Life Science Workflow Services (LifeSWS): motivations and architecture

Reza Akbarinia, Christophe Botella, Alexis Joly, Florent Masseglia, Marta Mattoso, Eduardo Ogasawara, Daniel de Oliveira, Esther Pacitti, Fabio Porto, Christophe Pradal, et al.

Life Science Workflow Services (LifeSWS): motivations and architecture

Reza Akbarinia[1], Christophe Botella[1], Alexis Joly[1], Florent Masseglia[1], Marta Mattoso[2], Eduardo Ogasawara[3], Daniel de Oliveira[4], Esther Pacitti[1], Fabio Porto[5], Christophe Pradal[1,7], Dennis Shasha[6], and Patrick Valduriez[1]

[1] Inria, Univ Montpellier, CNRS, LIRMM, France
[2] Federal University of Rio de Janeiro, Brazil
[3] CEFET/RJ, Brazil
[4] Fluminense Federal University, Brazil
[5] LNCC, Brazil
[6] New York University, USA
[7] CIRAD, AGAP Institute, Univ Montpellier, INRAE, Institut Agro Montpellier, France

Abstract. Data driven science requires manipulating large datasets coming from various data sources through complex workflows based on a variety of models and languages. With the increasing number of big data sources and models developed by different groups, it is hard to relate models and data and use them in unanticipated ways for specific data analysis. Current solutions are typically ad-hoc, specialized for particular data, models and workflow systems. In this paper, we focus on data driven life science and propose an open service-based architecture, Life Science Workflow Services (LifeSWS), which provides data analysis workflow services for life sciences. We illustrate our motivations and rationale for the architecture with real use cases from life science.

Keywords: Data driven science · Life science · Data science · Workflows · Model life cycle · Service-based architecture.

1 Introduction

Data driven science such as agronomy, astronomy, environmental, and life science must deal with overwhelming amounts of complex data, e.g., coming from sensors and scientific instruments, or produced by simulation. Increasingly, scientific breakthroughs will be enabled by advanced techniques from data science [23] that help researchers manipulate and explore these massive datasets [14].

Life science is the study of living organisms (plants, humans, micro-organisms, ...) and their association with internal or external conditions. It is an interdisciplinary domain including agronomy, biology, and botany. The data in life science comes from many different data sources produced by modern platforms, e.g., high-throughput phenotyping, next-generation sequencing, remote sensing, etc., or readily available as international databases, such as Data.World, GenomeHub,

AgMIP, EMPHASIS, etc. Such data is used to help producing/training models (statistical models, machine learning (ML) models, etc.) to derive information and knowledge or to make predictions using complex workflows. Since models are tailored to specific research questions, they are typically produced by different research groups and take various forms that reflect the researchers' approaches with their data.

Data processing with models typically involves complex data analysis workflows (*workflows*, for short hereafter). Unlike business workflow systems, e.g., new order processing, these workflows are compute- and data-intensive, may take hours or even days, but are often deterministic, and do not involve fine-grained transactions. They allow domain scientists (specialized in a science domain, e.g., plant biology), to express multi-step computational activities, such as loading input data files, processing the data, running analyses, and aggregating the results. Workflows have been implemented on top of scientific workflow systems such as Galaxy [1] and OpenAlea [30]. They frequently make use of data analytics engines such as Spark [36] and Flink [6], as well as Machine Learning (ML) libraries such as PyTorch [25] and Scikit-learn [26]. In order to scale to massive datasets, they make increasing use of distributed and parallel execution environments in the cloud.

While this paper (and project) focuses on data-driven life science, we believe the project can provide a framework for other application domains with similar requirements.

1.1 Use Cases

Let us illustrate the requirements for managing such data and models with real application examples from life science. In the context of climate change, agro-ecosystems face multiple challenges, including adaptation, resilience, epidemics, land-use conflicts, and the need for biodiversity conservation. Examples of practical questions that end-users might ask are:

- How to select or breed new plant varieties that are adapted to my local environmental conditions (e.g., drought, flooding, high temperature, disease)?
- Which treatments should be deployed on my farm depending on climatic conditions and geographical proximity to disease hot spots?

Addressing these questions requires multiscale modeling, e.g., modeling plants at different scales (e.g., organ, plant, crop, land surface, region) to predict the impact from heterogeneous data, e.g., data on plants, environment (weather, soil), and remote sensing. These models are the outcome of workflows, whose activities typically involve data extraction, data cleaning, machine learning, and visualization. Often the output of one workflow is the input to another.

With One Health, an approach that recognizes that the health of people is closely connected to the health of animals and our shared environment, understanding epidemic propagation at various levels (local, regional, national, global) has become critical for health authorities. The major problem is how to select

the best prediction model for a given region by combining propagation models from different regions as well as integrating various data sources (epidemic, climate, socio-economic, etc.) along some common dimensions, e.g., time, location, etc.

The practical difficulty to achieve such integration is that it is hard to relate models and data, which are typically produced by different people with different methods, formats and tools. International repositories for scientific data and models are useful but they tend to be specialized for specific purposes and research communities, e.g., genomics, phenotyping, and epidemiology. Similarly, the workflow systems to manipulate data and models are specialized for a research domain, e.g., OpenAlea for plant phenotyping, Galaxy for genomics. Thus, there is a pressing need for integrated data and model management in order to achieve consistency and ease of use through generic workflow services with the ultimate goal of improving model accuracy and predictions.

1.2 The Centrality of Workflows

In this paper, we propose an open service-based architecture, called Life Science Workflow Services (LifeSWS). The main objective of LifeSWS is to help managing complex workflows by organizing massive and heterogeneous data, in connection with models and making workflow artifacts (datasets, models, metadata, workflow components, etc.) easy to search, debug, and parallelize.

In many ways, workflows are to scientific data processing what queries are to business data processing. In business data processing, queries must be written (with some reuse of other queries), debugged and optimized (sometimes through parallelization), and should work across distributed servers, hardware, and operating systems. Scientific data processing is much more complex so workflows replace queries. The issues however are much the same. Workflows in scientific data processing must be written (with some reuse of other workflow components), debugged (often benefiting from provenance), optimized (often through parallelization and caching), and should work across distributed servers and operating systems. In addition, workflows should be fault tolerant and workflow component versions should be kept up-to-date. Thus, a technical goal of this project is to make workflows work as seamlessly with data as queries do in business processing.

LifeSWS capitalizes on our previous experience in developing major systems for scientific applications such as: polystores with CloudMdSQL [17], workflows with OpenAlea [30], model management with Gypscie [35] [38], querying data across distributed services with DfAnalyzer [32] and Provlake [33], monitoring and debugging applications implemented in big data frameworks such as Apache Spark [12], and debugging workflows with BugDoc [18] and VersionClimber [29].

1.3 Paper Outline

The paper is organized as follows. Section 2 develops our motivating examples from real life science applications. Section 3 presents our open, service-based

architecture for LifeSWS. Section 4 discusses platforms and infrastructures that can implement LifeSWS. Section 5 shows the use of LifeSWS with use cases from our motivating examples. Section 6 discusses related work. Section 7 concludes and discusses open research issues.

2 Motivating Examples

In this section, we introduce examples from real-life science applications that will serve as motivation for our work and as the basis for use cases with LifeSWS. These examples are in agro-ecosystems in the context of climate change and epidemic modeling. These examples share common requirements but have specific features that will show different uses of LifeSWS.

2.1 High-Throughput Phenotyping in the Context of Climate Change

As observed above, agro-ecosystems face multiple challenges, including climate change, epidemics, land-use conflicts, and the need to conserve biodiversity. To enhance the resilience of agro-ecosystems, interdisciplinary efforts are required, ranging from a detailed biological understanding of the physiology of plants with multiple stresses (e.g., drought, temperature, decease), agronomy to adapt agro-ecosystems to future challenges, as well as sociology, economy and politics to understand the impacts of changing public policy.

This challenge requires mobilizing all possible levers of plant adaptation, including the genotype, phenotype, and their interactions with the environment. The genetic/genomic revolution has allowed us to sequence and manipulate genes at a low cost and to generate an avalanche of information. But understanding genome-to-phenotype relationships is crucial. While national and international phenotyping platforms allow the capture of phenotypes at high-throughput, most of the traits that contribute to the performance of agro-ecosystems are environment-dependent. The performance of a variety that thrives in a particular environment may perform poorly in a different one. Thus, it is important to capture phenotypes in various environments using various sensors at different scales (IoT, images from drones, 3D point clouds from Lidars and remote sensing images from satellites).

Major efforts have been invested in crop breeding to improve crop yield for food security. However, profiling the crop phenome by considering the structure and function of plants associated with genetics and environments remains a technical challenge [34].

In the past decade, high-throughput phenotyping platforms have emerged, enabling the collection of quantitative data on thousands of plants under controlled environmental conditions. A good example is the French Phenome project, with seven facilities producing 200 Terabytes of diverse, multiscale data annually, including images, environmental conditions, and sensor outputs from different sites [13].

To support high throughput phenotyping, many workflows have been developed using OpenAlea to analyze, reconstruct, and visualize the spatial and temporal development of the geometry and topology of thousands of plants in various environmental conditions. For instance, the *Phenomenal* workflow supports the reconstruction in 3D and the segmentation of plant organs [2]. The *PhenoTrack* workflow, which is based on Phenomenal, allows the 3D reconstruction of plants with the temporal tracking of the growth of each organ for the entire developmental cycle [9]. Finally, *RootSystemTracker* provides a workflow for the automatic structural and developmental 2D root phenotyping of Arabidopsis plants in Petri dishes [10].

Figure 1 shows a) the Phenomenal workflow implemented in OpenAlea; b) 3D organ tracking of a maize plant with PhenoTrack3D [9]; and c) a reconstructed root system architecture through time using RootSystemTracker [10].

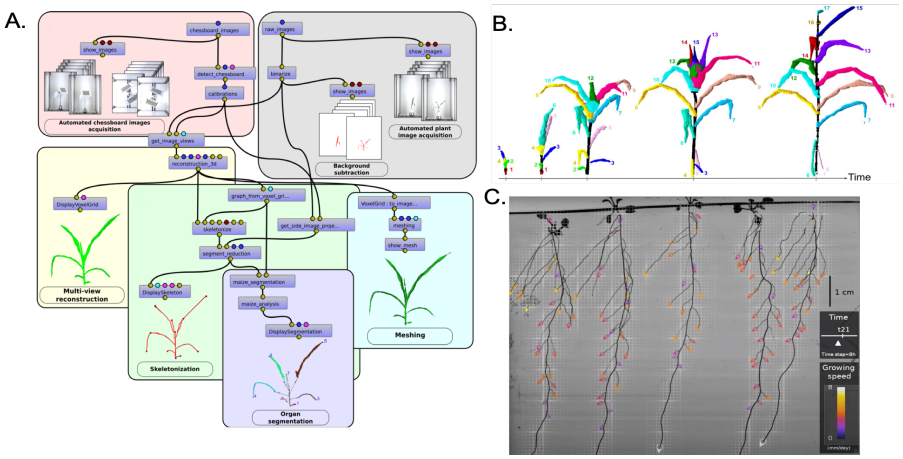

Fig. 1. Spatial and Temporal Workflows of Maize Shoot and Arabidopsis Root System Architecture

These workflows need to process large volumes of data on distributed infrastructures. To execute these workflows, we need to: 1) transfer large image datasets from a data center close to the phenotyping platforms to computing servers in the cloud; 2) distribute the execution on a cloud or grid infrastructure; 3) capture the provenance of the execution and cache intermediate results for later use; 4) rerun workflows with new processes and parameters; 5) provide execution results using dashboards to check the execution.

Furthermore, to understand the genotype-to-phenotype relationships, we need to be able to relate plant traits computed by phenotyping workflows (e.g., with OpenAlea) with genetic information using genotyping workflows such as genome-wide association studies (e.g., with Galaxy). Thus, we need to integrate heterogeneous workflows and be able to schedule their execution.

Finally, through time, phenotyping workflows evolve with new processes implemented in various libraries whose versions and dependencies change quickly. Parameters need to be calibrated on new phenotyping platforms which have new sensors, different light conditions, or new plant species. Furthermore, it is important to identify problems in the workflow specification (e.g., that may lead to deadlocks) before executing them in HPC environments to avoid sparing resources. Finally, the workflows need to be debugged to identify problems occurring in new settings.

2.2 Epidemic Modeling

Each year, dengue, zika, chikungunya, and other arboviruses disseminated by the Aedes Aegypti vector exert an extreme burden on populations' health, especially in low-income countries.

General statistical models that try to explain or predict dengue in large areas usually do not consider the diversity of the territory and the different or even contradictory relations that predictors can preserve with the outcome. Conversely, creating individual models for every possible geographic location is impractical and unfeasible. In addition, there are regions for which we don't have enough data to create prediction models.

Therefore, a more effective approach is to develop Machine Learning models tailored to the unique characteristics of each region, considering the specific meteorological, socio-economic, and sanitary conditions that affect the epidemic transmission in that area. Then, these models can be used for predicting the transmission in similar regions for which constructing specific models is not possible (due to lack of data). For efficient modeling of dengue and other arboviruses, we need tools that can facilitate the selection of ML models that are most suitable for predicting these viruses. By utilizing these tools, more accurate predictive models can be selected and used to better understand and prepare for the transmission of viruses in the given region.

However, providing these tools is challenging. The reason is that we need to gather different datasets and models, and develop novel algorithms to enhance the accuracy and reliability of the prediction models. The required datasets and models are as follows: 1) *propagation datasets* that contain information about the spread of disease; 2) *climate datasets* that provide crucial insights into environmental factors like temperature and precipitation that may impact disease transmission; 3) *socio-economic datasets* that help us to understand the social and economic factors that could influence disease spread; 4) *prediction models* generated for some regions. By utilizing these diverse datasets and models, we should be able to select more accurate and reliable models for query regions, which can ultimately contribute to better disease control and prevention strategies.

Figure 2 shows the Dengue cases in Brazil spanning from 2000 to 2019 (a), along with the geographical distribution of cases across various regions (b and c) [5]. For predicting the Epidemy in each region, we need to select a model (or set of

models) that takes into account the specific meteorological and socio-economic characteristics of the region.

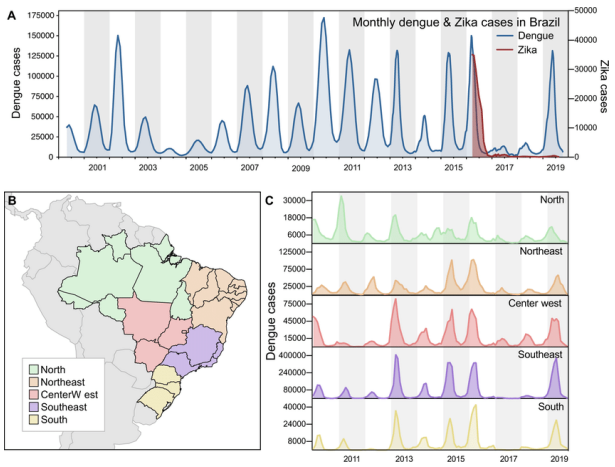

Fig. 2. Dengue Epidemy in Brazil

3 Architecture of LifeSWS

In this section, we introduce the service-based architecture of LifeSWS, with its functional architecture and three layers of services (presentation and directory, workflow and data management services).

3.1 Functional Architecture

Our design choices are guided by the requirements of our users. The main potential users of LifeSWS are: the domain scientists who wish to analyze the data using different models and workflows; the workflow providers who create, maintain or enhance workflows for domain scientists using their workflow tools; the model providers who build models; and the data providers who supply data sources to the workflows.

Our architecture capitalizes on the latest advances in web-oriented architectures, microservices, containers and distributed and parallel data management [24]. We adopt the main following design choices and principles:

- Ease of use through web interfaces, which are easy to develop and specialize for different kinds of users;
- Open architecture with open source services and tools, and well-defined APIs to foster services interoperability (like cloud web services);

- Distributed architecture to provide performance, scalability and ease of use in the cloud using distributed database principles;
- Support for various databases (SQL, NoSQL, SciDB, etc.) and scientific file types (e.g., HDF and NetCDF);
- Integrated services on top of various databases, which can be local (in the same data center) or remote (in remote data centers) and access to various tools and execution environments.

LifeSWS's functional architecture is shown in Figure 3. It has three main layers of services: (1) presentation and directory, (2) workflow and (3) data management. Each layer can use the services of the same layer or the layers below. To interface with different systems, services can also use three kinds of APIs: Worflow Access APIs, Data Source APIs and Data Store APIs.

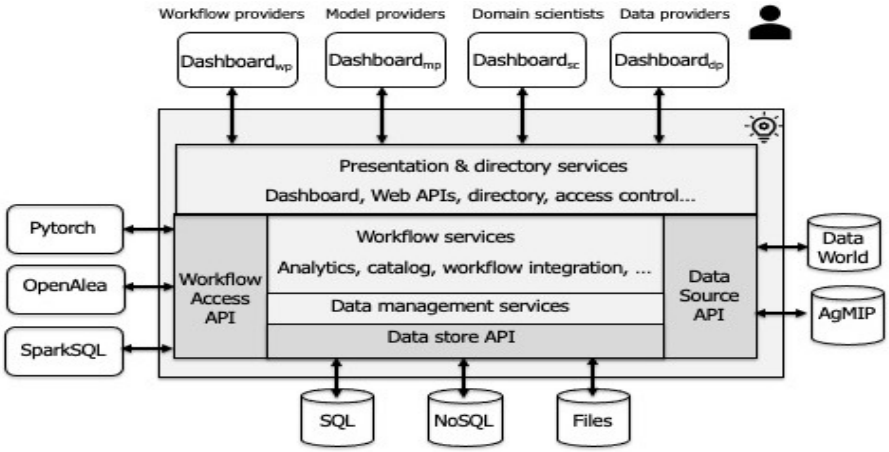

Fig. 3. LifeSWS Architecture

3.2 Presentation and Directory Services

Presentation and directory services provide users and applications with secure ways of accessing LifeSWS services. Presentation services include a Web dashboard service, a Web API and a directory service.

The Web dashboard service allows LifeSWS developers to build specific dashboards for different types of users (domain scientists, workflow providers, model providers and data providers). These dashboards allow users to analyze and display real-time data as charts and reports. They offer the following capabilities to developers: (1) a directory to publish and/or find data sources and workflow components; (2) tools for assembling workflows easily; (3) tools for debugging; and (4) scheduling workflows using workflow systems.

The directory stores data about LifeSWS users, access rights, dashboards and services. As a user directory, it helps register users, find out about them as well as authenticate them when accessing LifeSWS services. As a service directory, it provides a single place to publish, discover, and connect LifeSWS services as well as external services that can be distributed over the network. Additional network security, e.g., firewall, can be provided at this layer.

As an alternative to Web dashboards, the Web API is a server-side API that allows LifeSWS developers to access LifeSWS services from more general Web applications. This API consists of one or more publicly exposed endpoints that specify where and how to access the services with a request–response protocol, typically in JSON.

Finally, LifeSWS offers an external data view to ease the development of dashboards and workflows, which integrates observational and predictive data. This external view can be represented by a knowledge graph [15] extended to support the representation of observation time-series and predictive information metadata, such as: error estimate, multi-class prediction probabilities and etc..

3.3 Workflow Services

Workflow services make it easy for scientists to develop, debug and optimize their workflows for doing their scientific experiments and data analyses. The services should also support the of sharing data, models and workflow components. Because the users want to be able to use their familiar tools (e.g., workflow systems such as Galaxy or OpenAlea) and data sources (e.g., Data World), this layer provides efficient services to register and manage data and models, and allow model execution using different tools and data sources. The primary services provided at this layer are: catalog (including version management), provenance and cache, data analytics, and data management.

Catalog. The catalog is the central place to find out about all artifacts and tools of interest for LifeSWS users: data sources, datasets, models, workflows and code libraries. Artifacts can be found outside LifeSWS and thus accessed through some API, or stored within LifeSWS for efficient reuse. Each artifact has associated metadata that describes it and allows access to it, either locally if it is stored in LifeSWS, or through its URI if it is an external resource (tool or data source). With the catalog, one may register artifacts, change them or provide a new version. The catalog also knows about tools (e.g., OpenAlea, Spark) and code libraries that implement models (e.g., Phenomenal workflow). Finally, the catalog comes with a search capability that allows users to navigate through the hierarchy of artifacts.

Model Management. Various types of models are used in life sciences. Data-driven machine learning models adopt a learning strategy that updates a set of weights that approximates a function to the behavior of the learned phenomenon given by the patterns extracted from the input data. Typical tasks executed by

machine learning models include solving classification and numerical regression problems, which one may generalize as prediction tasks. Another relevant type of model extensively used in life science are mechanistic models, which refer to computational artifacts derived from the mathematical modeling of a phenomenon. The product of running mechanistic models for a certain number of time steps is referred to as a phenomenon simulation. For instance, crop simulation models reproduce the main functions of plants such as the evolution of plant architecture, light interception, photosynthesis, and water/nitrogen balance in the crop and soil [21].

The management of such life science model artifacts requires model life cycle management and model deployment, using specific tools that can be accessed through LifeSWS. Through a unified view of different model artifacts (produced with different tools), LifeSWS can improve model selection and allow for model integration.

Model selection allows the user to easily search for model artifacts of interest so they can be used for reproduction or integration. Searching can be done based on different criteria such as scientific domain or subdomain, metadata, format, tools and keywords. This capability uses the catalog of artifacts.

The performance monitoring of models in operation by the model management service is important to assess prediction quality and point to model updates. In particular, if the input data distribution changes, models built on past historical data must be flagged so they can be updated. For machine learning models, a concept-drift component must detect variations in input data patterns and launch alerts for downstream model updates. The latter are processed according to application requirements. LifeSWS supports complete and automatic model retraining, using ML tools, or it can delegate the model update process to components that implement a more sophisticated update procedure, involving for instance a fast training of a simple surrogate model, while the main model is updated.

Model integration allows combining different models, possibly produced using different tools. It can take different forms, depending on the model types and the integration objective. In machine learning, model integration may take the form of an ensemble of models [37]. An ensemble considers a set of models aiming at the prediction of the same target. The integration process is modeled as a pipeline that runs each individual participant model, possibly across different tools, over the same input and combines the individual results into an integrated one, often using a linear combination of the results. Using the DJEnsemble method [27], ensembles can be computed automatically by a LifeSWS platform such as Gypscie (see Section 4), so that the selection of participant models follows a cost-based selection approach. Model integration takes a different form in mechanistic models as they typically use scientific workflows for simulating the phenomenon. For example, to visualize a 3D model of plant growth in a local environment, the 3D plant structure and biophysical models such as light interception, carbon allocation and water and mineral uptake can be simulated by a workflow modeled in OpenAlea.

Workflow Integration. This service provides support for integrating and efficiently executing workflows on different workflow systems using the Workflow Access APIs. It shares some similar goals and functions found in data integration. The main functions provided by this service are workflow definition and execution, provenance and cache.

For workflow definition and execution, we plan to rely on the Common Workflow Language (CWL) [8], an open standard aiming to enable scientists to share complex data analysis and machine learning workflows. CWL supports connecting command line tools to create workflows that are portable across a variety of CWL-compliant platforms, from a single developer's laptop up to a massively parallel cluster in the cloud. The CWL project produces free and open standards for describing command-line tool based workflows. These standards are implemented in many popular workflow systems such as Galaxy, Pegasus, Streamflow, and CWL-Airflow. To enable portability and reusability, CWL is explicit about inputs/outputs to form the workflow, data locations and execution models, which can be deployed using software container technologies, such as Docker and Singularity.

Within the CWL project, we can contribute to the definition of integrated workflows that span multiple workflows and workflow systems. Once an integrated workflow has been defined and its mappings registered using CWL, it can be executed using a LifeSWS scheduler that orchestrates execution across different workflow systems, in connection with these systems' schedulers.

Provenance (also referred to as lineage) management helps to reproduce, trace, assess, understand, and explain how datasets have been produced. This is a useful underlying functionality for several strategic capabilities, including experimental reproducibility, user steering (i.e., runtime monitoring, interactive data analysis, runtime fine-tuning) and data analysis. These capabilities are essential building blocks towards the goal of storing and sharing results of executions that can be useful later (by the same or different users perhaps on very different platforms).

In addition to provenance management, this service includes cache management, using information about cache data, as well as the location of the cache data (e.g., files, Spark RDDs, ...). Caching datasets improves performance when they are produced at multiple times by different users or distributed at various sites. The decision whether to cache an intermediate result can be explicit (i.e., decided by the user) or made automatic based on workflow fragment analysis [13].

All information for this service is stored in a database that is relatively small. In particular, the cache itself is small (only references) and the cached data can be managed using the underlying execution environments accessed through the Workflow Access APIs. Using a database for this service provides the traditional advantages of data sharing, integrity and querying using an SQL-like language.

Analytics. This service allows scientists, through their specific dashboards, to perform analytics on the data produced by their workflows using the other

workflow services. Using the Catalog and Model Management services, the user is able to select models and datasets of interest, execute the selected models using various workflow execution environments (using the Workflow Access APIs), and analyze the results. It also allows the user to cache and explain the results, and reproduce executions using the Provenance and Cache services.

This service also facilitates the analysis of different types of data such as time series and spatial data, by incorporating advanced analytical techniques like anomaly detection, similarity search and clustering.

3.4 Data Management Services

These services make it easy for LifeSWS users to manage their artifacts (datasets, models, metadata, etc.) and session data (logs, intermediate datasets, etc.) with high-level capabilities using the Data Source, Model/Workflow and Data Store APIs. An important capability is moving data between different data sources, databases and execution environments using simple import-export functions. Another useful capability, using the Data Source APIs, is subscribing to some data sources that provide a publish API, to get warned of the new versions.

More advanced capabilities, similar to distributed databases [24] and polystores [3], could be provided at this level for integrating data from different data sources. In particular, the CloudMdsQL polystore [17] is efficient for querying multiple heterogeneous data sources (e.g. files, relational and NoSQL) in the cloud. A CloudMdsQL query may contain nested subqueries, and each subquery addresses directly a particular data store and may contain embedded invocations to the data store native query interface. Thus, the major innovation is that a CloudMdsQL query can exploit the full power of local data stores, by simply allowing some local data store native queries to be called as functions, and at the same time be optimized based on a simple cost model, CloudMdsQL can also access address distributed processing frameworks such as Apache Spark by enabling the ad-hoc usage of user defined map/filter/reduce operators as subqueries.

Data Store APIs. These APIs allow storing and accessing data in different data stores (SQL, NoSQL, Savime, files, ...), to support specific requirements. For instance, the catalog, provenance, and cache databases are typically in an SQL database such as PostgreSQL. By contrast, external data could be stored in the original format, e.g., JSON in a NoSQL document database, and extracted using data-specific APIs. Datasets produced using tools or cache data could be stored in files, e.g., Parquet, etc., or in a scientific database like Savime. The data store APIs should be based on standard APIs, such as JDBC, file system APIs, ...).

Data Source APIs. These APIs allow connecting to various web data sources, such as Data World, AgMIP, etc., and performing various tasks (search for datasets, extract a dataset's metadata, import a dataset, get changes, etc.).

They can be used to build user-friendly dashboards for domain scientists with semantic-based search capabilities, as for instance in the ontology-driven Phenotyping Hybrid Information System (PHIS [22]).

Workflow Access APIs. These APIs allow accessing and manipulating models and workflows as structured objects with their own semantics, execute them in their own execution environments, such as Pytorch (ML models), OpenAlea (workflows) and Spark (e.g., SparkSQL queries). These APIs make it possible to simply perform various tasks using models and workflows, such as import or export of models, executing them using a dataset, saving the result data (using the Data Store API, ...).

4 LifeSWS Platforms

LifeSWS services can be implemented and deployed in various platforms (using different software and hardware infrastructures) to address the specific requirements of vertical applications. Examples of platforms would be some LifeSWS services deployed in the cloud (public, private or hybrid) or on-premise clusters of servers, reusing existing software components that (partially) implement the services.

A good example of a LifeSWS platform is Gypscie [38], which provides services to develop, share, improve and publish scientific artifacts (datasets, models, etc.). Gypscie's services are available through two different interfaces. Figure 4 shows the interface that enables interactive access to services, including artifacts registration and service requests. The same functionality is available through a REST API based on the HTTP protocol.

These services make it easy for model providers and scientists to:

- Collect, curate and integrate heterogeneous data;
- Support the complete ML model life cycle, from model building to deployment, monitoring and policy enforcement;
- Find ready-to-use models that best fit a particular prediction problem;
- Compare and ensemble models;
- Execute models with various tools: ML engines, workflow systems, ...
- Use specific hardware infrastructures and corresponding algorithms according to a desired task, e.g., use a distributed training algorithm for a particular GPU-based server for training a large deep neural network.

Let us illustrate how LifeSWS services would be supported by Gypscie services for presentation & directory, model management, and data management.

4.1 Presentation & Directory

Gypscie offers a web interface that eases ML model management. It also offers a notebook interface for direct python scripts integration with the Flask framework. Furthermore, Gypscie enables other tools to access its services through a

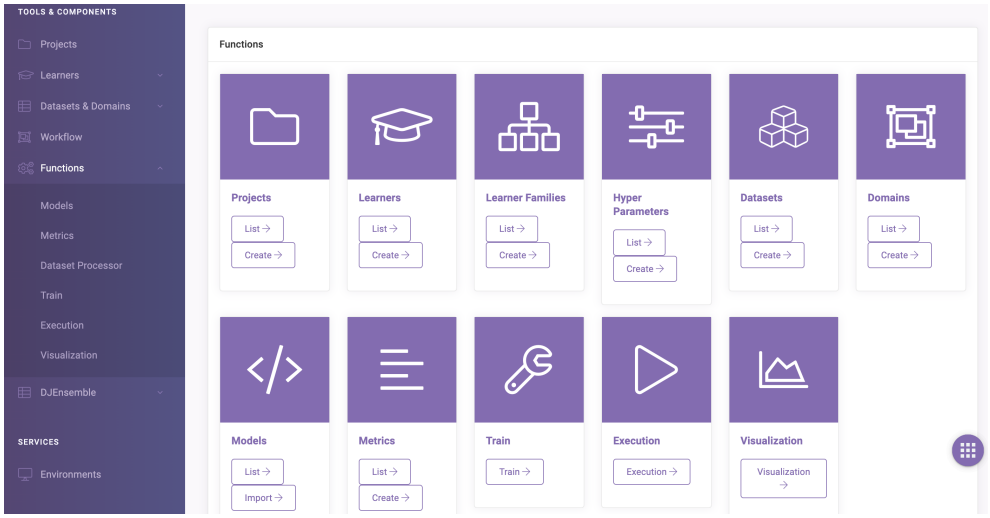

Fig. 4. Gypscie Web Interface

RESTful-based API. Users can build dataflows graphically to model data preprocessing tasks. Registered dataflows can be scheduled for execution and, during runtime, have their activities and involved data recorded for provenance. User dataflows typically include data pre-processing transformations in preparation for model training and model inference.

4.2 Model Management

The core functionalities of Gypscie cover the services needed to support the full ML life cycle. Regarding model management, the Gypscie data model fosters the reuse of all artifacts involved during the model's life cycle. As such, the user can publish the scripts involving the data preparation and model fitting for a particular learning algorithm (hereafter, denoted *learner*). For ease of browsing, learners are aggregated into learner family. We use the learning artifact to build models by providing the necessary training dataset. In addition, Gypscie allows models built in other external systems to be imported and registered into it. Thus, models can be automatically registered when built using a known learner and the Gypscie training functionality, or they can be manually imported. In both cases, once registered they are ready to be called for inference. The functionalities involving model training and inferencing are both implemented using *MLFlow*. The latter enables Gypscie to communicate with the most frequently used ML engines, such as Pytorch and Scikit-learn. Gypscie instruments the running scripts to register in MLFlow the values of performance metrics, which the system collects and stores into its catalog, once the job has finished its execution.

A particular feature of Gypscie is its ability to deal with spatio-temporal data, which are extremely common in scientific applications. Gypscie implements

the DJEnsemble [27] inference approach. The idea is to automatically select trained spatio-temporal models with performance guarantees for the scientific predictions. The approach considers a set of registered models for the execution of a certain task, for instance, rainfall prediction. The algorithm uses a cost-based strategy that strikes a balance between prediction precision and execution cost to select the best set of models that infer the rainfall prediction in a region of the space. Additionally, it specifies how to spatially allocate the selected models to cover the query region. Gypscie runs the optimization process, executes the selected models, and composes the final result. This is a very complex task that is completely abstracted from the final user, showing the potential of LifeSWS to create an easy ML environment.

4.3 Data Management

Data management involves the following services: (1) accessing registered data; (2) gathering provenance information, and (3) exploring the content of datasets. Gypscie registers metadata in its catalog for accessing data stored in an external data source, such as Databricks Delta Lake or Lustre file system. When a scheduled workflow requires a dataset, the dataset is automatically transferred from its stored location to a file system supporting the workflow execution environment.

As a workflow applies transformations on a dataset, Gypscie stores the provenance information regarding the operation. Thus, within Gypscie a user can always review the lineage of transformation that led to the dataset's current version.

Finally, Gypscie integrates the SAVIME in-memory multi-dimensional array database system [20]. SAVIME supports the expression of SQL-like queries over raw datasets. The query language enables the registration of prebuilt ML models that can be invoked over the results of a query expression.

Gypscie has been deployed on a server at LNCC, and interfaced with two execution environments (see Section 5 : the Santos Dumont HPC system at LNCC which could be used in large model training (e.g., using PyTorch) and a Spark shared-nothing cluster to perform large-scale data transformation.

5 Use Cases with LifeSWS

In this section, we show how the LifeSWS services could be used to support the requirements of the two motivating examples, referring to the services of the previous section.

5.1 High-Throughput Phenotyping

In this use case, LifeSWS is used by domain scientists in order to analyze, process and visualise High Throughput Phenotyping (HTP) experiments, with the following workflows, models and datasets:

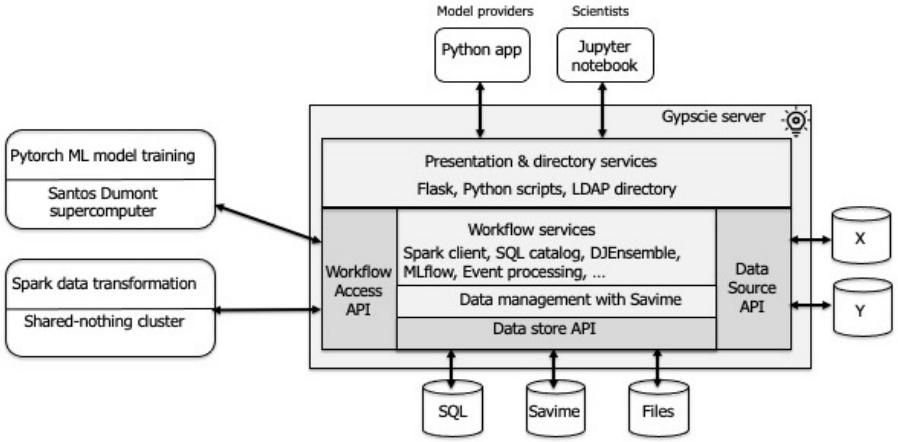

Fig. 5. LifeSWS platform with Gypscie

- OpenAlea workflows that implement a specific phenotyping processing such as 3D maize segmentation, organ tracking, or root system reconstruction;
- A Galaxy workflow for a Genome-Wide Association Study (GWAS);
- Image analysis algorithms to segment the background, to reconstruct the plant in 3D using space carving, semantic segmentation and tracking of the organs;
- Functional-structural plant models that are used either to compute non-observed information like light interception by leaves [2] or water fluxes inside the root system [4], or to generate synthetic data for training or validating methods of plant reconstruction;
- Raw image datasets obtained from the phenotyping platform, which contain timeseries of several images per plant;
- Outputs of plant traits (e.g., leaf angle, light use efficiency, or biomass accumulation) for each genotype are saved in the Phenotyping Hybrid Information System (PHIS) [22].

A domain scientist, often a plant biologist, searches for a specific OpenAlea phenotyping workflow based on its metadata. Then, she edits, visualizes, and executes the workflow on a small dataset. For instance she can reconstruct in 3D the growth and development of a maize plant during a growing season. Finally, she selects a full dataset from an existing phenotyping experiment and executes the workflow to obtain a set of plants traits specific to each genotype and to environmental conditions such as drought or temperature. Finally, to breed varieties tolerant to drought, some outputs of the workflow will be associated with genetic markers using a Galaxy workflow that implements a Genome-Wide Association Study (GWAS). GWAS allows to correlate favorable traits (e.g., responsible for drought tolerance) with a genomic region and thus to breed new varieties.

For instance, the Phenomenal workflow (Figure 1.a) is composed of different fragments, i.e. reusable sub-workflows: binarization, images calibration, 3D volume reconstruction, and organ segmentation. The intermediate datasets are also shown in the Figure. The raw data is produced by the Phenoarch platform, which has a capacity of managing 2500 plants within a controlled environment (e.g., temperature, humidity, irrigation) and automatic imaging through time. The total size of the raw image dataset for one experiment is 15 Terabytes. The raw data is stored on a server close to the experimental platform, but also referenced to the PHIS with all metadata.

To execute the Phenomenal workflow, the OpenAlea scheduler checks, using provenance data, whether some fragments have already been executed and are present in the cache. OpenAlea then schedules the execution of the workflow on a distributed infrastructure. After execution, the user can visualize the results either as classical plots or as 3D plots to inspect the reconstructed plants. The results are automatically stored in the PHIS, to associate each plant of each genotype with environmental data and the computed traits. These results are used as input of the Galaxy workflow to make a complete GWAS study.

Let us now explain how this use case can be realized using the services provided by LifeSWS. Required workflows of the use case can be searched and found using the Catalog service. This allows to navigate among OpenAlea and Galaxy workflows and to select the Phenomenal and GWAS workflows. Both workflows are composed of versioned tools, models, and workflow fragments that are retrieved from the Model Management service. Workflows are visualized and parameters are set via workflow dashboards of the Presentation and Directory services. After edition, new versions are stored using the Model Management service. Datasets of the Phenomenal experiment can be retrieved and accessed using the Data Source APIs with a connection to the PHIS.

LifeSWS looks up the provenance and cache, and triggers OpenAlea's distributed execution. Provenance supports the determination of whether a workflow fragment has been already computed with the same parameters and datasets. The cache enables the retrieval of previous intermediate results rather than recomputing them again. The cache and provenance information are updated during the execution of OpenAlea workflows using the Workflow Access API. LifeSWS provides Data Services to feed the Galaxy workflow with the output of OpenAlea workflow when execution has been done. Then, it triggers the execution of Galaxy on distributed resources.

The visualization of the intermediate and final results is done in a workflow dashboard and some specific outputs (e.g., plant traits) are updated in the PHIS using the Data Source APIs.

Moreover, OpenAlea models and workflows can be upgraded automatically using VersionClimber[29] from the Workflow Integration services to find the latest compatible versions of all the modela and libraries the workflows depend on.

5.2 Epidemic Modeling

In this use case, LifeSWS is used by scientists in order to select ML models that would perform best to predict the transmission of the Dengue virus in a particular region.

In this scenario, the following inputs may be used by LifeSWS:

- Propagation datasets obtained from public health institutions, which are likely to contain information on the spread of disease.
- Publicly available climate datasets, which could be used to identify environmental factors, such as temperature and precipitation, that may impact disease transmission.
- Socio-economic datasets publicly available for specific countries and regions, which may provide insight into social and economic factors that could influence the spread of disease.
- Prediction models provided by the system's users for different regions.

Let r be a region in a country given by a user, the objective is to find the best models that can predict the dengue transmission for r. If there are available models for the given region, then they are returned to the user.

Otherwise, the system should execute the following workflow activities to analyze the characteristics of region r, and find the appropriate prediction models. In this case, the system performs a similarity search between the variables representing r (e.g., meteorological and socio-economical variables) and those of the regions for which it has predictive models in its database. Using this similarity search, LifeSWS identifies the most similar region to r. Finally, it retrieves the predictive models associated with the most similar region and returns them as the best models for predicting Dengue transmission in r.

Let us now explain how this use case can be realized using the services provided by LifeSWS. The required datasets of the use case, mainly propagation, climate and socio-economic datasets can be accessed using the Data Source APIs. The prediction models are given to LifeSWS via the dashboards of the Presentation and Directory services. Then, they are stored using the Model Management service. The Catalog is used to index the metadata of the given datasets and models.

The whole workflow for finding the prediction models of the given region is executed by the scheduler of the Workflow Integration services. The users send their region r to LifeSWS via their dashboard. Then, the system checks the Catalog to determine whether there are any predicting models for the given region. If the answer is positive, the models are retrieved from the Model Management service. Otherwise, LifeSWS needs to find the similar regions to r and return their predicting models. For this, LifeSWS first uses the Data Store service, to find the metadata of the query r and other regions. Then, the Data Analytics service is used to perform a similarity search in order to find the most similar regions to r, which we denote as R. LifeSWS uses its Catalog to select the best models for predicting the transmission of the Dengue virus in the regions similar to r. Then, it calls the Cache management service to retrieve the selected models

if they are available in the cache. Otherwise, the Model Management service is used to access the selected models, which are then returned to the user using the dashboards.

6 Related Work

To address our objectives, many different approaches and solutions could be used with different trade-offs between development and maintenance cost, generality and efficiency. In this section, we discuss the main practical approaches and related technologies in a wide spectrum from generic to specific: cloud services, scientific workflow systems, heterogeneous data management systems, model life-cycle frameworks, and science platforms.

At one end of the spectrum (the most generic approaches), we have cloud services from major vendors (Amazon, Google, Microsoft, IBM, Oracle, ...). They provide many ready-to-use services within a Platform-as-a-Service (PaaS) to build applications that deal with Web data and enterprise data. They focus on ease-of-use, elasticity and interoperability through well-defined APIs that allow to use proprietary as well as open-source software. For instance, Amazon Web Services (AWS) is a large cloud computing platform, offering 200+ services, from basic services (storage, computing, database, containers, security, ...) to more advanced services (machine learning, data warehouse, data lake, search, ...). However, a first reason that prevents the use of such cloud platforms for LifeSWS is the lack of services directly available for scientific applications (workflows, provenance, numerical simulation, interface to HPC systems, ...). Another important reason for scientific organizations is that they prefer to rely on open vendor-neutral vendors.

At the other end of the spectrum (the most specific approaches), we have scientific workflows systems, such as Galaxy [1], Kepler [19] and OpenAlea [30], which are designed to help scientists developing complex applications. They typically include tools to model, design, debug, share and execute workflows, with interactive visualization of the results. To support result analysis and explaining and experiment reproducibility, workflow systems often support provenance, which captures the derivation history of a dataset, including the original data sources, intermediate datasets, and the computational steps that were applied to produce this dataset. Workflows are also often data-intensive, i.e., process, manage or produce huge amounts of data. Thus, in order to be executed in reasonable time, they require deployment in High Performance Computing (HPC) environments such as supercomputers, computer clusters or grids. For instance, DfAnalyzer [32] is a tool that enables monitoring, debugging, steering, and analysis of dataflows while the data is being generated by scientific applications. Most workflow systems are also open-source, providing access to community shared resources such as models, code libraries, and datasets. Thus, they tend to be specialized for some scientific domains. For instance, Galaxy is quite popular in bioinformatics, while OpenAlea is specialized in plant phenotyping. Kepler [19]

addresses other scientific domains such as chemistry, ecology, geology, molecular biology and oceanography.

More generic, we have the popular big data analytics engines such as Spark [36] and Flink [6] which allow for batch or realtime data processing, and ML libraries such as PyTorch[25] and Scikit-learn [26] with workflows to collect training data, preprocess data (cleaning, formatting, ...), build datasets, train and refine models and evaluate.

As workflows are getting used a lot in practice, the problem of debugging has become important. It is difficult since there are many potential sources of errors including: bugs in the code, input data, software updates, and improper parameter settings. To address this problem, BugDoc[18] automatically infers the root causes and derive succinct explanations of failures for black-box pipelines using the results from previous runs. VersionClimber [29] is another automated system that deals with the problem of the pipelines that apply multiple packages, each of which evolves independently, to one or several data sources. VersionClimber automatically discovers newer versions of these packages that are compatible.

For the applications envisioned in LifeSWS, these systems will help, because we may want to combine different workflows (e.g., Galaxy, OpenAlea and Spark), debug them with a tool like BugDoc with some other data analytics services (e.g., time series analysis) and keep versions up-to-date. To integrate and execute heterogeneous workflows, we plan to rely on the Common Workflow Language (CWL) standard [8], which helps creating portable workflows.

Heterogeneous data management systems provide capabilities to access heterogeneous different data sources, which are important for our objectives. The problem of querying heterogeneous data sources, i.e., managed by different data management systems such as relational or XML database systems, has long been studied in the context of multidatabase systems [24]. However, multidatabase systems have not been designed for the cloud, with a large variety of data stores such as SQL, NoSQL, NewSQL and HDFS. Furthermore, operating in a cloud infrastructure provides more control over where the system components can be installed, which makes it possible to design more efficient architectures. These differences have motivated the design of polystores (or multistore systems) that provide integrated access to a number of cloud data stores. For instance, CloudMdsQL [17] supports a functional SQL-like language, capable of querying multiple heterogeneous data stores within a single query that may contain embedded invocations to each data store's native query interface.

Spurred by the growing use of machine learning in all kinds of applications, many new model lifecycle systems have been proposed. Different from traditional software engineering, the development of ML applications is more iterative and explorative, yielding a variety of artifacts, such as datasets, models, features, hyperparameters, metrics, software code and pipelines. The objective of these new systems is to enable explainability, reproducibility, and traceability of ML executions by supporting the storage, management and reuse of these artifacts. The systematic literature review of more than 60 ML lifecycle management systems [31] shows that there is no precise functional scope, thus making comparison be-

tween systems difficult. Some systems focus on the management of ML artifacts only while some others add capabilities for the development of ML applications. The most complete systems come from cloud providers, e.g., Microsoft Azure ML, Amazon SageMaker and Google Vertex AI, as ML as a service (MLaaS) platforms. In contrast, open-source systems tend to be more focused. For instance, MLflow [7] focuses on capturing, storing, managing, and deploying ML artifacts using a standard format to store models and project code. It provides APIs to access ML development tools, such as PyTorch, Scikit-learn and Tensorflow. Also motivated by the objective of providing a holistic view to support the lifecycle of scientific ML, ProvLake [33] is a provenance data management system capable of capturing, integrating, and querying data across multiple distributed services, programs, databases, stores, and computational workflows by leveraging provenance data.

Science platforms are facilities that provide services and resources for research communities to perform collaborative research, observation and experimentation. They may include major scientific equipment, sometimes HPC machines, scientific datasets, data and research papers, code libraries and models. A common particular case is the science gateway (or science portal), which is a community-developed set of tools, applications, and data that are integrated through a web-based portal or a suite of applications. Science platforms are more or less specialized for some particular science, e.g., InfraPhenoGrid, PHIS, Plntnet and CyVerse.

InfraPhenoGrid [28] is a grid-based platform to efficiently manage datasets produced by the PhenoArch plant phenomics platform in Montpellier and deploy scientific workflows using a middleware that hides complexity.

PHIS [22] is a rich Phenotyping Hybrid Information System complementary to InfraPhenoGrid designed for plant phenomics. It allows storing and managing heterogeneous data (e.g., images, spectra, growth curves) and multi-spatial and temporal scale data (leaf to canopy level) coming from multiple sources (field, greenhouse). Its ontology-driven architecture is a powerful tool for integrating and managing data from multiple experiments and platforms including field and greenhouse. PHIS allows to enrich datasets with knowledge and metadata enabling the reuse of data and meta-analyses. In contrast, LifeSWS addresses a wider spectrum of applications in life sciences and provides key services such as user-specific Web dashboards, model management, provenance and cache, and workflow integration.

Pl@ntNet [16] is a participatory platform and information system dedicated to the production and sharing of botanical data in order to study biodiversity. The main application performs deep learning-based plant identification on a smartphone and returns, given a plant image, the ranked list of the most likely species and asks for interactive validation by the users.

CyVerse [11] is a platform for life sciences with services and resources to deal with huge datasets and complex data analyses. It includes a Web-based platform with data management services (storage, analysis, visualization, exploration), shared data and science APIs to access supercomputing resources. CyVerse is the

closest to LifeSWS, but lacks key services such as user-specific Web dashboards, model management, provenance and cache, and Model/Workflow APIs.

7 Conclusion

In this paper, we proposed LifeSWS, an open service-based architecture that implements data analysis workflow services for life sciences. The main objective of LifeSWS is to support the construction and maintenance of high quality, scalable and efficient workflows by organizing and making workflow artifacts (datasets, models, metadata, workflow components, etc.) easy to search and manipulate using various workflow systems.

Our architecture capitalizes on the latest advances in web-oriented architectures, microservices and distributed and parallel data management. It relies on open source services and tools, and well-defined APIs to foster services interoperability (like cloud web services). LifeSWS provides three main layers of services (presentation and directory, workflow and data management) and APIs to interface with different workflow systems, data sources and data stores.

LifeSWS services can be implemented and deployed in various platforms to address the specific requirements of vertical applications. We illustrated a LifeSWS platform with Gypscie, which provides services to develop, share, improve and publish scientific artifacts (datasets, models, etc.).

We also illustrated our proposed architecture with real use cases from life science. These examples are in agro-ecosystems in the context of climate change and epidemic modeling. These examples share common requirements but have specific features that show different uses of LifeSWS.

LifeSWS capitalizes on our previous experience in developing major systems for scientific applications. However, there are still major issues. To understand the research issues, let us consider two important scenarios in which LifeSDS should be able to help: (1) domain scientists build one or more datasets which may be in a variety of formats (relational database, csv files, etc.); (2) they also build workflows that make use of these datasets. LifeSDS comes to play a role in two major ways: (1) improve the maintenance and performance of existing workflows; (2) allow authenticated and efficient access and management of multiple workflows and datasets.

Based on our experience and application use cases, the main issues we plan to work on involve:

- Making it easy to integrate and run heterogeneous workflows defined using the Common Workflow Language (CWL), while providing reuse and reproducibility;
- Providing efficient execution of heterogeneous workflows by caching intermediate results and performing cache-aware scheduling;
- Making it easy for domain scientists to manage the model life cycle, perform model selection and model integration for different types of models managed using different tools;

– Assisting scientists in analyzing diverse data types, such as time series, through the integration of advanced analytical methods for different analytical requirements such as clustering, anomaly detection, kNN search, etc;
– Keeping track of the provenance of both data sources and software components, both to aid in debugging using tools such as BugDoc[18] and to enhance the reproducibility of these computational experiments.

Acknowledgement

This work is within the context of the HPDaSc associated team between Inria and Brazil. Some of us are supported by CNPq research productivity fellowships. C. Pradal has support from the MaCS4Plants CIRAD network, initiated from the AGAP Institute and AMAP joint research units, and EU's Horizon 2020 research and innovation program (IPM Decisions project No. 817617, Breeding-Value project No. 101000747).

References

1. Afgan, E., et al.: The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. Nucleic Acids Research **50**(W1), 345–351 (2022)
2. Artzet, S., Chen, T.W., Chopard, J., Brichet, N., Mielewczik, M., Cohen-Boulakia, S., Cabrera-Bosquet, L., Tardieu, F., Fournier, C., Pradal, C.: Phenomenal: an automatic open source library for 3d shoot architecture reconstruction and analysis for image-based plant phenotyping. BioRxiv p. 805739 (2019)
3. Bondiombouy, C., Valduriez, P.: Query processing in multistore systems: an overview. International Journal of Cloud Computing **5**(4), 309–346 (2016)
4. Boursiac, Y., Pradal, C., Bauget, F., Lucas, M., Delivorias, S., Godin, C., Maurel, C.: Phenotyping and modeling of root hydraulic architecture reveal critical determinants of axial water transport. Plant Physiology **190**(2), 1289–1306 (2022)
5. Brito, A., Machado, L., Oidtman, R., Siconelli, M.J.L., Tran Minh, Q., Fauver, J., Carvalho, R.D., Dezordi, F., Pereira, M., Castro Jorge, L., Minto, E., Passos, L., Kalinich, C., Petrone, M., Allen, E., España, G., Huang, A., Cummings, D., Baele, G., Grubaugh, N.: Lying in wait: the resurgence of dengue virus after the zika epidemic in brazil. Nature Communications **12**, 2619 (05 2021)
6. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink: Stream and batch processing in a single engine. IEEE Data Engineering Bulletin **38**(4), 28–38 (2015)
7. Chen, A., et al.: Developments in MLflow: a system to accelerate the machine learning lifecycle. In: Workshop on Data Management for End-To-End Machine Learning (DEEM@SIGMOD). pp. 5:1–5:4 (2020)
8. Crusoe, M.R., et al.: Methods included: standardizing computational reuse and portability with the common workflow language. Communications of the ACM **65**(6), 54–63 (2022)
9. Daviet, B., Fernandez, R., Cabrera-Bosquet, L., Pradal, C., Fournier, C.: Phenotrack3d: an automatic high-throughput phenotyping pipeline to track maize organs over time. Plant Methods **18**(1), 130 (2022)

10. Fernandez, R., Crabos, A., Maillard, M., Nacry, P., Pradal, C.: High-throughput and automatic structural and developmental root phenotyping on arabidopsis seedlings. Plant Methods **18**(1), 1–19 (2022)
11. Goff, S., et al.: The iplant collaborative: Cyberinfrastructure for plant biology. Frontiers in Plant Science **2** (2011)
12. Guedes, T., Martins, L.B., Falci, M.L.F., Silva, V., Ocaña, K.A.C.S., Mattoso, M., Bedo, M.V.N., de Oliveira, D.: Capturing and analyzing provenance from spark-based scientific workflows with samba-rap. Future Generation Computing Systems **112**, 658–669 (2020)
13. Heidsieck, G., de Oliveira, D., Pacitti, E., Pradal, C., Tardieu, F., Valduriez, P.: Cache-aware scheduling of scientific workflows in a multisite cloud. Future Generation Computer Systems **122**, 172–186 (2021)
14. Hey, T., Tansley, S., Tolle, K., Gray, J.: The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research (October 2009)
15. Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs. ACM Comput. Surv. **54**(4) (jul 2021). https://doi.org/10.1145/3447772, https://doi.org/10.1145/3447772
16. Joly, A., Goëau, H., Bonnet, P., Bakić, V., Barbe, J., Selmi, S., Yahiaoui, I., Carré, J., Mouysset, E., Molino, J.F., Boujemaa, N., Barthélémy, D.: Interactive plant identification based on social image data. Ecological Informatics **23**, 22–34 (2014), special Issue on Multimedia in Ecology and Environment
17. Kolev, B., Bondiombouy, C., Valduriez, P., Jiménez-Peris, R., Pau, R., Pereira, J.: The cloudmdsql multistore system. In: ACM SIGMOD International Conference on Management of Data. pp. 2113–2116 (2016)
18. Lourenço, R., Freire, J., Simon, E., Weber, G., Shasha, D.E.: Bugdoc: iterative debugging and explanation of pipeline. VLDB Journal **32**(1), 75–101 (2023)
19. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M.B., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system. Concurrency and Computation: Practice & Experience **18**(10), 1039–1065 (2006)
20. Lustosa, H.L.S., da Silva, A.C., da Silva, D.N.R., Valduriez, P., Porto, F.A.M.: SAVIME: An Array DBMS for Simulation Analysis and ML Models Predictions. Journal of Information and Data Management **11**(3), 247–264 (2021)
21. Muller, B., Martre, P.: Plant and crop simulation models: powerful tools to link physiology, genetics, and phenomics. Journal of Experimental Botany **70**(9), 2339–2344 (2019)
22. Neveu, P., et al.: Dealing with multi-source and multi-scale information in plant phenomics: the ontology-driven phenotyping hybrid information system. New Phytologist **221**(1), 588–601 (2019)
23. Özsu, M.T.: Data science: A systematic treatment. Communications of the ACM **66**(7), 106–116 (2023)
24. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, Fourth Edition. Springer (2020)
25. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Annual Conference on Neural Information Processing Systems (NeurIPS). pp. 8024–8035 (2019)
26. Pedregosa, F., et al.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

27. Pereira, R.S., Souto, Y.M., da Silva, A.C., Zorilla, R., Tsan, B., Rusu, F., Ogasawara, E.S., Ziviani, A., Porto, F.: Djensemble: a cost-based selection and allocation of a disjoint ensemble of spatio-temporal models. In: International Conference on Scientific and Statistical Database Management (SSDBM). pp. 226–231 (2021)
28. Pradal, C., Artzet, S., Chopard, J., Dupuis, D., Fournier, C., Mielewczik, M., Negre, V., Neveu, P., Parigot, D., Valduriez, P., Cohen-Boulakia, S.: InfraPhenoGrid: A scientific workflow infrastructure for Plant Phenomics on the Grid. Future Generation Computer Systems **67**, 341–353 (2017)
29. Pradal, C., Cohen-Boulakia, S., Valduriez, P., Shasha, D.: VersionClimber: version upgrades without tears. IEEE Computing in Science and Engineering **21**(5), 87–93 (2019)
30. Pradal, C., Fournier, C., Valduriez, P., Boulakia, S.C.: Openalea: scientific workflows combining data analysis and simulation. In: International Conference on Scientific and Statistical Database Management (SSDBM). pp. 11:1–11:6 (2015)
31. Schlegel, M., Sattler, K.: Management of machine learning lifecycle artifacts: A survey. ACM SIGMOD Record **51**(4), 18–35 (2022)
32. Silva, V., de Oliveira, D., Valduriez, P., Mattoso, M.: DfAnalyzer: Runtime Dataflow Analysis of Scientific Applications using Provenance. Proceedings of the VLDB Endowment (PVLDB) **11**(12), 2082–2085 (2018)
33. Souza, R., Azevedo, L.G., Lourenço, V., Soares, E.F.S., Thiago, R., Brandão, R., Civitarese, D., Brazil, E.V., Moreno, M.F., Valduriez, P., Mattoso, M., Cerqueira, R., Netto, M.A.S.: Workflow provenance in the lifecycle of scientific machine learning. Concurrency and Computation: Practice & Experience **34**(14) (2022)
34. Tardieu, F., Cabrera-Bosquet, L., Pridmore, T., Bennett, M.: Plant phenomics, from sensors to knowledge. Current Biology **27**(15), R770–R783 (2017)
35. Valduriez, P., Porto, F.: Data and Machine Learning Model Management with Gypscie. In: CARLA Workshop on HPC and Data Sciences meet Scientific Computing. pp. 1–2 (2022)
36. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. In: USENIX Workshop on Hot Topics in Cloud Computing (HotCloud) (2010)
37. Zhang, C., Ma, Y.: Ensemble Machine Learning, Methods and Applications. Springer (2012)
38. Zorrilla, R., Ogasawara, E.S., Valduriez, P., Porto, F.: A data-driven model selection approach to spatio-temporal prediction. In: Brazilian Symposium on Databases (SBBD). pp. 1–12 (2022)