Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

SHAPE Project: Global Surface Intelligence

Optimisation of national Vegetation Mapping Workflows

N. Banglawala[a], K. Stratford[a], and M. Howie[b,*].

[a]EPCC, The University of Edinburgh, UK
[b]Global Surface Intelligence Ltd., UK

Abstract

Global Surface Intelligence (GSI) uses earth observation to assess natural resources such as forestry, to monitor agricultural assets, and to provide business intelligence on land use and sustainability across the globe. Its operation, involving large volumes of satellite and other data, depends on efficient and effective software tools for machine learning and artificial intelligence workflows. The purpose of the work described here was twofold: first, to improve software effectiveness and software sustainability by making GSI's standard workflows more robust and easier to use by non-experts and second, to improve parallel efficiency in a small number of specific data analysis workflows.

1. Introduction

The increasing wealth of data available from earth observation -- be it from satellites, drones, aircraft, or other remote sensing platforms -- has opened the way to new opportunities to provide value to business, governmental bodies, and wider society. One company taking advantage of this opening is Edinburgh-based Global Surface Intelligence (GSI) [1] formed in 2012. To do this, GSI integrates data from disparate sources and use machine learning and artificial intelligence techniques to extract the information they and their customers require. This in turn necessitates a combination of software to implement the relevant data workflows, and high performance computing to provide the processing capacity to obtain timely results at the scale required.

GSI has significant activities in North America, but also has increasing interests worldwide. GSI's value is built on the measurement and monitoring of natural resources – forest inventories, crops, land cover classification maps. GSI is part of ForestMind [2], a project co-funded by the European Space Agency (ESA) and major UK-based food retailer Sainsburys, the aim of which is to investigate the provenance of international food supply chains. GSI also works with the UK Space Agency and another Scottish company, Omanos Analytics [3], to analyse land use changes local to development of geothermal plants in Kenya [4]. As an example of a typical product, GSI generates measurements of forest species and coverage (including height) using a combination of optical and synthetic aperture radar satellite data, LiDAR and ground survey data (Figure 2). This allows the extraction of phenological data (seasonal variations) using a patented time series analysis as the basis for extrapolation of local ground truth.

The increase in data volumes available from new instruments with higher spatial and temporal resolution mean that high performance computing (HPC) is essential to GSI's activities. In the past, GSI has used traditional HPC platforms including those at The University of Edinburgh, but GSI also needs to be able to operate in a cloud environment, such as Amazon Web Services. The need of HPC usage increases emphasis on issues such as parallelisation, portability and containerisation of standard workflows. While individual workflows are often

* Corresponding author email address: mark@surfaceintelligence.com

developed by domain specialists, GSI's future emphasis will be in more toward ease-of-use by computational non-specialists. This will require the integration of different tools to support scalability and automation of data production, reflecting a business need to complete product delivery at increased scale and speed.

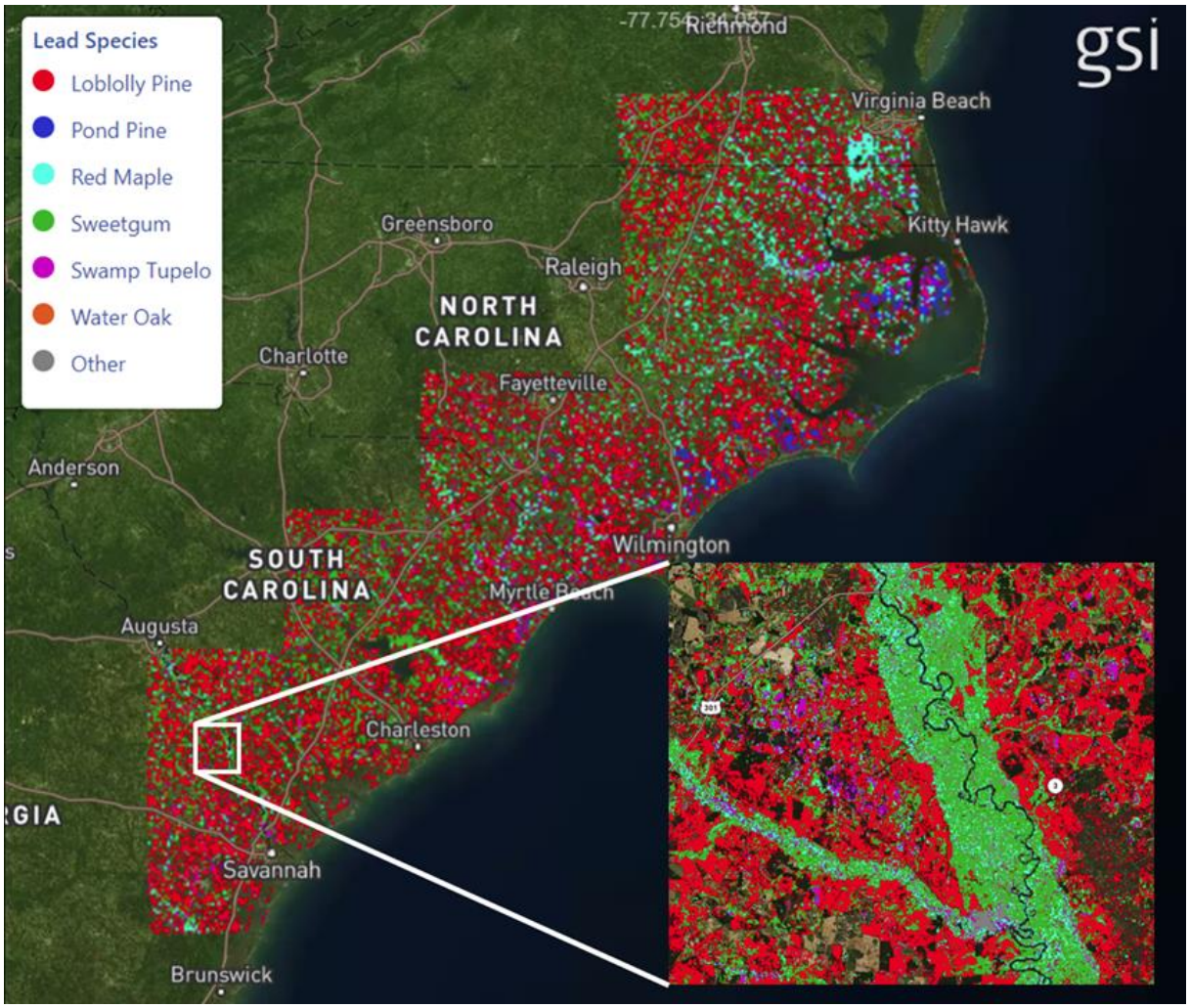

Figure 1: An example of a GSI product: a map of lead species in the eastern seaboard of the United States of America. Data is presented at 10m resolution, covering 17,000,000 hectares. Lead species is a derivative of per-species presence by basal area, generated through regression modelling. Other data layers include tree height and canopy closure.

This white paper describes a collaborative SHAPE project between GSI and EPCC at The University of Edinburgh [5] which investigated the sustainability of the software base which underpins GSI's future work, and looks at some of the parallel aspects of the computational workflows. In the following section we describe the work undertaken to improve sustainability, while in Section 3, portability and parallelisation considerations are described by looking at the case of implementing a workflow on a specific platform (NextGenIO [6]). Some comments on future developments conclude the report.

2. Software Sustainability

Background

GSI's work for a specific customer involves a combination of different software building blocks to orchestrate the relevant workflows or data pipelines needed to produce results. These building blocks may typically include standard toolboxes provided by remote sensing platforms for data processing, open-source packages for data manipulation, machine learning and so on, and in-house applications providing unique business insight. A set of software assets for a given project may share common elements with other projects for different clients, or it may

have specific customisations. GSI typically gives to each set of assets or project a codename: this report deals with the “moonshine” workflow.

Moonshine produces geospatial datasets of interest by applying machine learning to data from a variety of sources. These sources include the European Space Agency’s (ESA’s) Sentinel-2 platform [7] for which data discovery, download, and quality control is performed by the ESA’s Sen2Cor [8] processor. This provides multi-spectral data at 60m, 20m, or 10m resolutions in tiles covering different geographical areas, which is used as input for bespoke analysis (C or python code). This is combined with Shuttle Radar Topography Mission [9] data and/or US Geological Survey Lidar data for land elevation, and US Forestry Service [10] tree species data, to form the required input pipelines for machine learning and artificial intelligence stages. Orchestration is generally provided as python scripts, and various degrees of user intervention is either required, or desirable to refine the analysis.

The area of interest for each project can, potentially, span many tiles of satellite data (each around 100 km on a side), which provides one source of independent parallel computation. Orchestration here is done typically via shell scripts which may interact with the local job scheduler (e.g., PBS or SLURM). Other stages in the workflow are serial and therefore require synchronisation. Overall, it is important that the time-limiting steps are parallel to reduce time to solution.

Analysis and remediation

The type of workflow described above has been developed over time and stitched together on an expedient basis. This somewhat disjoint nature of the orchestration of different workflows had been identified by GSI as an area where possible improvements in sustainability could be made with a view to promote ease-of-use by non-developers.

To identify concrete areas for improvement we analysed a typical moonshine workflow covering a subset of its data pipelines by inspection, and by execution, and considered various characteristics of the software. A number of areas for improvement emerged where action could be taken.

A. Workflow configuration and management

1. The details of configuration required to specify the workflow (area of interest, time of interest, parameters to control various stages of the analysis and so on) should be unified in a single location rather than in individual scripts.

2. Likewise, local environment details (files locations, computational parameters, scheduling details and so on) should be managed separately to provide portability and platform independence.

3. A unified and general command line interface should be created to improve ease-of-use in the composition of workflows.

B. Workflow runtime

1. A single consistent logging facility should be used to capture information about the progress of a workflow and to provide improvements in repeatability and allow for auditing.

2. Robustness should be improved by augmenting consistency checks on user input, and providing uniform exception handling.

C. Workflow testing and code quality

1. Test coverage should be improved.

2. Repetition of code and common patterns between different workflows should be eliminated and replaced by unified facilities.

As a result of this analysis, EPCC developed a lightweight framework to capture these points. A schematic of the framework is given in Figure 1.

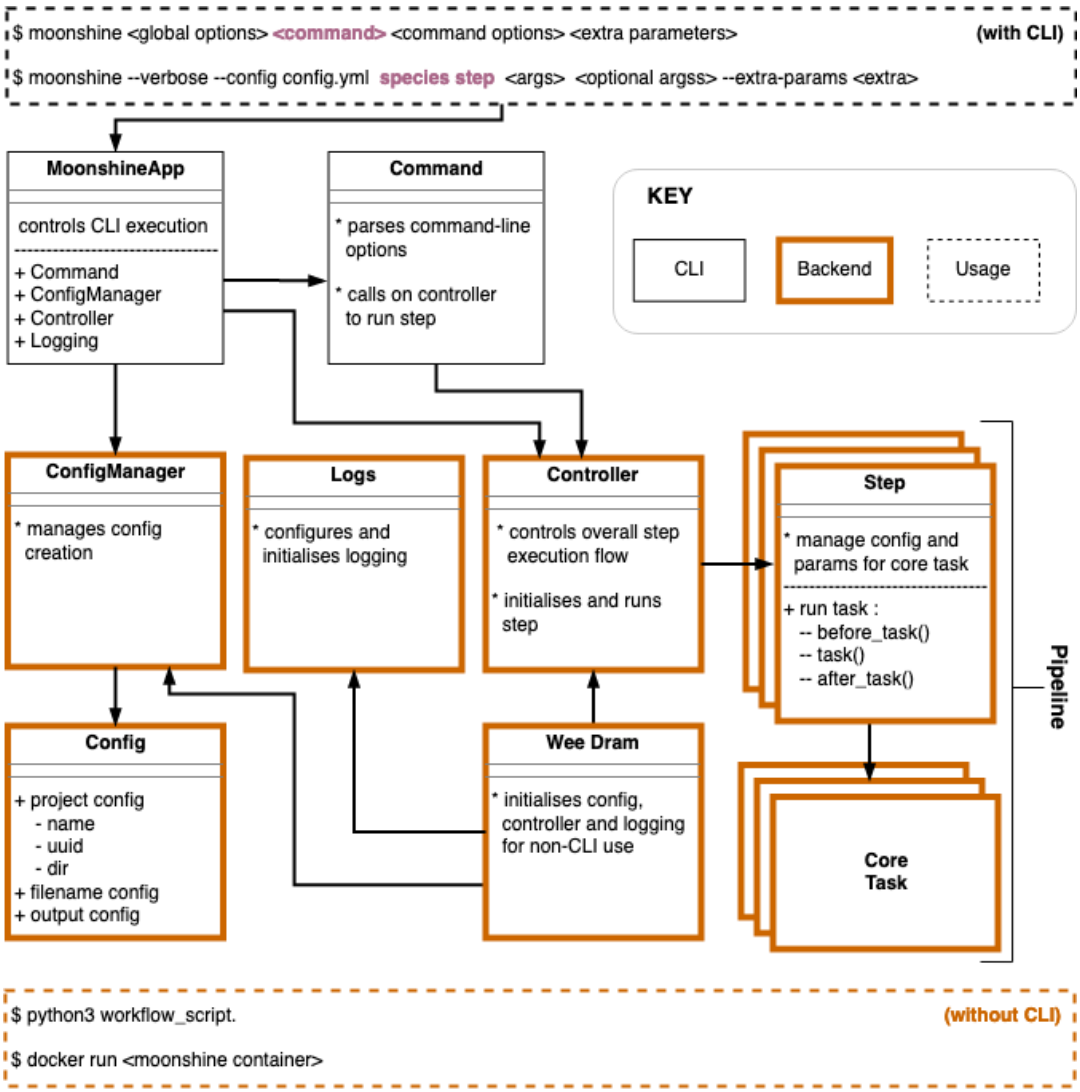

Figure 2: Moonshine Framework. The diagram shows the key classes that make up Moonshine's frontend, its command-line interface (CLI; thin black border) and Moonshine's backend (thick orange border). Example usage commands are shown (dashed line border) for the CLI (top) and for the non-interactive backend (bottom).

As shown in Figure 1, the Framework separates Moonshine's frontend, its command-line interface (CLI), from Moonshine's backend, its non-interactive functionality. The Framework is written exclusively in Python, and the CLI is based on CLIFF (Command-Line Interface Formulation [11]), a flexible command-line framework based on Python's standard Argparse module. A Moonshine command typically runs one processing step within a data pipeline. Moonshine's data pipelines share the following common processing steps:

- *Discover* datasets in a remote data resource;
- *Select* the desired datasets;
- *Pull* the datasets from the remote data resource;
- *Process* the pulled datasets;

These steps have been abstracted into decomposable sub-commands that can be used to run any Moonshine pipeline. For example, species data can be discovered in a remote data resource by issuing the following command:

```
$ moonshine species discover <args> <optional args>
```

The pipeline in the above example, “species”, can be substituted with another pipeline. This decomposable approach to commands makes it easy to add new commands to a given pipeline as new steps are developed. The Command class is essentially a CLI wrapper for executing a step. Output from a command or step can be logged; the CLI accepts global options to set log options and output verbosity levels. The Framework provides a consistent approach to logging, which is managed by the LogConfigurator class.

Moonshine’s pipelines grow as GSI develops novel solutions to new challenges. A key requirement for the Framework was to separate a pipeline’s core logic from the CLI: core tasks should not be “aware” of any external, executing framework. This separation also extends to configuration logic: core functionality should not need to handle any compute-platform specific configuration. The Step class is essentially a configuration layer, taking care of any configuration and I/O requirements for a given core task. For example, the Step object can run checks on input parameters, highlighting any issues early on a pipeline’s execution. Core tasks are executed by calling their corresponding Step object. When a Step object is run, the Framework returns a “return code” to indicate execution success or otherwise. This helps to improve the overall robustness of pipeline processes: if any non-fatal issues arise, for example one input file out of a hundred cannot be processed, the return code will flag this, and corresponding information will be logged, instead of fatally interrupting execution.

The CLI is managed by the MoonshineApp class, the CLI’s entry point, and the CommandManager class. The ConfigManager class parses configuration input and creates a Config object. The Config object, together with any additional input arguments and options, is eventually passed to the Controller. This class controls the overall final execution of a step. The Controller can be reached either by a Command from the CLI or via Moonshine’s backend, using the WeeDram class. The WeeDram class simply instantiates the necessary objects, such as the ConfigManageer, LogConfigurator and Controller, for executing a Step non-interactively. The Framework also comes with a set of regression tests that promote robustness and consistency for future development work.

The Moonshine Framework is a self-contained Python package that can easily be installed and run on HPC resources, either directly or within a containerized environment. The Framework provides an intuitive, easy-to-use CLI with default input parameters pre-set. The Framework enables a non-technical user to quickly get Moonshine pipelines up and running without needing to understand BASH scripts or specify advanced configuration or processing options, whilst also supporting advanced users and developers that require greater control over configuration and parameter settings. These different levels of usage are documented in the Framework’s User Guide and Developer Guide respectively. The Framework enables GSI’s technical team to rapidly develop new functionality easily and without compromising sustainability or performance. The Framework also enables GSI’s production team to easily run complex workflows in a repeatable and traceable manner without compromising on robustness and without needing advanced technical knowledge.

3. NEXTGenIO: test case for workflow portability

NEXTGenIO [6] is a HPC machine developed specifically to address the relative bottleneck presented by the imbalance between the speed of data access when held in main memory and the speed when files reside on disk. This makes it an interesting test case for data science workflows of the type performed by GSI. The solution to the imbalance involves the use of non-volatile RAM (NVRAM) to augment main memory (DRAM). NEXTGenIO's nodes feature 2 24 core Xeon CPUs each with 6x16 GB of standard DRAM, but also with 6x256 GB NVRAM. The NVRAM can be used in two different modes: a one level picture where the NVRAM is used as a file system with low latency and high bandwidth; and a two-level mode where NVRAM is essentially an extension to DRAM (with only slightly higher latency and lower bandwidth). NEXTGenIO uses a standard SLURM scheduler (with a number of local specific additions related to the use of NVRAM).

To allow GSI to run their workflows on NextGenIO, relevant improvements to portability were introduced. GSI then moved their production activities to NEXTGenIO from a more traditional Unix-based system. The larger memory available allowed GSI to expand the spatial extent of their analyses without refactoring the underlying processes. Processing times were improved by up to 5 times, and a significant reduction in the number of failed processes was observed reflecting improved robustness.

Conclusion

In this project, we developed a lightweight workflow management framework for GSI’s Moonshine codebase. The framework addressed areas of improvement, such as robustness, sustainability and ease-of-use of the Moonshine codebase. The work undertaken forms the basis of a more flexible integrated pipeline for data scientists at GSI to

perform their work without specific knowledge of the underlying computational resource. The framework approximately makes up 10% of a larger GSI project that aims to reduce production costs by 90%.

The framework created by this project has applications across the whole of GSI's toolset, and has the potential to reduce production and maintenance costs, whilst supporting sustainable scale-out and migration of production – across all of GSI's product and service offerings.

The framework also enables GSI's technical team to add new functionality without compromising on robustness, sustainability or ease-of-use. Combined with other strands of development at GSI, this will enable GSI to enable use a range of HPC and cloud services, e.g., Amazon Web Services [11]. We also improved the portability of GSI workflows between different HPC systems, making GSI more agile in its response to business-critical needs.

References

[1] https://www.surfaceintelligence.com/ (accessed: March, 2021).
[2] https://business.esa.int/projects/forestmind
[3] https://www.omanosanalytics.org/
[4] https://www.gov.uk/government/news/new-uk-space-projects-to-boost-global-sustainable-development-receive-34-million-cash-boost
[5] http://www.epcc.ed.ac.uk/ (accessed: March, 2021).
[6] http://www.nextgenio.eu/ (accessed: March, 2021).
[7] https://sentinel.esa.int/web/sentinel/missions/ (accessed: March 2021).
[8] https://step.esa.int/main/snap-supported-plugins/sen2cor/ (accessed: March, 2021).
[9] https://www2.jpl.nasa.gov/srtm/ (accessed: March, 2021).
[10] https://www.fia.fs.fed.us/ (accessed: March, 2021).
[11] https://docs.openstack.org/cliff/latest/
[12] https://aws.amazon.com/

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's Horizon 2020 Research and Innovation programme (2014-2020) under grant agreement 823767. We are grateful to Adrian Jackson of EPCC for help with NEXTGenIO.