Agris on-line Papers in Economics and Informatics

Volume XII Number 4, 2020

Open Source Framework for Enabling HPC and Cloud Geoprocessing Services

José Miguel Montañana[1], Paolo Marangio[2], Antonio Hervás[3]

[1] High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Germany

[2] ATOS Research and Innovation (ARI), AI, Data & Robotics Unit, Madrid, Spain

[3] Inst. Matemática Multidisciplinar (IMM), Universitat Politècnica de València, Spain

Abstract

Geoprocessing is a set of tools that can be used to efficiently address several pressing chal-lenges for the global economy ranging from agricultural productivity, the design of transport networks, to the prediction of climate change and natural disasters. This paper describes an Open Source Framework developed, within three European projects, for Ena-bling High-Performance Computing (HPC) and Cloud geoprocessing services applied to agricultural challenges. The main goals of the European Union projects EUXDAT (EUro-pean e-infrastructure for eXtreme Data Analytics in sustainable developmenT), CYBELE (fostering precision agriculture and livestock farming through secure access to large-scale HPC-enabled virtual industrial experimentation environment empowering scalable big data analytics), and EOPEN (opEn interOperable Platform for unified access and analysis of Earth observatioN data) are to enable the use of large HPC systems, as well as big data management, user-friendly access and visualization of results. In addition, these projects focus on the development of software frameworks, and fuse Earth-observation data, such as Copernicus data, with non-Earth-observation data, such as weather, environmental and social media information. In this paper, we describe the *agroclimatic-zones* pilot used to validate the framework. Finally, performance metrics collected during the execution (up to 182 times speedup with 256 MPI processes) of the pilot are presented.

Keywords

High performance computing, cloud computing, big data; agriculture, land monitoring, geoprocessing.

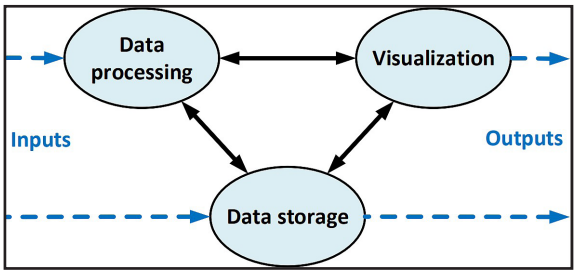
Montañana, J. M., Marangio, P. and Hervás, A. (2020) “Open Source Framework for Enabling HPC and Cloud Geoprocessing Services", *AGRIS on-line Papers in Economics and Informatics*, Vol. 12, No. 4, pp. 61-76. ISSN 1804-1930. DOI 10.7160/aol.2020.120405.

Introduction

Geoprocessing is a set of tools, generally intended for the mathematical processing carried out by a Geographic Information System (GIS). These tools consist of es-sentially three parts, as shown in Figure 1, namely data storage, computational pro-cessing, and visualization or access to re-sults.

During the last decades, the results of geo-processing have greatly improved thanks to the exponential technological progress in computational power. However, improving the efficiency of agricultural productivity requires solving the techno-logical challenge of increasing both the amount of data to be stored and the com-putational load by several orders of magnitude.

In order to tackle these challenges, during the last decades, researchers and professionals in the area have worked towards im-proving overall code performance in several ways, including parallelization of code libraries, structuring of the data, as well as balancing the computational load in clusters of computers (Figure 1).



Source: own research and processing

Figure 1: Fundamental components of geoprocessing systems and their interrelationship.

As a result, MPI and OpenMP now represent some of the most popular tools for code paralleliza-tion. And more recently, cloud computing and High-Performance Computing (HPC) have become the standard for Big Data processing. In particular, HPC systems are currently able to provide the best computing performance as well as enhanced data sharing between computing nodes (Mi-neter et al., 2000; Zhang, 2010; Li, 2020).

In the next sections, we define some theoretical and practical concepts that need to be considered for an efficient use of HPC systems.

Hardware for HPC

One important difference between HPC and Cloud systems is their interconnection network. Most modern HPC systems are clusters of Symmetric Multi-Processing (SMP) nodes with high-speed interconnection network, which eases the collaborative computation between nodes as well as the sharing of data between them. On the other hand, Cloud computing nodes have lower performance interconnection networks than HPC. Therefore, the parallelized applications running in Cloud should have less communication between nodes in order to not lose performance.

A SMP node consists of multiple identical processing elements, with identical memory access. The memory inside of the nodes allows strongly coupled processing and communication. The computation carried out among multiple nodes will have higher communication latency between cores when they are on different nodes and higher memory access latency when the data required by one node is stored in the memory of another node.

Parallelization Strategies for HPC

There are two main resources that can be distributed: the processing elements and the memory. Considering this, the parallelization of an hypothetical application will consists in deciding how to distribute the computational load among the processing elements, and how to distribute the data when using more than one node (case of distributed memory).

The distribution of the computation load and data requires defining how the internode communication will be performed, which is a very important aspect to take into account. Inefficient communication can make the processing elements stay idle while waiting for data from other processing elements or memory, which will produce an inefficient use of processing elements and therefore extend the execution time. Montañana (2010) provides more details on HPC architecture and interconnection networks.

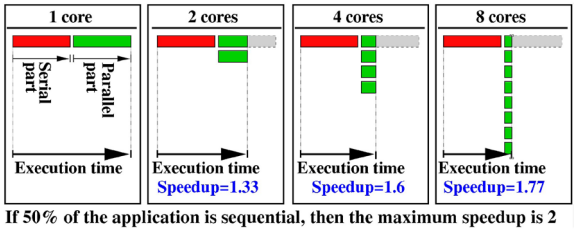
MPI and OpenMP are some of the most popular libraries and tools used for HPC systems. MPI is typically used for distributed memory systems.

Parallelization performance

A hypothetical application can be divided into two parts, namely the part that needs to be executed sequentially (s), and the part that can run in parallel (p). Equation 1 represents the decimal fraction of that distribution, where 1 stands for the total, and s and p values are between 1 and 0:

$$1 = s + p \tag{1}$$

Figure 2 describes the above concept pictorially, by showing an application that exhibits the potential of running 50% of its code in parallel (green bar in the figure). The figure shows that the total execution time becomes smaller as the number of used cores is increased. This is because the part of the code that runs in parallel is split evenly among the processors that execute their respective portion of the code concurrently.



Source: own representation of the Amdahl's law (Amdahl, 1967)

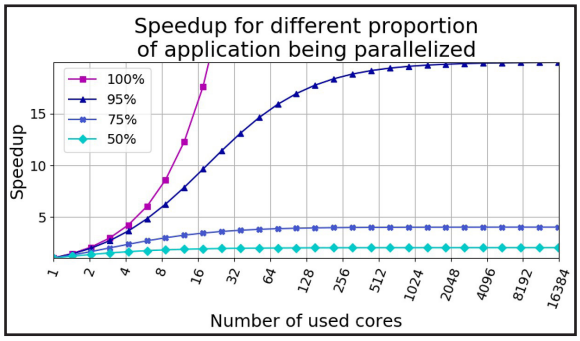
Figure 2: Reduction of execution time and increase of Speedup with the increase of the number of cores used.

The figure also illustrates the concept of *Speedup* (in latency), which corresponds to the ratio between the execution time sampled when the application is executed using only one core, and when using an increased number of them. Calculating the Speedup is useful for determining the im-pact of specific code changes on perfor-mance. Equation 2 shows how the Speedup is calculated as the division of Equation 1 for one and n processes, where n is the number of cores used, or the number of parallel executions of the part of the appli-cation that can be parallelized. This equa-tion is a formulation of Amdahl's law (Amdahl, 1967).

$$Speedup = \frac{s + \frac{p}{1}}{s + \frac{p}{n}} = \frac{1}{(1-p) + \frac{p}{n}} \tag{2}$$

From this, it follows that the execution time of a parallelized application is given by the execution time required to run the sequential part of the application plus the execution time of the parallelized part. This also means that the proportion of the application that can be executed in parallel dictates the theoretical maximum *Speedup* that can be achieved by a given parallel application.

Figure 3 shows the *Speedup* for different proportions of an application that can be parallelized. The ideal case where 100% of an application is parallelized is virtually impossible to attain due to software and hardware limitations, such as communication latencies and parallelization overheads.



Source: own processing of the Amdahl's law (Amdahl, 1967)

Figure 3: Speedup for different proportion of application being parallelized. The slopes of the curves are reduced from the point where the improvements no longer have a significant impact on the total run time. The curves saturate at the level where practically all the execution time is due to the non-parallelized part of the code.

Our experience has taught us that achieving performance improvement is relatively simple when the proportion of code being parallelized is small and the parallel code is executed on a small number of cores. However, the improvement becomes more difficult to achieve as the number of cores required for a given parallelization strategy increases. For this reason, there is a trade-off between the effort of parallelization and the performance improvements that can be obtained through parallelization. The most frequently used applications should also be the ones that receive the most parallelization efforts.

Contributions of the paper and overview of EU projects

In this paper, we significantly extend on our previous description of the Open-Source framework for enabling HPC and Cloud geoprocessing services (Montañana et al, 2020b). We will also show some preliminary results obtained by running the *agroclimatic-zones* pilot within the framework in the supercomputer Hawk at High Performance Computing Center Stuttgart (HLRS).

The main contribution of this paper is to showcase the latest developments in the creation of innovative platforms that solve several technological challenges that are relevant to geoprocessing, such as the integration of data from different origins and formats, the definition of interfaces for geoprocessing applications, and the capability of executing such applications on modern computing solutions like HPC and in the Cloud. Addressing these challenges requires a consideration of additional aspects such as allowing larger data transfer, and the enforcement of secure access and control of the data and the computational results.

All these challenges are of definite relevance to the EU projects EUXDAT (EUXDAT, 2020), EOPEN (VEOPEN, 2020), and CYBELE (Davy et al., 2020). In fact, these projects focus on developing solutions for the collection of big data from different sources, data transfer to large-scale HPC and Cloud Computing infrastructures for processing, the development of visualization tools as well as secure access to computational results.

In the next subsections, we provide a summary of the goals of the three projects:

EUXDAT

EXUDAT proposes an e-Infrastructure for enabling Large Data Analytics-as-a-Service, which addresses the problems related to the current and future huge amount of heterogeneous data to be managed and processed within the agricultural domain. EUXDAT builds on existing mature components by providing an advanced frontend, where users develop applications on top of an infrastructure based on HPC and Cloud. The frontend provides monitoring information, visualization, different distributed data analytic tools, enhanced data and processes catalogs. EUXDAT includes a large set of data connectors such as Unmanned Aerial Vehicles (UAVs), Copernicus, and field sensors for scalable analytics. Figure 4 shows the type of field sensors deployed for the EUXDAT project in farming areas. These instruments (Pessl, 2020) allow collection of a wide range of data about remote are-as, such as depth of precipitation, air tem-perature, air humidity, global radiation, wind speed, soil temperature, and leaf wetness.

Source: own processing, pictures took by authors

Figure 4: Example of a typical field sensor deployed in farming areas proposed within EUXDAT project. The sensor shown is an iMETOS 3.3 data logger developed by Pessl Instruments GmbH.

As for the brokering infrastructure, EUXDAT aims at optimizing data and resource usage. In addition to a mecha-nism for supporting data management linked to data quality evaluation, EUXDAT proposes a method to orches-trate the execution of tasks that is able to identify whether the best target for execut-ing a given application is HPC or Cloud. It uses monitoring and profiling information for making decisions based on trade-offs related to cost, data constraints, efficiency, and resource availability. During the pro-ject, EUXDAT is in contact with scientific communities, in order to identify new trends and datasets, for guiding the evolu-tion of the e-Infrastructure. The result of the project will be an integrated e-Infrastructure that encourages end-users to create new applications for sustainable development.

EUXDAT demonstrates real agriculture scenarios, land monitoring, and energy efficiency for sustainable development, as a way to support planning policies.

CYBELE

CYBELE is a European research project combining Agriculture, HPC, and Big Da-ta. It involves 31 research institutes and enterprises across EU countries. It stands for "*Fostering Precision Agriculture and Livestock Farming through Secure Access to Large-Scale HPC-Enabled Virtual Industrial Experimentation Environment Empowering Scalable Big Data Analytics*" (Perakis, 2020).

CYBELE generates innovation and creates value in the domain of agri-food, and its verticals in the sub-domains of Precision Agriculture (PA) and Precision Livestock Farming (PLF) specifically, as demonstrated by the real-life industrial cases to be supported, empower capacity building within the industrial and research community. The project aspires at demonstrating how the convergence of HPC, Big Data, Cloud Computing and the Internet of Things (IoT) can revolutionize farming, reduce food scarcity, increase food supply, bringing social, economic and environ-mental benefits. It develops large scale HPC-enabled testbeds and delivers a distributed big data management architecture and a data management strategy.

EOPEN

The objective of EOPEN is to fuse Earth Observation (EO) data with multiple, heterogeneous, and big data sources, in order to improve the monitoring capabilities of the future EO downstream sector. EO data consists of the Copernicus and Sentinel data, while the non-EO data is weather, environmental, and social media infor-mation.

The fusion between these diverse types of data is carried out at the semantic level, to provide reasoning mechanisms and in-teroperable solutions, through the seman-tic linking of information. The processing of large streams of data is based on open-source and scalable algorithms in change detection, event detection, data clustering, which are built on HPC infrastructures.

Alongside this enhanced data fusion ap-proach, an innovative architecture over-arching Joint Decision & Information Governance is combined with the tech-nical solution to assist with decision making and visual analytics. EOPEN is demonstrated through real use case scenarios in flood risk monitoring, food security, and climate change monitoring.

Materials and methods

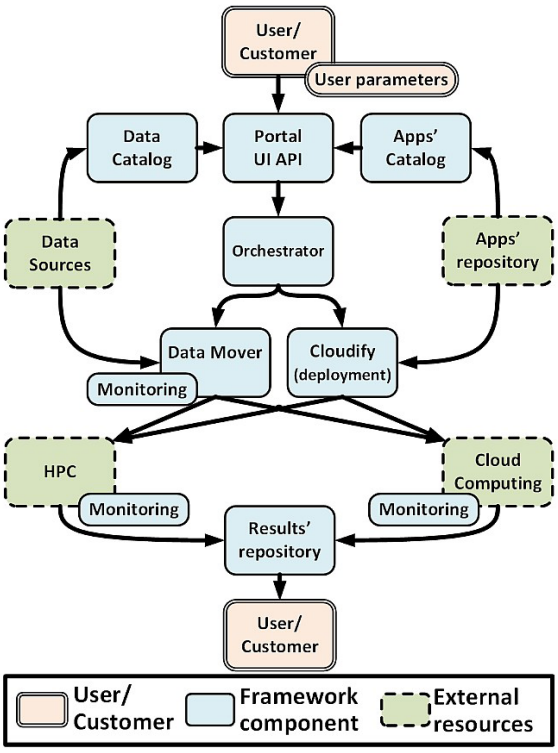
Platform implementation

The common goal shared across these projects is to develop a sustainable approach that facilitates access to data, geoprocessing applications, state-of-the-art solutions for big-data management, as well as computational resources provided by Cloud platforms and HPC centers.

Therefore, the target of the implementation of the platform is to provide an open source system that can be used on HPC and Cloud computing

systems for hosting commercial applications or products, as well as to ensure continuity in the use of a given platform after the respective EU project has been finalized. For example, although it may not be apparent, the reason for including support for accounting and billing in such platforms is to facilitate the future reuse of code. In fact, in order to ensure that a given piece of code can be successfully reused later in time, the costs of using large computer systems, as well as the cost of data acquisition from proprietary sources must be already considered during the development stage.

Figure 5 illustrates the main components of a typical infrastructure platform as well as their interrelationship. Each of the components in a given infrastructure platform has a clearly defined User Interface (UI).



Source: own research and processing

Figure 5: Representative infrastructure platform for the three EU projects described in the paper.

Portal UI API

The first component encountered by the user is the Portal User Interface (UI) Application Programming Interface (API). The portal UI API provides users with a list of available applications and the data catalog available for them. This component supports the development of applications such as mobile devices or web interfaces while abstracting away the complexity of the other components. For example, the user does not need to consider neither the complexity or format of the data, nor the different data sources, because it is encapsulated by the platform internally. Thus, once the user selects the task to perform, such as the prediction of temperature for a particular land area on a particu-lar date, the user just waits for the result. Needless to say, the time needed until a response is received is reduced by multiple orders of magnitude when using a large-scale HPC system.

Data Catalog

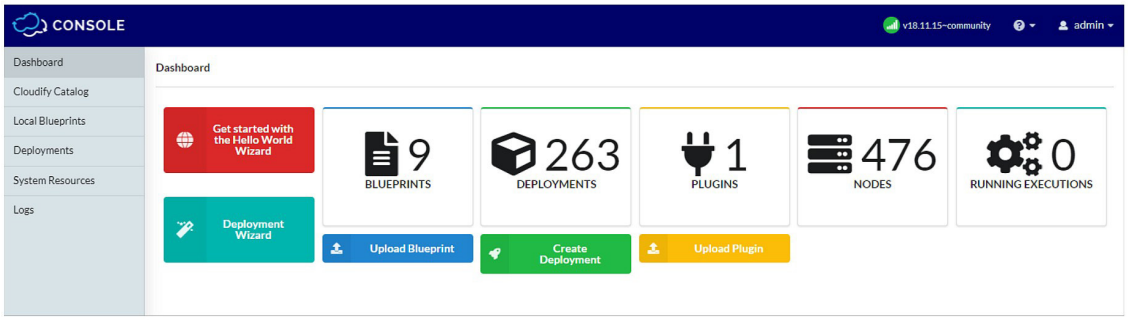
The data catalog collects data from different data sources, which may or not be free of charge. Similarly, the catalog of applications may include free applications such as those developed and hosted within the platform of a given project, or commercial applications. This is done in order to ad-vertise the platform to third parties that may wish to use its services in order to commercialize applications or data.

Orchestrator

The user request is submitted to the orchestrator, which is responsible for the transfer and execution of the applications on the computing resources. Based on the user request, the orchestrator selects the appropriate computational resource (i.e. HPC or Cloud) onto which to execute the task. Orchestrators such as Cloudify typically use an application model Domain Specific Language (DSL) based on Topology and Orchestration Specification for Cloud Applications (TOSCA), which encourages modularity of applications. This is encoded in the `blueprint' files, in which the orchestrator packages the specifica-tions of the user parameters, the input and output files, as well as the binary files to be transferred into the computational re-sources. A fragment of a blueprint is shown in Listing 1. In particular, Cloudify provides a user-friendly GUI that allows to easily manage blueprint files and associat-ed deployments (Figure 6). Moreover, the blueprint file allows the orchestrator to delegate the required staging of input and output data to the Data Mover component.

For instance, the *data_mover_options* fields in the blueprint specify the files to be transferred, as well as the source, desti-nation, and the user credentials.

In addition, the orchestrator's API allows the execution to be requested from a user-friendly web interface as shown the next sections.



Source: Screenshot from http://cloudify-api.test.euxdat.eu/console

Figure 6: Cloudify GUI. Cloudify offers an intuitive GUI that enables users to upload blueprints, create deployment. The GUI also gives an overview of the number of compute nodes and running execution.

Hybrid orchestrator

Many applications that run on HPC are usually part of bigger workflows that run in the cloud, such as those with tightly-coupled data or extensive big data analytics. In a similar fashion as for micro-services applications running solely in the Cloud, an automatic hybrid deployment and management of a modularized application in HPC and Cloud is expected to optimize the overall performance and enable new development architectures. In order to address this technical gap, the AI, Data & Robotics Unit (former *Advanced Parallel Computing lab*) Lab at ATOS Research and Innovation (ARI) Spain has recently developed within the EUXDAT project a plugin for Cloudify called Crou-pier (Carnero and Nieto, 2018) written in Python that adapts the Cloudify orchestrator algorithm for the management of HPC resources, so that Cloudify can orchestrate a hybrid environment including both Cloud and HPC. Croupier has been developed to essentially bring the latest and greatest features that have been enabled by Cloud architectures, such as modularity, interoperability, software as a service (SaaS), infrastructure as code (IaC), con-tinuous integration and deployment (CI/CD), to the world of HPC. Thanks to Croupier, it is for example possible to run batch applications on both HPC and Cloud.

Data mover

The size of the data files requires an efficient transfer method for transferring them into the computational resources.The current network protocol used for such pur-poses in centers like HLRS is GridFTP. Published results show that GridFTP pro-vides better performance, between 5 and 10 times faster, than the standard FTP protocol (Esposito et al., 2003). Other ad-vantages of using GridFTP include the security provided based on x509 certificates and the capability to carry out third party transfers (Figure 7).

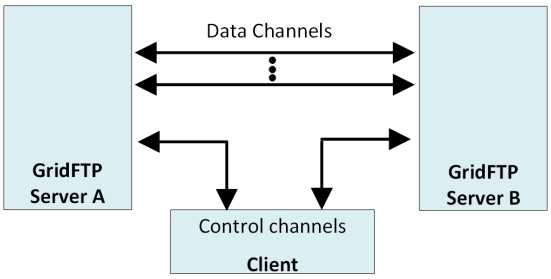
```
node_templates:
job:
    type: croupier.nodes.job
    job_options:
        type: "SRUN"
        command:"coordinates.txt"
        nodes: 100
        max_time: "04:00:00"
data_mover_options:
    workspace: wsdata
    create_ws: TRUE
    source: "ATOS"
    destination: "HLRS"
    source_input: demo_cloud_folder
    dest_output: demo_hpc_folder
    grid_userkey:
        --BEGIN ENCRYPTED PRIVATE KEY--
        ...
        --END ENCRYPTED PRIVATE KEY--
    grid_usercert:
        -----BEGIN CERTIFICATE-----
        ...
        -----END CERTIFICATE-----
```

Source: own research and processing

Figure 7: Example of a fragment of a TOSCA blueprint file.

Figure 8 shows the main elements and their communication in a GrdiFTP third party transfer. The concept consists of a client point from where the transfer is re-quested, but the data does not have to traverse it as in other protocols. Instead, the data is directly transferred between servers (servers A and B in the figure) using multiple communication channels in parallel. This protocol already proved better performance on data transfers performed over the internet because it allows using the high bandwidth available in the communication channels in the servers, by multiplexing the traffic over multiple channels in the internet that have lower bandwidth. It is highly desired to avoid using any intermediate storage, because it requires storage space, which slows down the data transfer. For that reason, we ex-plored the state of the art on large-scale data management solutions that support the GridFTP network protocol.

The most advanced tool for using GridFTP that we could find was Rucio. Rucio is an open source tool developed within the context of the ATLAS project for manag-ing big data at the European Organization for Nuclear Research (CERN). It is cur-rently used to transfer more than one petabyteworth of data per day, and more than one million files per day (Serfon, C. et al., 2019). However, after performing detailed evaluation we identified two ma-jor issues in using Rucio and therefore decided to develop our own plugin instead. The first issue is that Rucio does not allow running third party transfers. The second one is that Rucio cannot handle recursive directory transfers, when defining the des-tination folder (non-deterministic model). Although an alternative exists (determinis-tic model, where Rucio creates folders labeled with an alphanumeric string), it still makes it difficult to integrate the tool with our framework.



Source: own figure based on the GridFTP protocol (www.globus.org)

Figure 8: GridFTP direct third party transfer. Client requests a site-to-site direct data transfer. The transfer is multiplexed among multiple channels.

The development of the Data Mover plugin in place of usring Rucio allowed us to fulfill our project requirements as well as its integration with the other compo-nents. Its implementation is based on the use of the Globus-Client (SURFsara, 2015) and uberFTP (NCSA, 2020). The Data Mover plugin has been developed within the EUXDAT project and is now an inte-gral part of the Croupier plugin for Cloudi-fy (Montañana and Gorroñogoitia, 2020a). The Data Mover plugin supports GridFTP direct data transfers between data sources and computational resources without intermediate staging, as required for the previously described pilot and use cases.

HPC and Cloud Computing

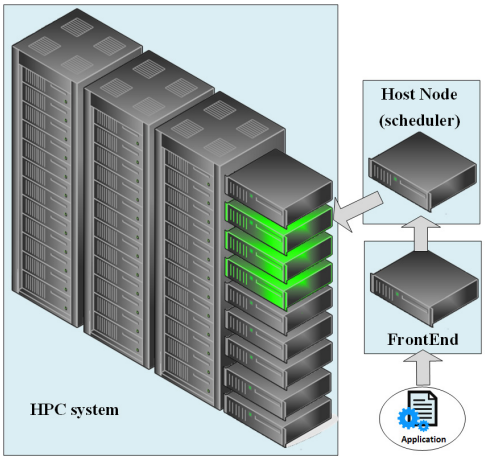
HPC and Cloud computing systems exhibit differences in terms of performance and cost. Moreover, the implementation of a given application on either of these sys-tems may differ significantly in order to achieve the best performance and resource utilization.

An HPC system consists of a large number of compute nodes that are physically close to each other (ideally all in the same room) with a high-performance interconnection network running between them. Figure 9 outlines the different nodes a user application goes through when it is submitted to an HPC system.

HPC systems rely on low-latency communication to share computational results between compute nodes. Moreover, HPC systems tend to have higher costs than Cloud systems for executing applications, due to the high cost of the interconnection network as well as the maintenance cost incurred due to having to run cooling sys-tems needed to dissipate the large heat generated by the hardware that is physically located in a relatively small space.

On the other hand, Cloud computing systems consist of different types of physical hardware (e.g., networking equipment, load balancers, servers) that can be located in different geographical locations. Virtual-ization is also typically employed in such systems in order to connect servers together, and also to divide and abstract resources in order to make them accessible to users. Cloud computing systems typically do not require a high-performance interconnection network between compute nodes, and as these are not located in the same space Cloud computing systems do not incur additional costs for cooling systems.

Applications executed in HPC centers typically show better performance than those executed in the Cloud. However, in order to make efficient use of resources additional effort is needed in order to `parallelize' applications. Parallelizing an application essentially entails spreading the workload among different computing nodes. On the other hand, applications to be executed in the Cloud can be easier to implement as it is possible to simply submit replicas of the same application as different independent jobs, each targeting a different portion of input data.



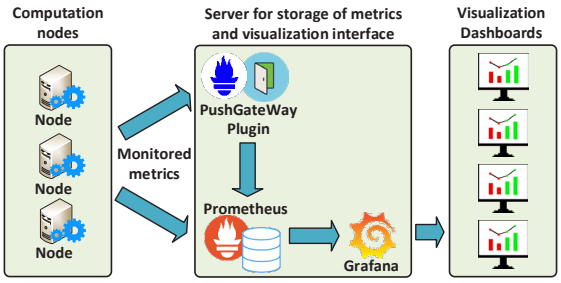
Source: own research and processing

Figure 9: Different nodes involved in the execution of an application in an HPC system. User uploads the application on a `frontend' node. The job is then dispatched to a scheduler node that manages where and when it will be run. The computation nodes, highlighted in green, are responsible for running jobs.

Although using HPC systems may lead to greater performance than Cloud computing systems, it requires higher implementation efforts and costs for execution. Hence, there is not a general agreement on which system to run a given application on, and it is up to the user to decide what to trade between implementation effort and execution cost.

Monitoring

In order to improve future application executions, the metrics related to the utilization of different resources are registered into a monitoring server such as Prometheus. Using an open source system moni-toring and alerting toolkit such as Prome-theus facilitates the decision of where to allocate future task requests depending on specific user constraints, such as reduced computation time or cost. Figure 10 shows the overall monitoring process, which includes collection of metrics at the computation nodes, storing the metrics in a Prometheus server (with metrics being pushed by the Pushgateway plugin to the server if they cannot be directly sampled at the computation nodes by Prometheus), and accessing the stored data through the Grafana visualization interface. This system allows to inspect metrics in real time as soon as they are stored in Prometheus.



Source: own research and processing. Logos of Pushgateway, Prometheus, Grafano took from https://prometheus.io and https://grafana.com

Figure 10: Collection and access to the monitored metrics.

Once the computation is completed and the monitoring metrics have been collected, the results are moved into a repository that can be accessed by the user, and the user is notified.

Use cases and pilots

The three EU projects presented above are focused on the development and testing of solutions for the field of agriculture. Agriculture is a key player in economic and political stability.

Because of its importance, governments are funding the development of solutions data access systems, geoprocessing, and tools for decision making.

The different uses cases demonstrate the capacity of the HPC solutions proposed across the projects.

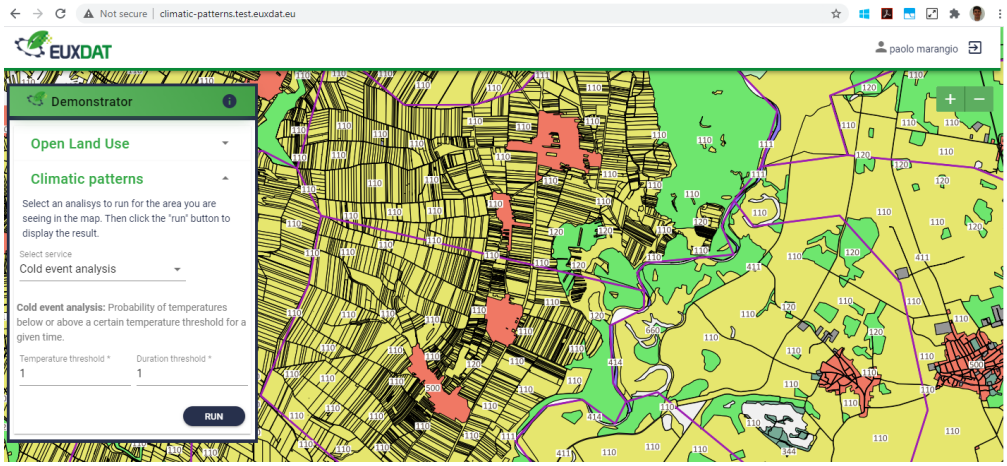
The use cases cover a wide range of real-life applications ranging from detection of weather conditions, humidity or crop dis-eases, to precision agriculture, livestock farming, and exploration. In the next sections, we provide a brief description of a selection of use cases. It should be noted that access to the implementations of the use cases is currently limited to consorti-um partners. Towards the end of the respective projects, the use cases will be made available for the end users commu-nities.

OpenLand monitoring and sustainable magement

This use case aims at developing a deep learning algorithm that uses a range of input data for predicting soil and crop status. The input data includes images generated by multirotor UAV systems with a hyperspectral camera as well as Earth observation and meteorological data.

3D Farming

This use case focuses on analytics models within the context of spatial analysis for farming (e.g., locating the highest productivity zones). It will



Source: screenshot from http://climatic-patterns.test.euxdat.eu

Figure 11: Web interface of the OpenLand monitoring pilot.

provide 3D visualization for the obtained results, which is especially helpful for understanding different soil parameters such as amount of water, and solved particles and nutrients (Figure 11).

Organic soya yield and protein content prediction

There is a strong interest in the predictive analytics of soybean farming, mainly because the EU is strongly dependent on other continents for sourcing plantbased proteins. In light of this, this use case develops methods for predicting maps indicating soybean yield and protein-content based on crowd-sourced data, satellite imagery, and additional information if available, such as electromagnetic soil scans and other sensory data.

Climate-smart predictive models for viticulture

This use case addresses the development of complex, highly-nonlinear models for vine and grape growth, which rely on a large number of variables that have been shown to affect the quality and quantity of the produced yields. The range of input data includes soil/elevation maps, earth observations, genomics, chemical analysis, environmental and climatic data.

Climate services for organic fruit production

This use case aims at helping with the prevention of damaging effects caused by frost and hail. The solution under development focuses on providing risk probability mapping calculated based on models obtained by machine learning techniques.

In order to train the predictive model, a wide range of data sources are used including but not limited to climate instability indices, digital terrain models, in-situ environmental and climatic data, and satellite images.

Optimizing computations for crop yield forecasting

This use case aims at developing a crop yield monitoring tool that can be used for agricultural monitoring (e.g., early warning and anomaly detection), index-based insurance (index estimates), and farmer advisory services. Its goal is to compute a productivity estimation based on cropping systems model and a combination of different datasets, such as ingest crop, soil, historic weather, and weather forecasts data. The computation underlying this use case becomes more challenging as the amount and resolution of available data are increased.

Evaluation

Next, we describe the pilot used to validate the framework. Results of the perfor-mance of the pilot are shown after that.

Case study: Agroclimatic-zones pilot

In this section, we focus on the *agroclimatic-zones pilot* for the validation of the framework proposed in this paper. We decided to use this pilot because at the time of writing it was one of the most mature pilots available across all the projects. From a computing point of view, the algo-rithm associated with this use case is also relatively 'simple' while still holding the potential to be parallelized using one of the parallel programming frameworks for execution on HPC described above.

Currently, the available maps of climatic zones are very generic and exhibit low granularity. Although they are able to display some differences in topography between areas, the areas shown by such maps tend to be quite large and do not include, for example, seaside buffer zones, weather divides, and South-North differences.

The idea of this algorithm is to create a classification system that allows to characterize land areas as different *agroclimatic* groups, based on long-term climate data, land cover, and topography information.

The goal of the algorithm is to generate local climate maps that take into account general weather conditions (large-scale weather models), local topography (with North/South slopes), buffer effects (such as lakes, sea, or swamps) and soil types.

The tool is primarily intended for:

- Agricultural extension counselors, or technical farm organizations wishing to make an investment in frost protection, irrigation, etc.
- Insurance and other financial institutions wishing to make decisions on quality and risk of agricultural invest-ments.
- Researchers interested in making decisions related to field trial (climatic) representativeness.
- Researchers and advisors interested in checking the impact of climate change on a given area and making decisions about future management strategies.

The expected frequency of use for the tool is once per year, while the type of data queries could be either local or regional (e.g., comparisons of several sites). By using the proposed tool it will be possible to predict long-term climate changes, which will in turn allow to make better-informed long-term decisions about crops and use resources more efficiently. One example of this is frost protection. In the past few years, significant parts of the Central EU Orchard and Vineyard industry have been affected by late frosts, which required making critical decisions about anti-frost protection measures, varietal changes, and risk mitigation strategies (Vitasse and Rebetz, 2018). In the rest of the paper, the application of the tool for computing frost-related information that may be useful in frost protection and management will be presented.

Inputs to the algorithm

The algorithm takes as inputs two different types of data as well as a set of input parameters provided by the user. The first piece of input data is meteorological data in the ERA5-Land (ECMWF/CDS) or NEMS30 (Meteoblue AG) format/model. The data is encoded in NetCDF files used as input data in the algorithm. The second piece of input data is optional and it includes topography maps (EU-DEM format) and land cover/soil maps (Joint Research Center or Open Land Use Map).

The input parameters provided by the users are:

- Area of interest encoded by polygon drawn over map presented to the user
- Start year
- End year
- Probability of first/last frost day
- Frost temperature (in degree Celsius)
- Daily hours with minimum tempera-ture: start hour, end hour (0-23)
- Length of stretch of last and first frost days to be found for a single year

Output of the algorithm

The output of the algorithm is a set of values corresponding to *agroclimatic* variables calculated based on the input data and user input parameters. For every point in the polygon specified by the user, the algorithm computes the following:

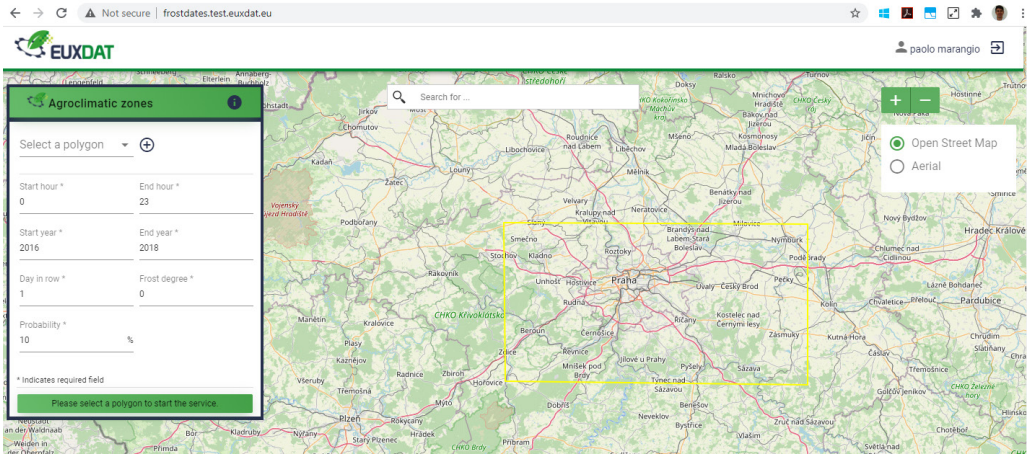
- Last spring frost date
- First fall frost date
- Length of frost-free season
- Number of frost days
- Average number of frost days for the period spanning the start and end year

The output information is stored as a GeoJSON file.

Proposed parallelization strategy

At the smallest level, the *agroclimatic-zones* algorithm calculates, for a given position on the map identified by a pair of latitude and longitude values, the first and last frost date for each year over a user-defined range of years. From a programmatic point of view (Figure 14). This is achieved through the sequential invocation of three nested functions, with information for a single year and grid point in latitude and longitude dimensional space being computed by the innermost function (*findfrostdates*), followed by an intermedi-ate function that aggregates this information across multiple years (*frostdateyearly*). Finally, the outermost function invokes the two inner functions over mul-tiple points in latitude and longitude dimensional space (*frostdatesplaces*). Since for a typical execution of the algorithm the number of years (maximum value is 37) tends to be much smaller than the total number of pairs of latitude /longitude val-ues (maximum value for this study was 70 longitude values x 27 latitude values = 1890 grid points), it seemed more intuitive to perform a parallelization of the outermost function *frostdatesplaces*, such that each MPI process would essentially be responsible for calculating the output in-formation for an independent pair of latitude/longitude values. Parallelizing this function would not break the code or cause bottlenecks because the computations for different pairs of latitude/longitude values are independent and can, therefore, be performed asynchronously. Hence, we essentially used MPI to schedule parallel execution of an identical computation (with the only difference being the input grid point of latitude/longitude values) over multiple processes, rather than using point-to-point communication for directly improving the performance of the serial/sequential *agroclimatic-zones* algorithm.

Even if the serial application was written in Python, introducing the proposed changes was easily achieved by using the Message Passing Interface (MPI) for Python package (Dalcin, 2019) (Figure 12).



Source: screenshot from http://frostdates.test.euxdat.eu

Figure 12: Web interface of the agroclimaatic zones pilot.

Results and discussion

Application requirements evaluation for execution of parallel agroclimatic-zones algorithm in HPC

The resource requirements for the *agroclimatic-zones* can be evaluated considering the infrastructure in which the algorithm will be implemented and finally executed. All of the projects have access to the Hewlett Packard Enterprise Apollo 9000 Hawk supercomputer available at the HPC Center in Stuttgart (HRLS). Table 1 shows the key features of Hawk, which was launched in February 2020.

Name	HPE Apollo 9000 Hawk
Number of node	5,632
Number of cores	720,896
Peak performance	26 Petaflops
Disk storage capacity	25 PB
Interconnection net	InfiniBand HDR (200Gbit/s)
Power consumption	2112 KW, to be increased

Source: https://www.hlrs.de/systems/hpe-apollo-9000-hawk

Table 1: Characteristics of the HLRS Hawk HPC system.

The simultaneous use of HPC systems by a large number of users requires that each user's execution request includes a specification of the number of computing nodes and software to be used. After a request for execution is submitted, it is queued until all the resources required for the execution become available. Most HPC centers need to have very high usage in order to be viable. This, however, means that the waiting time for the execution of a given application can range from a few minutes to a few days depending on the workload of the HPC center and the resource requirements specified in a given request. Obviously, the user is only billed for effective compu-tation time, not for time spent waiting in the queue. Hence, the required computation time is an important aspect to consider when executing geoprocessing applications in such infrastructures. In this respect, an option is to upload the required data to the HPC system prior to execution as this can save a significant amount of computation time (and in turn cost).

```
input:
  startlat, startlon, endlat, endlon, startyear, en-dyear,
  probability, frostdegree, starthourday, endhourday,
  dayinrow
output:
  firstfrosday, lastfrostday frostfreeperiod,
  numbfrostdays, avgnumbfrostdays
begin
  call function frostdatesplaces
    loop over lat & lon
      call function frostdatesyearly
        initialize nmbfrdayslist = empty list
        loop over years
          call function findfrostdates
            initialize numbfrostdays = 0
            initialize lastfrostday = 0
            initialize firstfrostday = 0
            loop over days between Jan and Jul
              initialize daymin = 50
              loop over hours
                calculate currentemp
                if currenttemp < daymin:
                  daymin = currenttemp
```

Source: own research and processing

Figure 13: Pseudocode of serial agroclimatic-zones algorithm (to be continued).

```
    if daymin <= frostdegree:
      numbfrostdays += 1
  output lastfrostday
  loop over days between Jul and Jan
    initialise daymin = 50
    loop over hours
      calculate currentemp
      if currenttemp < daymin:
        daymin = currenttemp
    if daymin <= frostdegree:
      numbfrostdays += 1
    output firstfrostday
    output numbfrostdays
    frostfreeperiod = (see next line)
    firstfrostday - lastfrostday
    output frostfreeperiod
    append numbfrostdays to nmbfrdayslist
  avgnumbfrostdays = mean(nmbfrdayslist)
  output avgnumbfrostdays
```

Source: own research and processing

Figure 13: Pseudocode of serial agroclimatic-zones algorithm (continuation).

Table 2 shows the preliminary requirements of two applications belonging to use cases from different projects. We chose these applications because their resource requirements are common among all of the listed use cases. In particular, the estimated size of data to be transferred and the computational load of the applications for computing *agroclimatic-zones* and land morphometry characteristics are shown. Since a use case is composed of a series of geoprocessing applications, the computational and data storage requirements of a use case presented here correspond to the accumulation of the analyzed requirements of the individual constituent applications.

While consideration must be placed in evaluating application requirements before execution, the execution time of an application is also an important aspect that factors in the decision of which infrastructure the application should ultimately be run in. For instance, if a farmer needed to know whether the next morning's temperature was going to be below 27 degrees (Muhollem, 2017), the farmer would need to receive the application output before the morning would come in order to successfully safeguard his blossoming crop. For such applications, it is more suitable to carry out the execution on HPC rather than Cloud since the computation will be completed earlier on HPC (assuming a suitable level of parallelization has been introduced).

Applications	**Agroclimatic-zones frost date calculation**	**Morphometry characteristic calculation**
Storage requirements	316 MB (ERA5-Land Czechia)	25 GB (Austria Area) 1 TB (Full Europe)
Computation time in core-hours	70 (Czechia)	3000 (Full Europe)

Source: Pavel Hájek (http://www.wirelessinfo.cz) and Dr. Karl Gutbrod (https://meteoblue.com)

Table 2: Requirements of selected applications.

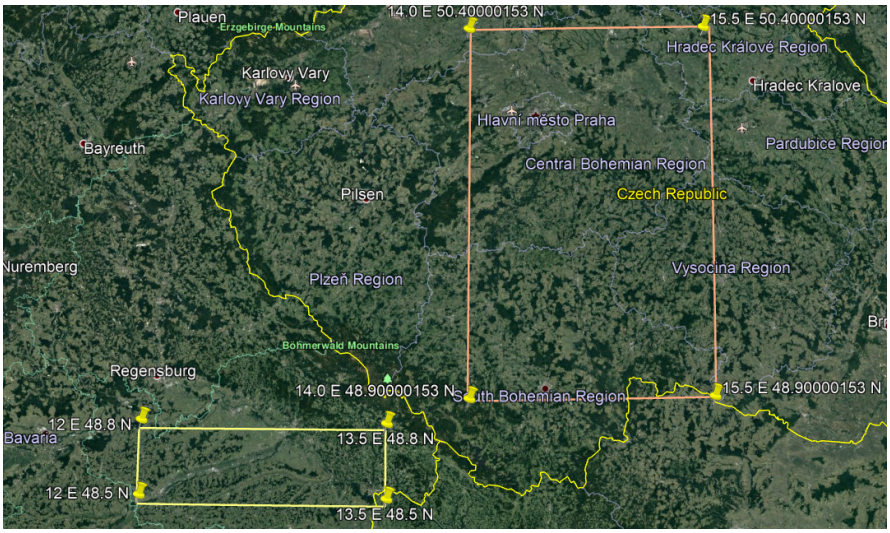
Results of benchmarking tests of parallel *agroclimatic-zones* algorithm

In order to properly benchmark the paralel *agroclimatic-zone*s algorithm, we strived to test it on an input with size and complexity consistent with those to be found when the algorithm would be in production. This required making some choices in terms of the number of years worth of data and the number of pairs of latitude/longitude values to be processed.

After some experimentation, it was found that the serial agroclimatic-zones algo-rithm was able to process 12-15 years worth of input data and 64-256 pairs of latitude/longitude values in the order of 2-85 hours. This range of execution time and problem size seemed like a reasonable starting point for experimentation, while also allowing us to more clearly showcase the capability of HPC to improve perfor-mance of a typical medium to large scale application. Therefore, all benchmarking experiments presented here were conducted with either 64 (i.e., smaller input problem size) or 256 (i.e., larger input problem size) pairs of latitude/longitude values (Figure 14) and 12 years worth of input data.

Benchmarking of the parallel algorithm was performed in HLRS Hawk. A standard compute node from this infrastructure con-sists of a single 2.25GHz, 64-core AMD Epyc Rome 7742 processors with 256 GB memory. Each of the cores in the proces-sor supports 2 hardware threads (also known as hyperthreading), meaning that a single node can execute up to 128 threads.

In order to determine the performance of the parallel algorithm, the total execution time of the program (after all MPI processes had finished execution) was measured. The parallel algorithm was run on 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512 MPI processes on one node with the exception of 256 and 512 processes which were run on 2 an 4 nodes. As shown in Figure 14, the parallel algorithm ran with 64 pairs on input data reached the lowest execution time of approximately 2min 30 sec



Source: Screenshot from http://climatic-patterns.test.euxdat.eu

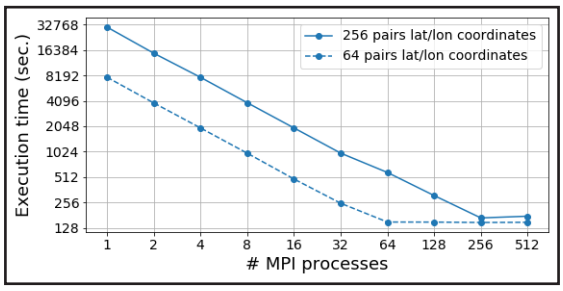
Figure 14: Map showing boundaries of grid of latitude and longitude values processed by the *agroclimatic-zones* algorithm representing a relatively small (shown by yellow rectangle) and large (shown by orange rectable) problem size in this study. Assuming a spatial resolution of 0.1x0.1° between these boundary values, the yellow rectangle corresponds to 64 pairs of latitude/ longitude values (or grid points), while the orange rectable correspondes to 256 pairs of latitude/ longitude values.

with 64 MPI processes. When run with a input problem size of 256 pairs of lati-tude/longitude values, the parallel algorithm managed to record the same low execution time as with the smaller input problem size, but using 256 MPI processes. Considering that the serial algorithm reported an execution time of approxi-mately 2h10min with 64 input pairs of latitude/longitude values, and 8h with 256 pairs, the parallel algorithm achieved a maximum speedup of approximately 52 times with the smaller input problem size, and speedup of approximately 182 times with the larger input problem size.

Overview of metrics collected during program execution in HLRS Hawk

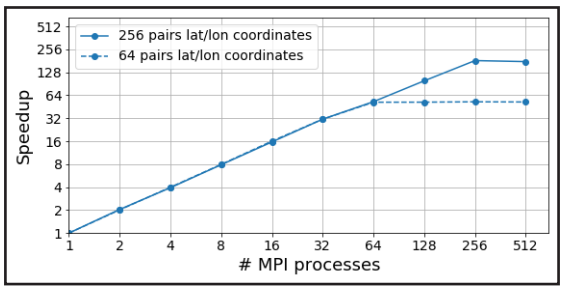
The metrics collected in Prometheus in the current implementation of the parallel *agroclimatic-zones* algorithm are the amount of data transferred by the Data Mover, and the average bandwidth on each request of transfer of a set of files (i.e., it can consist of multiple files or a single one). Additional metrics related to the performance of the applications are also automatically collected.

The Data Mover was tested by transferring test files with different sizes (i.e., 100MB, 1GB, and 10 GB) between the Cloud resources of ATOS in France and HPC resources of HLRS in Germany. The test files contained random data in order to avoid data compresion. The typical band-width measured with GridFTP transfers was between 70 and 90 MB/s. However, it is not possible to use these results to pre-dict the future performance of a given transfer because it will ultimately depend on the network load in the internet as well as in the data centers (Figure 15 and 16).



Source: own research and processing

Figure 15: Number of MPI processes vs. Execution time. Input data is 64 (shown by dashed line)or 256 (shown by the solid line) pairs of latitude/longitude values and 12 years worth of ERA5-Land data.



Source: own research and processing

Figure 16: Number of MPI processes vs. Speedup. Input data is 64 (shown by dashed line) or 256 (shown by the solid line) pairs of latitude/longitude values and 12 years worth of ERA5-Land data.

Analysis of results

The infrastructure platform proposed within the context of the EU projects discussed in the paper currently satisfies all the re-source requirements for the agroclimatic-zones pilot, and no deficiencies could be detected.

The benchmarking tests of the parallel algorithm used for deploying the *agroclimatic-zones* pilot have clearly demonstrated the power of HPC for parallelizing data-intensive applications. Thanks to the proposed parallelization strategy and by running the application on multiple compute nodes of an HPC cluster, the final parallel algorithm was 52 times faster than the original sequential program for a relatively small yet realistic input size. Put in a different perspective, these results al-ready means that a user can now execute his/her program in a couple of minutes instead of several hours. Considering that an even greater speedup (i.e., approximately 182 times faster than sequential pro-gram) was achieved when problem size was quadrupled (i.e., 256 pairs of latitude/ longitude values is doubled) as well as the number of MPI processes, these results indicate that the parallel algorithm scales well with problem size. This strong-ly suggests that the newly developed algo-rithm presented in this paper is a very efficient, parallel algorithm that should be of general interest to the geoprocessing community.

Based on the results and the evaluation presented in this study it can be argued that the proposed framework simplifies the deployment and execution of geoprocessing tasks. Thanks to its data moving approach and the use of HPC resources, the framework is able to achieve an efficient transfer of data and computation in a significantly smaller amount of time, therefore also reducing costs. Based on the current body of knowledge the proposed framework seems to be very cost-effective for geoprocessing and is particularly suitable for large projects such as large scale studies conducted by governments. It is also attractive for companies interested in selling the results of geopro-cessing to small customers that do not have access to the data or the software necessary for running applications by themselves.

Conclusions

In this paper, an open source framework underpinned by an infrastructure suitable for HPC and Cloud computing of geopro-cessing services has been described. We have demonstrated that the infrastructure can support the execution of realistic use cases within the context of several EU projects, and achieve large speedup (up to 182 times) when running data-intensive applications.

The solutions being developed by the EU projectsshowcased in the paper will greatly support improving farming performance and competitiveness. This is not only because the developed tools are fit for purpose, but also because they leverage time-efficient computational resources. These tools will exhibit a simplified access for non-technical users. They are attractive also for customers that do not have access to the data, software or hardware needed. Moreover, the intention is that the developed platforms will stay operational after the end of the respective projects. In particular, the partners in the projects are in-terested in using them for selling their products, such as datasets and weather forecasting services directly to farmers after the respective projects are over. In order to ensure this, the consortium part-ners are committed to perform the roles of software, HPC and Cloud platform provid-ers after the projects are over.

Additionally, it should be noted that the developed platform for agriculture geo-processing is also suitable for other pur-poses than agriculture, such as providing optimum paths through transportation networks, predicting disasters like wildfire and flooding, or the effects of a storm. Considering this broader scope, potential users can therefore also include local au-thorities interested in urban and regional planning and water management, or insurance companies interested in risk prevention or disaster resilience.

Acknowledgments

This work has been carried out within the context of the following projects: *European e-infrastructure for extreme data ana-lytics in sustainable development* (EUXDAT); *Fostering precision agricul-ture and livestock farming through secure access to large-scale HPC-enabled virtual industrial experimentation environment empowering scalable big data analytic* (CYBELE); *Open interoperable platform for unified access and analysis of Earth observation data* (EOPEN).

Further information about the projects is available at the respective web pages (Nieto et al., 2020; Vingione et al., 2020; Davy et al., 2020). The research leading to these results has received funding from the European Unions Horizon 2020 Research and Innovation Programme, grant agreements n. 777549, 825355, 776019, respectively.

We wish to thank Dimitrij Kozuch (Plan4all), Pavel Hájek (WirelessInfo), Jiří Valeš (University of West Bohemia) and Dr. Karl Gutbrod (Meteoblue AG) for providing the applications described or showcased in this paper, as well as the estimation of their requirements. We wish to thank Naweiluo Zhou for providing an up to date description of the CYBELE project. We would also like to thank the (30+) research institutes and enterprises across several EU countries that have or are still collaborating on these three Euro-pean projects for their valuable work.

Corresponding authors
José Miguel Montañana Aliaga, Ph.D, Senior Researcher
Höchstleistungsrechenzentrum Stuttgart (HLRS)
Nobelstraße 19, 70569 Stuttgart, Germany
E-mail: jmmontanana@gmail.com

References

[1] Amdahl, G. M. (1967) "Validity of the single processor approach to achieving large scale computing capabilities", In *Proceedings of the Spring Joint Computer Conference (NY, USA), AFIPS '67 (Spring)*, Association for Computing Machinery, p. 483-485. DOI 10.1145/1465482.1465560.

[2] Carnero, J. and Nieto, F. J. (2018) "Running simulations in HPC and cloud resources by implementing enhanced Tosca workflows", In *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 431-438. DOI 10.1109/HPCS.2018.00075.

[3] Dalcin, L. (2019) "*MPI for Python*". [Online]. Available: https://mpi4py.readthedocs.io/en/stable/index.html [Accessed: 15 Sept. 2020].

[4] Davy, S. (2020) "*CYBELE Fostering Precision Agriculture And Livestock Farming Through Secure Access To Large-Scale Hpc-Enabled Virtual Industrial Experimentation Environment Empowering Scalable Big Data Analytic*". [Online]. Available: https://www.cybele-project.eu [Accessed: 15 Sept. 2020].

[5] Esposito, R., Mastroserio, P., Tortone, G. and Taurino, F. M. (2003) "Standard FTP and GridFTP protocols for international data transfer in Pamela Satellite Space Experiment. In *Proceedings from the 13th International Conference on Computing in High-Enery and Nuclear Physics (CHEP 2003).*

[6] Li, Z. (2020) "Geospatial Big Data Handling with High Performance Computing: Current Approaches and Future Directions", In *High Performance Computing for Geospatial Applications*, pp. 53-76. E-ISBN 978-3-030-47998-5, ISBN 978-3-030-47997-8. DOI 10.1007/978-3-030-47998-5_4.

[7] Mineter, M. J., Dowers, S. and Gittings, B. M. (2000) "Towards a HPC Framework for Integrated Processing of Geographical Data: Encapsulating the Complexity of Parallel Algorithms", *Transactions in GIS,* Vol. 4, No. 3, pp. 245-262. E-ISSN 1467-9671. DOI 10.1111/1467-9671.00052.

[8] Montañana, J. M. (2010) "Providing Fault Tolerance in Interconnection Networks for PC Clusters: Efficient Mechanisms", Lap Lambert Academic Publishing. ISBN-13 978-3838318905, ISBN-10 9783838318905.

[9] Montañana, J. M. and Gorroñogoitia, J. (2020a) "*Data mover plugin provides support for GridFTP data transfers to croupier cloudify orchestrator*". [Online]. Available: https://github.com/ari-apc-lab/croupier/tree/euxdat/croupier_plugin/data_mover [Accessed: 20 Sept. 2020].

[10] Montañana, J. M., Hervás, A. and Hoppe, D. (2020b) "HPC-Enabled Geoprocessing Services Cases: EUXDAT, EOPEN, and CYBELE European Frameworks", In *Proccedings of the 12th International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOProcessing)*, pp 31-35.

[11] Muhollem, J. (2017) "*Warm winter has put state's apple crop at risk, expert warns", Pennsylvania State University*. [Online]. Available: https://phys.org/news/2017-03-winter-state-apple-crop-expert.html [Accessed: 20 Sept. 2020].

[12] NCSA (2020) "*uberftp - GridFTP-enabled client*". Linux man page. [Online]. Available: https://linux.die.net/man/1/uberftp [Accessed: 20 Sept. 2020].

[13] EUXDAT (2020) "*EUXDAT European e-Infrastructure for Extreme Data Analytics in Sustainable Development*". [Online]. Available: https://www.euxdat.eu [Accessed: 20 Sept. 2020].

[14] Perakis, K., Lampathaki, F., Nikas, K., Georgiou, Y., Marko, O. and Maselyne, J. (2020) "CYBELE – Fostering precision agriculture & livestock farming through secure access to large-scale HPC enabled virtual industrial experimentation environments fostering scalable big data analytics", *Computer Networks*, Vol. 168, ISSN 1389-1286. DOI 10.1016/j.comnet.2019.107035.

[15] PESSL INSTRUMENTS GMBH (2020) "*Stations and datalogger*". [Online]. Available: http://metos.at/micrometos-clima [Accessed: 15 Aug. 2020].

[16] Serfon, C., Barisits, M., Beermann, T., Garonne, V., Goossens, L., Lassnig, M., Nairz, A. and Vigne, R., ATLAS Collaboration (2019) "Rucio, the next-generation Data Management system in ATLAS", *Nuclear and Particle Physics Proceedings*, Vol. 273-275, pp. 969-975. ISSN 2405-6014. DOI 10.1016/j.nuclphysbps.2015.09.151.

[17] SURFsara (2015) "*Globus client. Grid Documentation v1.0.*" [Online]. Available: http://doc.grid.surfsara.nl/en/latest/Pages/Advanced/storage_clients/globus.html [Accessed: 15 Aug. 2020].

[18] EOPEN (2020) "*EOPEN Open interoperable platform for unified access and analysis of earth observation data*". [Online]. Available: https://eopen-project.eu [Accessed: 20 Sept. 2020].

[19] Vitasse, Y. and Rebetez, M. (2018) "Unprecedented risk of spring frost damage in Switzerland and Germany in 2017", *Climatic Change*, Vol. 149, pp. 233-246. E-ISSN 1573-1480, ISSN 0165-0009. DOI 10.1007/s10584-018-2234-y.

[20] Zhang, J. (2010) "Towards personal high-performance geospatial computing (hpc-g): perspectives and a case study", In *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems (HPDGIS)*, pp. 3-10. DOI 10.1145/1869692.1869694.