Data Processing and Analytics for Data-Centric Sciences

Leonardo Candela(✉), Gianpaolo Coro, Lucio Lelii, Giancarlo Panichi, and Pasquale Pagano

National Research Council of Italy, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Via G. Moruzzi, 1, 56124 Pisa, Italy
{leonardo.candela,gianpaolo.coro}@isti.cnr.it

Abstract. The development of data processing and analytics tools is heavily driven by applications, which results in a great variety of software solutions, which often address specific needs. It is difficult to imagine a single solution that is universally suitable for all (or even most) application scenarios and contexts. This chapter describes the data analytics framework that has been designed and developed in the ENVRIplus project to be *(a)* suitable for serving the needs of researchers in several domains including environmental sciences, *(b)* open and extensible both with respect to the algorithms and methods it enables and the computing platforms it relies on to execute those algorithms and methods, and *(c) open-science-friendly*, i.e. it is capable of incorporating every algorithm and method integrated into the data processing framework as well as any computation resulting from the exploitation of integrated algorithms into a "research object" catering for citation, reproducibility, repeatability and provenance.

Keywords: Data analytics · Open science · Virtual Research Environment

1 Introduction

Data processing and analytics are playing a key role in modern science [8]. The paradigms of data-intensive science [20] and open science [5] as well as the opportunities offered by cloud computing and the *as-a-Service* delivery model [1, 21] continuously drive the development of ever new and diverse solutions to data analytics needs. In fact, a plethora of solutions has been designed and developed ranging from programming models to analytics frameworks and systems, notebooks, workflow management systems and science gateways [6, 9, 22, 25]. Such heterogeneity provides a rich possibility for the scientific community to select suitable solutions, case by case; however, it also results in high costs to support the sharing of analytics methods and algorithms across domains.

In order to provide scientists with an analytics platform aiming at offering both (a) the freedom to develop analytics methods by using a rich array of programming approaches (e.g. R scripts, Python programs and Unix compiled code) and (b) a simple use and re-use of analytics methods developed by others according to open science practices, advanced

Z. Zhao and M. Hellström (Eds.): Towards Interoperable Research Infrastructures for Environmental and Earth Sciences, LNCS 12003, pp. 176–191, 2020.
https://doi.org/10.1007/978-3-030-52829-4_10

infrastructures like D4Science have been equipped with a set of services realising a solution for data analytics promoting the above principles [3, 4]. This chapter presents the data analytics solutions developed as part of D4Science infrastructure in the context of ENVRIplus [37], and discusses the benefits resulting from its uptake and exploitation in several use cases.

The rest of the chapter is organised as follows. Section 2 overviews existing solutions for data analytics. Section 3 discusses the data analytics solution developed in the project. Section 4 examines the proposed solution and discusses some use cases. Finally, Sect. 5 concludes the chapter by highlighting future works.

2 State of the Art

Khalifa et al. [22] surveyed existing solutions for data analytics and observed that (i) "existing solutions cover bits-and-pieces of the analytics process" and (ii) when devising an analytics solution there are six pillars representing issues to deal with. The six pillars identified by Khalifa et al. include (i) *Storage*, i.e., how data to be analysed are going to be handled; (ii) *Processing*, i.e., how the pure processing task is going to happen; (iii) *Orchestration*, i.e., how computing resources are going to be managed to reduce processing time and cost; (iv) Assistance, i.e., how scientists and practitioners are provided with facilities helping them to perform their task; (v) *User Interface*, i.e., how scientists and practitioners are provided with the data analytics system to run their analytics, monitor the execution and get back the results; (vi) *Deployment Method*, i.e., how the analytics system is offered to the end-users.

A lot of technologies and approaches have been developed to support data processing and analytics tasks including (i) High-performance computing (HPC) solutions, i.e., aggregated computing resources to realise a "high-performance computer" (including processors, memory, disk and operating system); (ii) Distributed Computing Infrastructures, i.e., distributed systems characterised by heterogeneous networked computers that offer data processing facilities. This includes High-throughput computing (HTC) and cloud computing; (iii) Scientific workflow management systems (WMS), i.e., systems enacting the definition and execution of scientific workflows consisting of: a list of tasks and operations, the dependencies between the interconnected tasks, control-flow structures and the data resources to be processed; (iv) Data analytics frameworks and platforms, i.e., platforms and workbenches enabling scientists to execute analytic tasks. Such platforms tend to provide their users with implementations of algorithms and (statistical) methods for the analytics tasks. These classes of solutions and approaches are not isolated, rather they are expected to rely on each other to provide end-users with easy to use, efficient and effective data processing facilities, e.g. scientific WMS rely on distributed computing infrastructures to actually execute their constituent tasks.

Liew et al. [25] have recently analysed selected scientific WMSs that are widely used by the scientific community, namely: Airavata [30], Kepler [27], KNIME [7] Meandre [26], Pegasus [18], Taverna [31], and Swift [34]. Such systems have been analysed with respect to a framework aiming at capturing the major facets characterising WMSs: (a) processing elements, i.e., the building blocks of workflows envisaged to be either web services or executable programs; (b) coordination method, i.e., the mechanism

controlling the execution of the workflow elements envisaged to be either orchestration or choreography; (c) workflow representation, i.e., the specification of a workflow that can meet the two goals of human representation and/or computer communication; (d) data processing model, i.e., the mechanism through which the processing elements process the data that can be bulk data or stream data; (e) optimisation stage, i.e., when optimization of the workflow (if any) is expected to take place that can either be build-time or runtime.

A series of *platforms and frameworks* have been developed to simplify the execution of (scientific) distributed computations. This need is not new; it is actually rooted in high-throughput computing which is a well-consolidated approach to provide large amounts of computational resources over long periods of time. The advent of Big Data and Google MapReduce in the first half of 2000 brings new interests and solutions. Besides taking care of the smart execution of user-defined and steered processes, platforms and environments start offering ready to use implementations of algorithms and processes that benefit from a distributed computing infrastructure.

There exist many data analytics frameworks and platforms, including:

- Apache Mahout[1] is a platform offering a set of machine-learning algorithms (including collaborative filtering, classification, clustering) designed to be scalable and robust. Some of these algorithms rely on Apache Hadoop, others are relying on Apache Spark.
- Apache Hadoop[2] is a basic platform for distributed processing of large datasets across clusters of computers by using a MapReduce strategy [24]. In reality, this is probably the most famous open-source implementation of MapReduce, a simplified data processing approach to execute data computing on a computer cluster. Worth to highlight that one of the major issues with MapReduce – resulting from the "flexibility" key feature, i.e., "users" are called to implement the code of map and reduce functions – is the amount of programming effort. In fact, other frameworks and platforms are building on it to provide users with data analytics facilities (e.g. Apache Mahout).
- Apache Spark [33] is an open-source, general-purpose cluster- computing engine which is very fast and reliable. It provides high-level APIs in Java, Scala, Python and R, and an optimised engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.
- iPython/Jupyter [29] is a notebook-oriented interactive computing platform which enacts to create and share "notebooks", i.e., documents combining code, rich text, equations and graphs. Notebooks support a large array of programming languages (including R) and communicate with computational kernels by using a JSON-based computing protocol. Similar solutions include knitr [32] which works with the R programming language and Dexy[3], a notebook-like program that focuses on helping users to generate papers and presentations that incorporate prose, code, figures and other media.

[1] http://mahout.apache.org/.
[2] https://hadoop.apache.org/.
[3] http://www.dexy.it.

3 DataMiner: A Distributed Data Analysis Platform

DataMiner (DM) [3, 11, 14] aims to provide a cloud-based data analysis platform. It is designed to support the following key principles:

- *Extensibility*: the platform is "open" with respect to (i) the analytics techniques it offers and supports and (ii) the computing infrastructures and solutions it relies on to enact the processing tasks. It is based on a plug-in architecture to support adding new algorithms/methods, new computing platforms;
- *Distributed processing*: the platform is conceived to execute processing tasks by relying on "local engines"/"workers" that can be deployed in multiple instances and execute parallel tasks in a seamless way to the user. The platform is able to rely on computing resources offered by both well-known e-Infrastructures (e.g. EGI) as well as resources made available by the Research Infrastructure to deploy instances of the "local engines"/"workers". This is key to make it possible to "move" the computation close to the data;
- *Multiple interfaces*: the platform offers its hosted algorithms as-a-service and via a Web graphical user interface. This allows using the algorithms from software capable to execute processing tasks from well-known applications (e.g. R and KNIME);
- *Cater for scientific workflows*: the platform is exploitable by existing WorkFlow Management System (WFMS) [36] (e.g. a node of a workflow can be the execution of a task/method offered by the platform) and supports the execution of a workflow specification (e.g. by relying on one or more instances of WFMSs);
- *Easy to use*: the platform was designed in order to accommodate usability requirements for different types of users ranging from undergraduate students to expert scientists;
- *Open science-friendly*: the platform aims at supporting the open science paradigm in terms of repeatability and reproducibility of the experiments and the re-usability of the produced results. This goal is achieved via computational provenance tracking and the production of "research objects" that make it possible for users to repeat the experiment done by other users while respecting access policies to data and processes.

3.1 Development Context

The development of DataMiner is in the context of the D4Science infrastructure [3, 4] which is designed and developed to provide researchers and practitioners with working environments where resources of interests (datasets, computing, and services) for each designated community are easily made available by coherent and aggregated views and where open science practices are transparently promoted. This infrastructure is built and operated by relying on gCube technology [4], a software system specifically conceived to enable the construction and development of *Virtual Research Environments* (*VREs*) [10], i.e. web-based working environments dynamically build thus to be tailored to support the needs of their designated communities, each working on a research question. Every VRE offers domain-specific facilities, i.e. datasets and services suitable for the specific research question, as well as a rich array of basic services supporting collaboration and cooperation among its users, namely: (i) a *shared workspace* to store and organise any version of a research artefact; (ii) a *social networking area* to have discussions on any

topic (including working version and released artefacts) and be informed on happenings; (iii) a *data analytics platform* to execute processing tasks either natively provided by VRE users or borrowed from other VREs to be applied to VRE users' cases and datasets, and (iv) a *catalogue-based publishing platform* to make the existence of a certain artefact public and disseminated. These facilities are at the fingerprint of VRE users.

The data analytics part mainly comprises two major components further discussed below, i.e. *DataMiner* realising the analytics engine supporting both providers and consumers of analytics methods to make use of a distributed computing infrastructure; and *Statistical Algorithms Importer (SAI)* realising the facilitator helping analytics methods owner to onboard their methods into the DataMiner engine. These components are nicely integrated with the services discussed above, thus the overall analytics platform result from the intertwining of them with the rest.

3.2 Architecture

DataMiner is based on the 52° North WPS implementation[4] but extends this implementation in order to meet Open Science oriented requirements. DM is a Java-based Web service running on an Apache Tomcat instance, enhanced with libraries to make it work in the D4Science e-Infrastructure. The DM architecture is made up of two sets of machines (clusters) that operate in a Virtual Research Environment: the Master and the Worker clusters, as shown in Fig. 1. The Master and the Worker clusters are dynamically provisioned by D4Science through an orchestration engine based on the OpenStack platform for Cloud computing. A load balancer distributes the computational requests uniformly to these machines. When a WPS computational request comes to the Master cluster balancer, this distributes the request to one of the services in the Master DM cluster. The DMs host processes provided by several developers. Each machine is endowed with a DM service that communicates with the D4Science Information System to notify its presence and capabilities. The load balancer is the main access point to interact with DM services. The machines of the Worker cluster have the same computational power as the other machines but can run only one (parallelised) process at a time and principally serve Map-Reduce computations.

Two types of algorithms are hosted by DataMiner: "Local" and "Cloud" algorithms. Local algorithms are directly executed on the Master machines and possibly use parallel processing on several cores and a large amount of memory. Instead, Cloud algorithms use distributed computing with a Map-Reduce approach and rely on the Worker cluster (Cloud nodes). With respect to the standard 52° North implementation, DM adds a number of features. First, it returns a different list of processes according to the VRE in which the service is invoked. When an algorithm is installed on a DM, it is also indexed in the D4Science Information System as an infrastructural resource. Thus, an e-Infrastructure manager can make it available to a number of VREs according to the algorithm's publication policies. When invoked in a VRE, DM returns only the subset of processes that have been assigned to that VRE.

[4] 52° North Web Processing Service software website https://52north.org/software/software-projects/wps/.

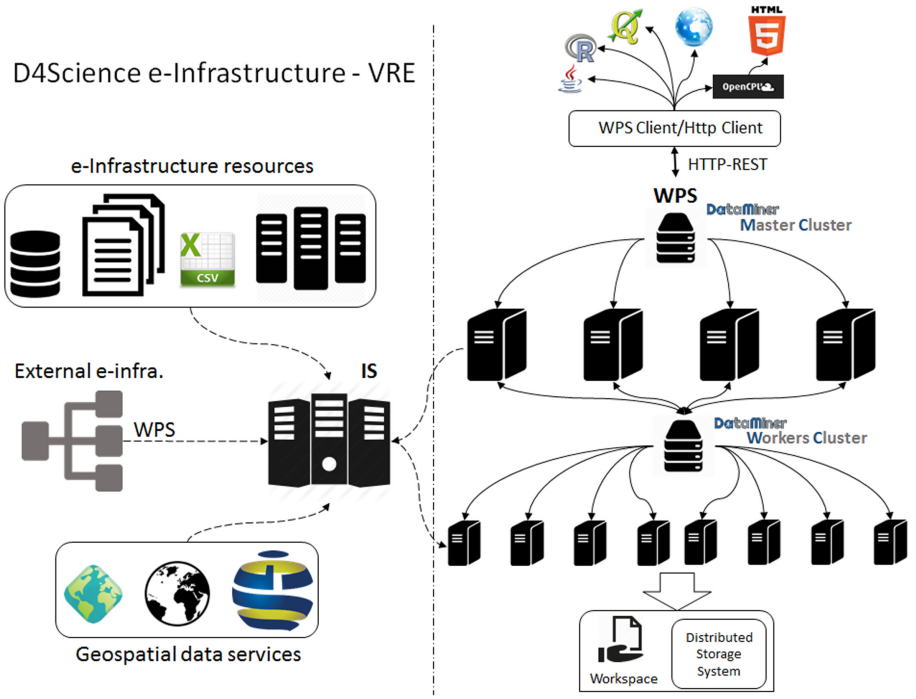

Fig. 1. The architecture of the D4Science DataMiner system.

The DMs are stateful services, i.e. when a computation starts, they communicate the status of the execution to asynchronous clients. The WPS standard embeds a direct URL to the DM machine that is carrying out the computation to check for the status. For each executed computation, DM releases an "equivalent HTTP-GET" request to repeat the experiment via a Web browser.

The DataMiner services use the security services of D4Science and require a user token to be specified via HTTPS-access authentication for each operation. The token identifies both a user and a Virtual Research Environment and is used to size the resources available to the user.

The DataMiner computations can use inputs loaded onto the D4Science Workspace, which may come also from Workspace folders shared among several users. This enables collaborative experimentation already at the input selection phase. Inputs can also come from external repositories because a file can be provided either as an HTTP link or embedded in a WPS execution request. The computational outputs are written onto the D4Science Workspace at the end of the computation. A Workspace folder is created that contains the input, the output, the parameters of the computation, and a provenance document in the Prov-O format summarizing this information. This folder can be shared with other people and can be used to execute the process again.

3.3 System Implementation

Overall, the following system components have been developed in DataMiner:

- The GUI: a web-based user interface enacting users to select an existing process, execute it, monitor the execution and access to the results (cf. Sect. 3.4);
- The DataMiner Master computational cluster: this web service is in charge to accept requests for executing processes and executing them, either locally or by relying on the DataMiner Worker(s) depending from the specific process. The service is conceived to work in a cluster of replica services and is offered by a standard web-based protocol, i.e. OGC WPS;
- The DataMiner Worker: this web service is in charge to execute the processes it is assigned to. The service is conceived to work in a cluster of replica services and is offered by a standard web-based protocol, i.e. OGC WPS;
- The DataMiner Processes: this a repository of processes the platform is capable to execute. This repository if equipped with a set of off-the-shelf processes and it can be further populated with new processes either (a) developed from scratch in compliance with a specific API or (b) resulting from annotating existing processes (cf. Sect. 3.5).

DataMiner [11, 14] has been included in the D4Science environment as a cloud computing platform, which currently makes ~400 processes available as-a-service.

Every analytics process is automatically exposed by the Web Processing Service (WPS) standard protocol and API of the Open Geospatial Consortium[5]. Thanks to this, every single process can be exploited by a number of clients embedded in third-party software that can interact with the DataMiner hosted processes through WPS.

DataMiner allows the hosted processes to be parallelised for execution both on multiple virtual cores and on multiple machines organised as a cluster. A typical DataMiner cluster to support big data processing is made up of 15 machines with Ubuntu 16.04.4 LTS x86 64 operating system, 16 virtual cores, 32 GB of RAM and 100 GB of disk space. A further cluster with a similar configuration is available to manage Map-Reduce computations. The DataMiner machines are hosted principally by the National Research Council of Italy[6], the Italian Network of the University and Research (GARR)[7], and the European Grid Infrastructure (EGI)[8]. A load balancer distributes computational requests uniformly to the machines of the computational cluster. Each machine hosts a processing request queue that allows a maximum of 4 concurrent executions running on one machine. With this combination of parallel and distributed processing, DataMiner allows processing big data while enabling provenance tracking and results sharing.

At the end of computation, the meta-information about the input and output data, and the parameters used (i.e. the computational provenance) are automatically saved on the D4Science Workspace and are described using the Prov-O XML ontological standard [23].

---

[5] Open Geospatial Consortium https://www.opengeospatial.org/standards/wps.

[6] National Research Council of Italy website www.cnr.it.

[7] GARR Consortium website www.garr.it.

[8] EGI Foundation website www.egi.eu.

A Web interface is available for each process, which is automatically generated based on the WPS interpretation. Through this interface, users can select the data to process from the Workspace and conduct experiments based on shared folders that allow automatic sharing of results and provenance with other users.

DataMiner also offers tools to integrate algorithms written in a multitude of programming languages [17].

3.4 Data Provenance During Data Processing

The research objects produced by the DataMiner computations are a set of files organised in folders and containing every input, output, an executable reference to the executed method as well as rich metadata including a PROV-O provenance record. The research objects are saved together with their metadata on the D4Science Workspace. Objects in the Workspace can be shared with co-workers and can be published by a catalogue with a license governing their usage.

DataMiner can operate within one or more Virtual Research Environments, i.e. it interoperates with additional VRE-specific services that manage multi-tenancy, communities, social networking communications, and collaboration among the VRE members.

From the end-user perspective, DataMiner offers a working environment oriented to collaborative experimentation where users:

- can easily execute and monitor data analytics tasks by relying on a rich and open set of available methods, either via WPS service endpoints or via Web GUI;
- can easily share & publish their analytics methods (e.g. implemented in R, Java and Python) within a Virtual Research Environment and make them usable by other processes supporting WPS;
- are provided with a "research object" describing every computational job executed by the workbench, which enables repeatability, reproducibility, re-use, citation, and provenance tracking for the experiments.

Overall DataMiner gives access to two types of resources:

- A distributed, open, and heterogeneous computing infrastructure for the execution of data analysis and mining tasks;
- A large pool of methods integrated with the platform, each made available as-a-service under the WPS standard according to VRE-specific access policies, i.e. public, private, shared with selected users, unavailable.

3.5 The Web-Based User Interface

DataMiner offers a web-based GUI to its users, as shown in Fig. 2. On the left panel (Fig. 2 a), the GUI presents the list of capabilities available in the specific "application context", which are semantically categorised (the category is indicated by the process provider). For each capability, the interface calls the WPS DescribeProcess operation to get the descriptions of the inputs and outputs. When a user selects a process in the right

panel, the GUI on-the-fly generates a form with different fields corresponding to the inputs. Input data can be selected from the Workspace (the button associated with the input opens the Workspace selection interface). The “Start Computation” button sends the request to the DM Master cluster, which is managed as explained in the previous section. The usage and the complexity of the cloud computations are completely hidden from the user, but the type of computation is reported as metadata in the provenance file.

A view of the results produced by the computations is given in the “Check the Computations” area (Fig. 2 b), where a summary sheet of the provenance of the experiment can be obtained (“Show” button, Fig. 2 c). From the same panel, the computation can be also resubmitted. In this case, the Web interface reads the XML file containing the PROV-O information associated with computation and rebuilds a computation request with the same parameters. The computation folders may also include computations executed and shared by other users. Finally, the “Access to the Data Space” button allows obtaining a list of the overall input and output datasets involved in the executed computations (Fig. 2 d), with provenance information attached that refers to the computation.

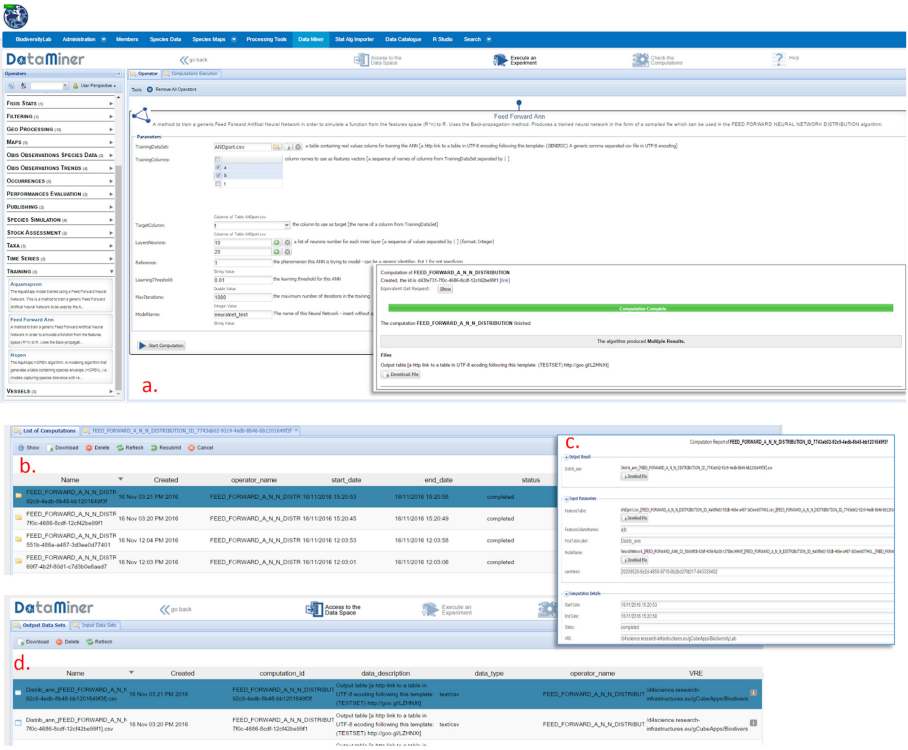

Fig. 2. The web interface of DataMiner in a D4Science VRE.

3.6 The Algorithms Importer

DataMiner allows importing prototype and production scripts written in a variety of languages (R, Java, Python, .Net, Fortran, Linux bash scripts, etc.). The tool was initially conceived to support scientists making prototype scripts, who needed to share results and provide their models and methods for use by other scientists. To this aim, DataMiner allows scientists to publish as-a-service scripts usually running on private desktop machines.

The Statistical Algorithms Importer (SAI) is the DataMiner interface that allows scientists to easily and quickly integrate their software with DataMiner, which in turn publishes this software-as-a-service under the WPS standard while managing multi-tenancy and concurrency. Additionally, it allows scientists to update their software without following long software re- deploying procedures each time.

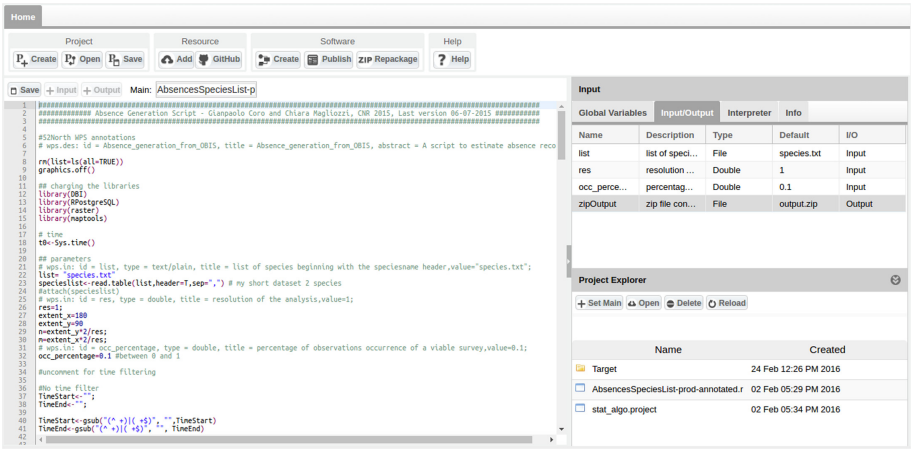

Fig. 3. The interface of the DataMiner Algorithms Importer tool.

The SAI interface (shown in Fig. 3 for R Script importing) is managed through a control panel in the top bar. The "Project" button allows creating, opening and saving a working session. A user uploads a set of files and data on the workspace area (lower-right panel). Upload can be done by dragging and dropping local desktop files. As a next step, the user indicates the "main script", i.e. the script/program that will be executed on DataMiner and that will use the other scripts and files. After selecting the main script, the left-side editor panel visualises it (when this is not compiled software) and allows modifying it. Afterwards, the user indicates the input and output of the script by highlighting variable definitions in the script and pressing the +Input (or +Output) button: behind the scenes the application parses the script strings and guesses the name, description, default value and type of the variable. This information is visualised in the top-right side Input/Output panel, where the user can modify the guessed information. Alternatively, for R scripts SAI can automatically compile the same information based on WPS4R annotations in the script. Other tabs in this interface area allow setting global variables and adding metadata to the process. In particular, the Interpreter tab allows

indicating the programming language interpreter version and the packages required by the script and the Info tab allows indicating the name of the algorithm and its description. In the Info tab, the user can also specify the VRE in which the algorithm should be available.

Once the metadata and the variables information have been compiled, the user can create a DataMiner as-a-service version of the software by pressing the "Create" button in the Software panel. The term "software", in this case indicates a Java program that implements an as-a-service version of the user-provided scripts. The Java software contains instructions to automatically download the scripts and the other required resources on the server that will execute it, configure the environment, execute the main script and return the result to the user. The computations are orchestrated by the DataMiner computing platform that ensures the program has one instance for each request and user. The servers will manage concurrent requests from several users and execute code in a closed sandbox folder, to avoid damage caused by malicious code. Based on the SAI Input/Output definitions written in the generated Java program, DataMiner automatically creates a Web GUI (cf. Sect. 3.4). By pressing the "Publish" button, the application notifies DataMiner that a new process should be deployed. DataMiner will not own the source code, which is downloaded on-the-fly by the computing machines and deleted after the execution. This approach meets the policy requirements of those users who do not want to share their code. The "Repackage" button re-creates the software so that the computational platform will be using the new version of the script. The repackaging function allows a user to modify the script and to immediately have the new code running on the computing system. This approach separates the software updating and deployment phases, making the script producer completely independent on e-Infrastructure deployment and maintenance issues. However, deployment is necessary again whenever Input/Output or algorithm's metadata are changed [38, 39].

To summarise, the SAI Web application enables a user integrated software to run as-a-service features. SAI reduces integration time with respect to direct Java code writing. Additionally, it adds (i) multi-tenancy and concurrent access, (ii) scope and access management through Virtual Research Environments, (iii) output storage on a distributed, high-availability file system, (iv) graphical user interface, (v) WPS interface, (vi) data sharing and publication of results, (vii) provenance management, (viii) accounting facilities, and (ix) Open Science compliance.

4 Discussion

The heterogeneity characterising existing data analytics systems makes it evident that when discussing data processing "solutions" there are different angles, perspectives and goals to be considered. When analysing technologies from the scientist-perspective, the following trends should be considered:

- Technology should be "*ease of (re-)use*", i.e., it should not distract effort from the pure processing task. Scientists should be exposed to technologies that are flexible enough to enable them to quickly specify their processing algorithm/pipeline. It should not require them to invest effort in learning new programming languages or in deploying, configuring or running complex systems for their analytics tasks. Methods and

algorithms are expected to be reused as much as possible, thus data processing should enable them to be "published" and shared.
- "*as-a-Service*" rather than "*do-it-yourself*", i.e., scientists should be provided with an easy to use working environment where they can simply inject and execute their processing pipelines without spending effort in operating the enabling technology. This makes it possible to rely on economies of scale and keep the costs low.
- Solutions should be "hybrid", i.e., it is neither suitable nor possible to implement one single solution that can take care of any scientific data processing need. Certain tasks must be executed on specific infrastructures while other tasks are conceived to crunch data that cannot be moved on other machines from where they are stored.

These trends actually suggest that scientists are looking for things like "workbenches" or "virtual research environments" or "virtual laboratories" [10] providing them with easy to use tools for accessing and combining datasets processing workflows that behind the scene and almost transparently exploit a wealth of resources residing on multiple infrastructures and data providers (according to their policies). Such environments should not be pre-cooked or rigid, rather they should be flexible to enable scientists to enact their specific workflows. They should provide their users with appropriate and detailed information enacting to monitor the execution of such a workflow and be informed of any detail occurring during the execution. Finally, they should promote "open science" practices, e.g. they should record the entire execution chain leading to a given result, they should enact others to repeat/repurpose an existing process.

When analysing the overall solution to be developed and operated by RIs, the following aspects (going beyond the technology) are worth being considered:

- Provide support for research developers who produce and refine the code and workflows that underpin many established practices, scientific methods and services. Without their efforts in understanding issues, in explaining software behaviour, and improving quality, scientists would struggle to continue to handle existing methods and explore new opportunities. They need tools that inform them about the operational use of their products and technologies that protect their invested effort as platforms evolve. They are in danger of being undervalued, overwhelmed by the complexity and the pace of change, and of being attracted to the "big data" industry.
- Provide support for operations teams who need to keep the complex systems within and between RIs running efficiently as platforms change and communities' expectations rise while funders become more miserly. The tools and support they need are similar to those discussed in the previous bullet. They are not the same as the e-Infrastructure providers, they deploy and organise above those resources, but depend on them.
- Provide support for scientific innovators. They need to play with ideas, work on samples in their own favourite R&D environment, and then test their ideas at a moderate and growing scale. The provided facilities should allow them to move easily between developing ideas and proto-deployment, and eventually, when their ideas work out, to production deployment.
- The majority of researchers do not want to innovate, they just want to get on with their daily job. As much care as possible must be invested in protecting their working practices from change. However, if tools become available, e.g. driven from provenance

data, which help their work by removing chores, such as naming, saving, moving and archiving data, without them feeling they have lost responsibility for quality, then they will join in, and that eventually leads to fewer errors and better-curated data [28].
- There are some computational processes that require expert oversight while they are running, that can save substantial waste or steer to better results.

All in all, data processing is strongly characterised by the "one size does not fit all" philosophy. There is no, and there will arguably never be, a single solution that is powerful and flexible enough to satisfy the needs arising in diverse contexts and scenarios.

The tremendous velocity characterising technology evolution calls for implementing sustainable data processing solutions that are not going to require radical revision by specialists whenever the supporting technologies evolve. Whenever a new platform capable of achieving better performance compared to existing ones becomes available, users are enticed to move to the new platform. However, such a move does not come without pain and costs.

Data analytics tasks tend to be complex pipelines that can require combining multiple processing platforms and solutions. Exposing users to the interoperability challenges resulting from the need to integrate and combine such heterogeneous systems strongly reduces their productivity.

There is a need to develop data processing technologies that address the problem by abstracting from (and virtualising) the platform(s) that take care of executing the processing pipeline. Such technologies should go in tandem with optimisation technologies and should provide the data processing designer with fine-grained processing directives and facilitate detailed specification of processing algorithms [35].

The solution for data analytics we presented so far was designed and implemented by taking all of this into account thus resulting suitable for several application scenarios. Concrete use cases have been discussed in previous works, e.g. [2, 12, 13, 15, 16, 19, 35]. Coro et al. [11] discuss how the overall solution presented here has been exploited to implement a complex use case in computational biology reducing the time from months to a few hours.

ENVRIplus use cases developed by relying on this solution are discussed in Chapter 17.

5 Conclusion and Future Work

Data Processing is a wide concept embracing tasks ranging from (systematic) data collection, collation and validation to data analytics aiming at distilling and extracting new "knowledge" out of existing data by applying diverse methods and algorithms. When devising a solution suitable for the heterogeneity characterising science nowadays it is immediate to realise that it is almost impossible to envisage a solution that is powerful and flexible enough to satisfy the needs arising in diverse contexts and scenarios.

In this chapter, we presented a solution for data analytics that is open by design, i.e. conceived to (a) host and enact data analytics processes implemented by relying on several languages, and (b) transparently offer computing capacity from several and heterogeneous providers. Moreover, the envisaged solution has been intertwined with

other services thus to facilitate the implementation of open science practices. Such a solution proved to be effective in several application contexts.

Future work includes the need to enhance the facilities aiming at exploiting integrated processes into notebooks and WFMS. In fact, although WPS facilitates this activity some development is needed to invoke every process. Moreover, mechanisms aiming at transforming the platform into a proactive component that by considering the user task can suggest suitable processes to play with.

Acknowledgements. This work was supported by the European Union's Horizon 2020 research and innovation programme via the ENVRIplus project under grant agreement No. 654182.

References

1. Allen, B., et al.: Software as a service for data scientists. Commun. ACM **55**(2), 81–88 (2012)
2. Assane, M., et al.: Realising a science gateway for the agri-food: the aginfra plus experience. In: 11th International Workshop on Science Gateway (IWSG) (2019)
3. Assante, M., et al.: Enacting open science by D4science. Future Gener. Comput. Syst. **10**(1016), 555–563 (2019). http://www.sciencedirect.com/science/article/pii/S0167739X1831464X
4. Assante, M., et al.: The gcube system: delivering virtual research environments as-a-service. Future Gener. Comput. Syst. **95**, 445–453 (2019)
5. Bartling, S., Friesike, S. (eds.): Opening Science: The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-00026-8
6. Belcastro, L., Marozzo, F., Talia, D.: Programming models and systems for big data analysis. Int. J. Parallel Emergent Distrib. Syst. **34**(6), 632–652 (2019)
7. Berthold, M.R., et al.: Knime-the konstanz information miner: version 20 and beyond. AcM SIGKDD Explor. Newsl. **11**(1), 26–31 (2009)
8. Bordawekar, R., Blainey, B., Apte, C.: Analyzing analytics. ACM SIGMOD Record **42**(4), 17–28 (2014)
9. Calegari, P., Levrier, M., BalczyÅski, P.: Web portals for high-performance computing: a survey. ACM Trans. Web **13**(1), 1–5 (2019). http://doi.acm.org/10.1145/3197385
10. Candela, L., Castelli, D., Pagano, P.: Virtual research environments: an overview and a research agenda. Data Sci. J. **12**, GRDI–013 (2013)
11. Coro, G., Candela, L., Pagano, P., Italiano, A., Liccardo, L.: Parallelizing the execution of native data mining algorithms for computational biology. Concurrency Comput. Pract. Exp. **27**(17), 4630–4644 (2015). https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3435
12. Coro, G., Masetti, G., Bonhoeffer, P., Betcher, M.: Distinguishing violinists and pianists based on their brain signals. In: Tetko, I.V., Kůrková, V., Karpov, P., Theis, F. (eds.) ICANN 2019. LNCS, vol. 11727, pp. 123–137. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30487-4_11
13. Coro, G., Pagano, P., Ellenbroek, A.: Combining simulated expert knowledge with neural networks to produce ecological niche models for Latimeria chalumnae. Ecol. Model. **10**(1016), 55–63 (2013)
14. Coro, G., Panichi, G., Scarponi, P., Pagano, P.: Cloud computing in a distributed e-infrastructure using the web processing service standard. Concurrency Comput. Pract. Exp. **29**(18) (2017). https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4219

15. Coro, G., Webb, T.J., Appeltans, W., Bailly, N., Cattrijsse, A., Pagano, P.: Classifying degrees of species commonness: North Sea fish as a case study. Ecol. Model. **10**(1016), 272–280 (2015)
16. Coro, G., Large, S., Magliozzi, C., Pagano, P.: Analysing and forecasting fisheries time series: purse seine in Indian Ocean as a case study. ICES J. Mar. Sci. **73**(10), 2552–2571 (2016)
17. Coro, G., Panichi, G., Pagano, P.: A web application to publish R scripts as-a-service on a cloud computing platform. Boll. di Geofis. Teorica ed Appl. **57**, 51–53 (2016)
18. Deelman, E., et al.: Pegasus, a workflow management system for science automation. Future Gener. Comput. Syst. **46**, 17–35 (2015)
19. Froese, R., Thorson, J.T., Reyes, J.R.: A bayesian approach for estimating length-weight relationships in fishes. J. Appl. Ichthyol. **30**(1), 78–85 (2014)
20. Hey, A.J., Tansley, S., Tolle, K.M., et al.: The Fourth Paradigm: Data-Intensive Scientific Discovery, vol. 1. Microsoft Research Redmond, Redmond (2009)
21. Josep, A.D., Katz, R., Konwinski, A., Gunho, L., Patterson, D., Rabkin, A.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
22. Khalifa, S., et al.: The six pillars for building big data analytics ecosystems. ACM Comput. Surv. (CSUR) **49**(2), 33 (2016)
23. Lebo, T., et al.: Prov-o: the PROV ontology. W3C Recommendation **30** (2013)
24. Li, F., Ooi, B.C., Özsu, M.T., Wu, S.: Distributed data management using MapReduce. ACM Comput. Surv. (CSUR) **46**(3), 31 (2014)
25. Liew, C.S., Atkinson, M.P., Galea, M., Ang, T.F., Martin, P., Hemert, J.I.V.: Scientific workflows: moving across paradigms. ACM comput. Surv. **49**(4), 1–66 (2016). http://doi.acm.org/10.1145/3012429
26. Llorà, X., Ács, B., Auvil, L.S., Capitanu, B., Welge, M.E., Goldberg, D.E.: Meandre: semantic-driven data-intensive flows in the clouds. In: 2008 IEEE Fourth International Conference on eScience, pp. 238–245. IEEE (2008)
27. Ludäscher, B., et al.: Scientific workflow management and the Kepler system. Concurrency Comput. Pract. Exp. **18**(10), 1039–1065 (2006)
28. Myers, J., et al.: Towards sustainable curation and preservation: the SEAD project's data services approach. In: 2015 IEEE 11th International Conference on e-Science, pp. 485–494. IEEE, Munich, Germany (2015). https://doi.org/10.1109/eScience.2015.56
29. Pérez, F., Granger, B.E.: Ipython: a system for interactive scientific computing. Comput. Sci. Eng. **9**(3), 21–29 (2007)
30. Pierce, M.E., et al.: Apache airavata: design and directions of a science gateway framework. Concurrency Comput. Pract. Exp. **27**(16), 4282–4291 (2015)
31. Wolstencroft, K., et al.: The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Res. **41**(W1), W557–W561 (2013)
32. Xie, Y.: Dynamic Documents with R and Knitr. Chapman Hall/CRC, Boca Raton, Florida (2015)
33. Zaharia, M., et al.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016)
34. Zhao, Y., et al.: Swift: fast, reliable, loosely coupled parallel computation. In: 2007 IEEE Congress on Services (Services 2007), pp. 199–206. IEEE (2007)
35. Zhou, H., et al.: CloudsStorm a framework for seamlessly programming and controlling virtual infrastructure functions during the DevOps lifecycle of cloud applications. Softw. Pract. Exper. **49**, 1421–1447 (2019). https://doi.org/10.1002/spe.2741
36. Evans, K., et al.: Dynamically reconfigurable workflows for time-critical applications. In: Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science (WORKS 2015), pp. 1–10. ACM Press, Austin, Texas (2015). https://doi.org/10.1145/2822332.2822339

37. Zhao, Z., et al.: Reference model guided system design and implementation for interoperable environmental research infrastructures. In: 2015 IEEE 11th International Conference on e-Science, pp. 551–556. IEEE, Munich, Germany (2015). https://doi.org/10.1109/eScience.2015.41
38. Hu, Y., et al.: Deadline-aware deployment for time critical applications in clouds. In: Rivera, F.F., Pena, T.F., Cabaleiro, J.C. (eds.) Euro-Par 2017. LNCS, vol. 10417, pp. 345–357. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64203-1_25
39. Hu, Y., Zhou, H., de Laat, C., Zhao, Z.: Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. Future Gener. Comput. Syst. **102**, 562–573 (2020). https://doi.org/10.1016/j.future.2019.08.025