patterns have to be implemented to be able to handle them. In the current implementation, it is possible to write programs that call several patterns but no real optimization of the workflow is done. An analysis of the program workflow would allow us to better distribute the data among the nodes to reduce communications and improve performance of the generated programs.

References

1. K. Bourgeois, S. Robert, S. Limet, and V. Essayan. Efficient implicit parallel patterns for geographic information system. In *International Conference on Computational Science, ICCS*, pages 545–554, 2017.
2. A. Van Deursen, P. Klint, and J. Visser. Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000.
3. GDAL Development Team. *GDAL - Geospatial Data Abstraction Library, Version x.x.x*. Open Source Geospatial Foundation, 201x.
4. V. Janjic, C. Brown, K. Mackenzie, K. Hammond, M. Danelutto, M. Aldinucci, and J. Daniel García. RPL: A domain-specific language for designing and implementing parallel C++ applications. In *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP*, pages 288–295, 2016.
5. C. Ledur, D. Griebler, I. Manssour, and L. G. Fernandes. Towards a domain-specific language for geospatial data visualization maps with big data sets. In *Computer Systems and Applications (AICCSA), 2015 IEEE/ACS 12th International Conference of*, pages 1–8. IEEE, 2015.
6. M. Neteler, H. M. Bowman, M. Landa, and M. Metz. Grass gis: A multi-purpose open source gis. *Environmental Modelling & Software*, 31:124–130, 2012.
7. M. Odersky, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, M. Zenger, and et al. An overview of the scala programming language. Technical report, 2004.
8. L. Rauchwerger, F. Arzu, and K. Ouchi. Standard templates adaptive parallel library (stapl). In David R. O'Hallaron, editor, *Languages, Compilers, and Run-Time Systems for Scalable Computers*, pages 402–409, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
9. A. Sujeeth, H. Lee, K. Brown, T. Rompf, H. Chafi, M. Wu, A. Atreya, M. Odersky, and K. Olukotun. Optiml: an implicitly parallel domain-specific language for machine learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 609–616, 2011.
10. D. G. Tarboton. Terrain Analysis Using Digital Elevation Models (TauDEM). *Utah Water Research Laboratory, Utah State University*, 2005.
11. QGIS Development Team et al. Qgis geographic information system. open source geospatial foundation project, 2012.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_83