Species Distribution Modeling in the Cloud

Leonardo Candela*, Donatella Castelli, Gianpaolo Coro,
Pasquale Pagano, Fabio Sinibaldi

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Consiglio Nazionale delle Ricerche (CNR), Pisa, Italy

SUMMARY

Species distribution modeling is a process aiming at computationally predicting the distribution of species in geographic areas on the basis of environmental parameters including climate data. Such a quantitative approach has a lot of potentialities in many areas that include setting up conservation priorities, testing biogeographic hypotheses, assessing the impact of accelerated land use. In order to further promote the diffusion of such an approach it is fundamental to develop a flexible, comprehensive, and robust environment capable of enabling practitioners and communities of practice to produce species distribution models more efficiently. A promising way to build such an environment is offered by modern infrastructures promoting the sharing of resources, including hardware, software, data and services. This paper describes an approach to species distribution modeling based on a Hybrid Data Infrastructure that can offer a rich array of data and data management services by leveraging other infrastructures (including Cloud). It discusses the whole set of services needed to support the phases of such a complex process including access to occurrence records and environmental parameters and the processing of such information to predict the probability of a species' occurrence in given areas.

Received . . .

KEY WORDS: Species Distribution Modeling; Hybrid Data Infrastructure; Cloud Computing;

1. INTRODUCTION

Species distribution models† [1, 2, 3] aiming at estimating the presence of a species in a given area are essential instruments in the development of strategies and policies for the management and the sustainable and equitable use of living resources [4]. These models rely on available occurrence records to investigate the relationships between observed species presence and the underlying environmental parameters that – either directly or indirectly – determine a species distribution in a known area and use this information to predict the probability of a species occurrence in other areas [5].

Despite their importance in the development of informed conservation and management strategies, the generation of such models is nowadays still limited to few specific cases. There are two main factors that prevent the vast majority of researchers from exploiting such approaches on a large scale. First, the generation of these models requires very often large computing capabilities and appropriate modeling tools. These are often not available in the research centers where scientists operate. Second, meaningful models can only be developed when both a sufficient amount of

*Correspondence to: Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" (ISTI)
Consiglio Nazionale delle Ricerche (CNR)
Via G. Moruzzi, 1 – 56124, Pisa – Italy
E-mail: leonardo.candela@isti.cnr.it

†This is also known as bioclimate models, climate envelopes, ecological niche models, habitat models, resource selection functions, range maps, correlative models or spatial model.

good quality occurrence point datasets and suitable environmental datasets are available. A lot has been done in the last years to collect such datasets. However, they are still scattered in many different heterogeneous databases. This makes very difficult and time consuming to use them on a large scale in an integrated way. Recently, a large project, the Map of Life [6], has been launched in order to respond to the latter need. It aims at realising a global and web-based infrastructure for storing, sharing, producing, serving, annotating and improving diverse types of species distribution information. It responds to the demand for integrating diverse types of species distribution information thus mitigating the limitations they individually have, e.g., spatial or temporal grain, false positives or false negatives, global uniformity. Moreover, it highlights how this integration of disparate data types offers both new opportunities and new challenges for species distribution modeling.

The requirements behind the generation of predictive models are common to many other scientific areas where modern data-intensive science approaches are used [7]. In these contexts data come in all scales and shapes and innovative technological solutions are continuously introduced that promote "new ways of acquiring, storing, manipulating and transmitting vast data volumes, as well as stimulating new habits of communication and collaboration amongst scientists" [8].

Among the new technological solutions a primary role is played by *e-Infrastructures*, and in particular by Hybrid Data Infrastructure (HDI) [9]. These aim at supporting large-scale resource sharing – where resources range from hardware to data and software – and at providing scientists with *resource-as-a-service*[10, 11]. This last paradigm extends the notions of applications as services (a.k.a. "*SaaS*") and hardware and software systems as services (a.k.a. "*IaaS*" and "*PaaS*") as introduced in Cloud computing [12]. HDIs offer a rich array of data and data management services by leveraging other infrastructures (including Cloud). Their goal is to enable a data-management-capability delivery model where computing, storage, data and software are made available "as-a-Service" by the infrastructure. By means of these capacities, an HDI supports the creation and operation of *virtual research environments* [13, 14], i.e., web-based cooperation environments equipped with all the resources needed to accomplish a scientific investigation.

This paper introduce an Hybrid Based Infrastructure, D4Science [15], developed to serve the needs of biodiversity scientists. In particular, it discusses the capabilities this infrastructure offers for facilitating species distribution modeling. These consist both of large scale data processing facilities, obtained through the use of services compliant to the cloud paradigms, and of services supporting users in accessing environmental data spread among distributed data providers.

The rest of the paper is organized as follows. Section 2 describes the requirements that effective species distribution modeling processes pose and the potential benefits resulting from an HDI-based approach. It also presents the D4Science infrastructure by highlighting its cloud-oriented capabilities and the family of services that it offers to support the species distribution modeling. Related works are discussed in Section 3. Finally, conclusions and directions for future work are given in Section 4.

2. A HYBRID DATA INFRASTRUCTURE-BASED APPROACH FOR SPECIES DISTRIBUTION MODELING

As already introduced, species distribution modeling refers to the process of using computer algorithms to predict the distribution of species in geographic space on the basis of a mathematical representation of their known distribution in the environmental space. It is a complex and iterative process which include at least the following key steps [2]: (*i*) identification of *relevant data*; (*ii*) *modeling*, i.e., deciding how to deal with the correlated prediction variables, selecting the appropriate algorithm, training the model, assessing the model; and (*iii*) *mapping* predictions to geographic space.

A Hybrid Data Infrastructure can support such a complex process by offering both services oriented to simplify data discovery by users, when datasets are scattered among heterogeneous and distributed repositories, and services support the processing of such datasets for various purposes.

Table I. AquaMaps datasets characteristics

Table	Content	Size
HCAF	Environmental data (e.g., salinity, chlorophyl) on 0.5 degrees cells	259,200 records
HSPEN	Species tolerance wrt environmental parameters	11,549 records
HSPEC	Estimation of species occurrence by species and cell	135,439,014 records

In order to better illustrate the challenges involved in implementing such kind of workflows and the needs of the scientists in performing them, a description of a typical and large-scale modeling scenario in the marine biology domain is given in the next section (cf. Sec. 2.1). This description is followed by a presentation of the basic facilities characterising the D4Science Hybrid Data Infrastructure as a modern computing platform is discussed (cf. Sec 2.2). In addition to the basic facilities, D4Science provides practitioners involved in species distribution modeling with (*a*) facilities specifically conceived to support environmental data discovery and access when data are federated from a number of data providers (cf. Sec 2.3) and (*b*) facilities supporting a cloud oriented computation of species distribution modeling (cf. Sec. 2.4). Finally, a detailed description of how such facilities have been exploited to support species distribution modeling phases and the resulting benefits are discussed (cf. Sec. 2.5).

2.1. The AquaMaps Scenario

Species distribution can be predicted using a range of different approaches and tools. All these approaches are based on the usage of occurrence records. Some of them use simple environmental envelope models that require only information about where a species has been observed. Others use more complex models that exploit information about areas where species are absent [5]. Species distributions are often represented through geographical maps but mapping large-scale distributions is an issue. Occurrence data are often fragmented and potentially non-representative. The solution requires an enrichment of occurrence records with independent knowledge about species distributions and habitat usage. One of the most successful approaches in this direction is AquaMaps.

AquaMaps [16, 17] are computer-generated geographical maps that show large-scale predictions of where a marine species may occur. These maps are generated using known occurrences of a species and such a species preferences with respect to environmental properties such as depth, salinity, temperature, primary production and its association to land and sea ice. AquaMaps offers scientists an interactive and powerful access to a large volume of data and provides them with both a high level view of the environment and the ability to filter on areas, species and environmental features.

The environmental features are obtained from a dataset named *Half-degree Cell Authority File* (HCAF), which defines both static physical properties – e.g., sea depth, tides, land distribution – and dynamic properties – e.g., temperature, ice concentration, primary production.

The high resolution of this dataset (0.5 degree cells, about 50x35 km), the large number of considered species (tens of thousands) and the variability of model parameters call for a computationally and data intensive task, whose requirements cannot be met by standalone solutions. The first step towards generating species distributions is to model how much species are compatible with life supporting parameters, by taking into account the environmental conditions in the locations where the species has been observed. These observations are used to create a *Half-Degree Species Environmental Envelope* (HSPEN) table, which, for each species, contains ranges of suitable and preferred values for each of the environmental properties.

The *Half-Degree Cells Species Assignments* (HSPEC) table is generated for each species at each location where environmental data are available by using the HSPEN table and the environmental data contained in the HCAF table. Hence, the HSPEC table is a Cartesian product of the HSPEN and HCAF tables.

A characterization of these three datasets is given in Table I.

The production of AquaMaps is (*i*) based on a sequential algorithm developed by the biologists, (*ii*) intended to be performed through a web interface, and (*iii*) able to produce the potential (*Aquamaps Suitable*) and the actual (*Aquamaps Native*) geographic distributions for 11,549 marine species. Such number refers to the species for which there are sufficient information to produce maps. However, the information about species occurrence points and their related environmental characteristics are continuously added and contribute quarterly to the update of the HSPEN table. The overall goal is to be able to produce the species distribution for all species whose biological data are maintained and made available by the Fishbase [18] organization.

At present, the complete process takes about 48 hours, resulting in the production of 11,549 maps stored on a single GeoServer [19] instance.

2.1.1. Enhancing the AquaMaps Procedure The species distribution modeling approach discussed in this paper is intended to shorten the time length of producing AquaMaps by relying on a cloud processing paradigm. In particular, the developed approach parallelises the processing performed by training and projection algorithms by distributing the computation on a cloud processing platform. The parallelisation degree depends on which resources are available to the infrastructure at runtime and on the parallelizable nature of the algorithm to execute. The most simple parallelisation happens in the cases where a set of species $S = \{s_1, s_2, \ldots, s_m\}$ has to be projected on a certain area A, at a fixed resolution and according to a predefined set of features. The procedure acts as follows:

1. the area A is divided into a set of c-squares [20] C according to the wanted resolution. In this case, we use c-squares as a notation for compactly encoding latitude and longitude coordinates along with resolution information;
2. for each c-square c in C the set of values $\{f_1, f_2, \ldots, f_n\}_c$ for the predefined features is extracted by intersecting all the feature layers, hosted on the geo-spatial architecture described in Section 2.3, with the c-square areas;
3. for each species s in S, a request for projection on C is submitted to the D4Science Cloud Computing platform;
4. at the end of the procedure, a table is produced containing a set of values representing the projection probability $\{P(s, c)\}$ for each species in each c-square according to the selected algorithm and features.

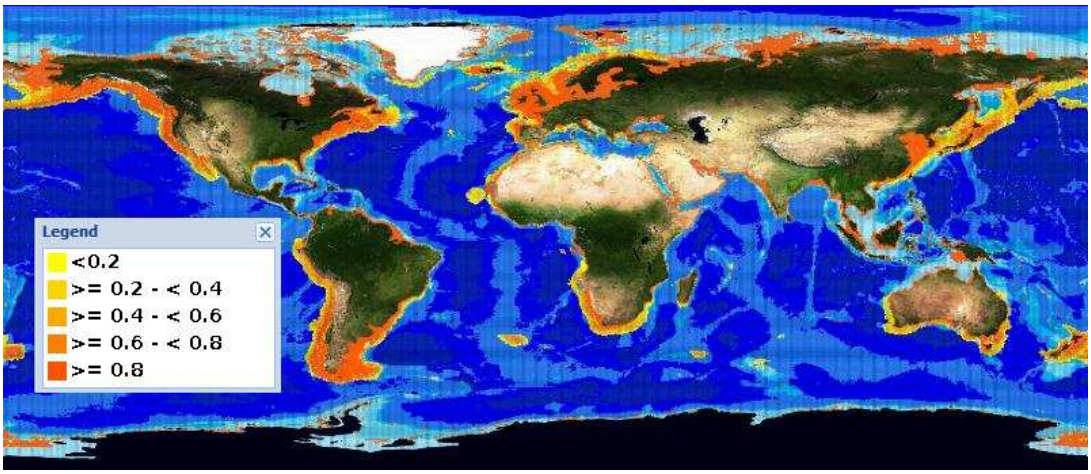

Figure 1. The Aquamaps Suitable distribution for the Basking Shark species.

The details of the cloud computing processing platforms are described in Section 2.4.

2.1.2. AquaMaps Examples Figure 1 shows a potential distribution map for the Basking Shark species (*Cetorhinus maximus*) produced by means of the *Aquamaps Suitable* algorithm, which is

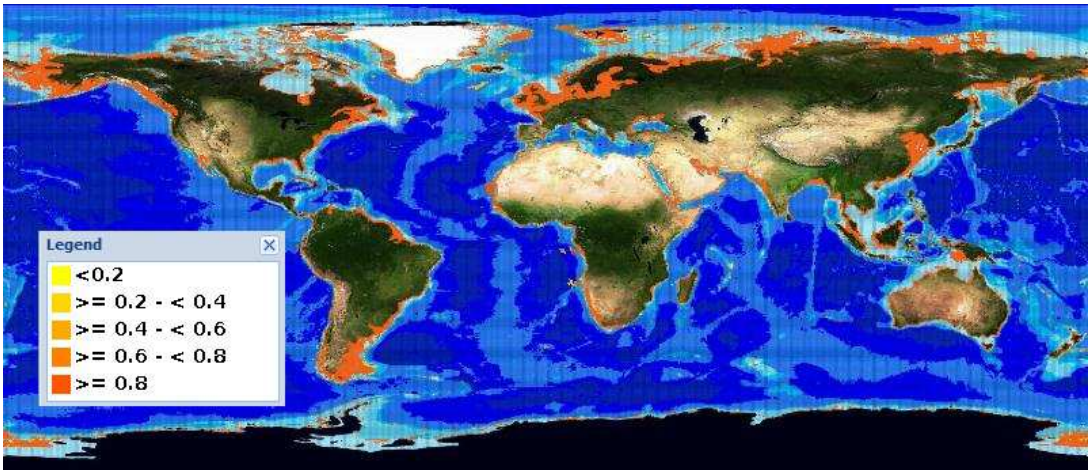

Figure 2. The Neural Network distribution for the Basking Shark species.

a knowledge base driven procedure, while Figure 2 reports a map produced for the same species by a Neural Network. In both cases the resolution was 0.5 degrees and the set of features, for each c-square, was made up of 10 values belonging to the following list:

(*i*) mean depth (in m);
(*ii*) maximum depth (in m);
(*iii*) minimum depth (in m);
(*iv*) mean annual sea surface temperature (in $^{\circ}C$);
(*v*) mean annual sea bottom temperature (in $^{\circ}C$);
(*vi*) mean annual salinity at surface (in psu);
(*vii*) mean annual primary production ($mgCm^{-2}day^{-1}$);
(*viii*) mean annual ice concentration (in percentage);
(*ix*) mean distance from land (in m);
(*x*) ocean area contained in the cell (in m^2).

2.2. *Opportunities for Enhancement: D4Science Infrastructure Services for Computing*

As previously discussed, the concept of *infrastructure* supporting eScience has been recently used to describe a tool whose primary goal is to provide scientists with the resources they need, while reducing the quantity of efforts they have to spend in activities different from the pure scientific activity [10, 11]. This tool is a vehicle aiming at promoting the sharing and cross-fertilization across scientific disciplines and communities. At the same time, it reduces the overall costs by realising an *economy of scale*, e.g., the same resource is deployed once and exploited to serve many domains.

D4Science is among the first concrete realizations of the Hybrid Data Infrastructure concept. Its enabling technology, *gCube*[‡] [21], is a software system designed to facilitate research collaborations that span institutions, disciplines, and countries. Its primary goal is to enable and simplify the creation and operation of an infrastructure capable of federating resources from other infrastructures and to provide *virtual research environments* (VRE). In a VRE users come together to analyse, curate, and publish all the outputs of research life-cycles.

gCube abstracts over a variety of technologies related to data, processing and resource management on top of Cloud enabled middleware. It exposes such technologies through a comprehensive and homogeneous set of APIs and services. gCube aims to offer solutions to abstract over differences in location, protocols, and models by scaling no less than the interfaced resources,

[‡]http://www.gcube-system.org/

by keeping failures partial and temporary, and by automatically reacting to and recovering from a large number of potential issues. It doesn't hide infrastructures middleware and technologies. Rather it turns infrastructures and technologies into a utility by offering a single submission, monitoring, discovery, and access facility. It offers a common framework to programming in the large and in the small. It allows exploiting concurrently private virtualized resources organized in sites with resources provided by IaaS and PaaS cloud providers.

gCube is an open-source framework developed in the last 8 years by an international consortium of IT companies and research institutions. It is composed by more than 350 software packages that are certified and distributed under the same EUPL license [22]. It encompasses a large spectrum of functionality related to (*i*) computational and storage resources registration, monitoring, discovery, and access; (*ii*) data registration, harmonization, curation, discovery, and access; and (*iii*) processes registration, discovery, and execution. It relies on a rich and open array of mediator services for interfacing with *Grid*, e.g., the European Grid Infrastructure, *commercial cloud*, e.g., Windows Azure and Amazon EC2, and *private cloud*, e.g., based on OpenNebula, *infrastructures*. Relational databases, geospatial storage systems, e.g., GeoServer, nosql databases, e.g., Cassandra and MongoDB, and reliable distributed computing platform, e.g., Hadoop, can all be exploited as infrastructural resources.

Among its services (which are systems on its own since they consists of multiple web services and software libraries), the following ones are key to configure an infrastructure such as D4Science as a modern computing platform. In particular, the Information System (IS) (cf. Sec. 2.2.1) plays the role of registry for the whole infrastructure. The Resource Management (cf. Sec. 2.2.2) orchestrates the entire set of resources partaking to the infrastructure by taking care of their monitoring and dynamic (un-)deployment. The Distributed Storage (cf. Sec. 2.2.3) provides for a data storage infrastructure supporting a rich array of objects. The Messages Queue (cf. Sec. 2.2.4), Executor (cf. Sec. 2.2.5), and Transformation Service (cf. Sec. 2.2.6) realise an environment for executing computing tasks by relying on diverse computing nodes. The execution of a task is coordinated by relying on a messaging system. Moreover, a complex task might benefit from a number of data transformations tasks that reconcile the data produced by a computing node with the expectations of the consumers nodes in the task workflow. Finally, the VRE Management Service (cf. Sec. 2.2.7) complements these services by offering facilities for the creation and operation of web-based cooperation environments.

2.2.1. Information System This service is a key one in a gCube-based infrastructure since it offers functionalities for publishing, monitoring, discovering and accessing the set of resources forming the infrastructure. Computational and storage resources, data, services, and applications are all described and modelled as resources. For each profile defining a resource typology, an *application profile* [23] has been defined by exploiting metadata from several element sets, e.g., Glue Schema [24] to model Grid entities and WSDL [25] to model network services as a set of endpoints operating on messages. The Information System, distributed on a number of services, acts as a single registry where all the resources are registered. Thus, every service partaking in the infrastructure must refer to it to dynamically discover the other infrastructure constituents. Moreover, the approach provided by the IS is of great support for the dynamic deployment capabilities of gCube.

2.2.2. Resource Management This service is responsible for seamless access to and management of shared, distributed and heterogeneous resources. It ensures resource management capabilities by offering resource deployment, configuration, staging, scoping, monitoring and secure operation of services that become fully dynamic and a responsibility of the infrastructure. In particular, it manages the entire lifecycle of services, engaging in autonomous interactions with the infrastructure and local environment, and allowing customisation of deployment, initialization, activation and failure response; it enforces the policies and security rules associated with shared resources; it implements, on behalf of the services, publication, access, and notification of change to service state governing transparently the service lifetime, and managing its persistence on different storage media, including its recovery from remote media upon service migrations; it standardises the use of

systemic faults within service interfaces and implementations, transparently supporting retry-same and retry-equivalent semantics and converting faults into equivalent lighter-weight exceptions at service boundaries.

2.2.3. Distributed Storage This service provides access to different storage back-ends guaranteeing an appropriate portfolio to the applications deployed in the infrastructure. Its different storage layers offer facilities for handling: (*a*) multi-versioned software packages and dependencies resolution among them; (*b*) large scientific data-sets accessible as tables; (*c*) Time Series offered through an OLAP interface; (*d*) structured objects storable as trees of correlated information objects; (*e*) geo-coded datasets compliant with OGC-related standards; and, finally, (*f*) plain files.

In particular, the management of files is based on a network of distributed storage nodes managed via specialized open-source software for document-oriented databases. This management is offered by the gCube Storage Manager, a Java-based software that presents a unique set of methods to be used by the services and applications running on the e-Infrastructure. In its current implementation, two possible document store softwares are used [26], MongoDB and Terrastore. It is able to upload files on different distributed storage systems.

The Storage Manager has been designed to reduce the time required to add a new storage system to the e-Infrastructure. This promotes openness versus other document stores, e.g., CouchDB [27], while hiding the heterogenous protocols offered by those systems to the services and applications exploiting the e-Infrastructure storage capabilities.

This ensures sustainability while preserving the reliability of the proposed solution since each node hosts parts of a file and these parts are replicated on different nodes.

The Storage Manager provides the callers with URIs embedding information about a storage system and a file location. This is a fundamental feature since each document store has its own way to retrieve the contents of the files and not always a single URL for accessing a file is provided. Then the Storage Manager resolver is responsible for transforming a URI into a download process that uses the suited approach for the storage system hosting the file. In order to implement such process a Java URI Connection, based on a protocol named SMP, was implemented. This is a proprietary protocol that abstracts over the storage system hosting the file. Being a proprietary protocol, it contributes to the safeguard of the files hosted on internal storages from external access, by allowing retrieval only through an e-Infrastructure authorized service. However, with the increase acceptance and adoption of the CDMI international standard [28] endorsed by SNIA, the Storage Manager is evolving towards the support of such a standard interface. Transport security and security capabilities, which are mandatory to implement, will not be the only security measures supported. Rather, the Storage Manager will extend them to include additional, and optional according to the CDMI standard, capabilities such as: (*i*) user and entity authentication; (*ii*) authorization and access controls; and (*iii*) data at-rest encryption.

Looking at the end user's perspective, each participant has a dedicated *Workspace*. All the Workspaces are connected to the distributed storage system, thus being the main data sharing tool in a gCube-empowered e-infrastructure such as D4Science. A user, once finished a processing phase, can choose to save a dataset, e.g., a csv file, and share it with other e-Infrastructure members. Temporary and persisted shared folder are both supported as well.

2.2.4. Message Queue This service is actually based on an instance of Apache Active Message Queue [29] to support a queue-based mechanism for distributing messages to consumers. Diverse queues can be instantiated by services and applications in the e-Infrastructure as to enable the building of proper communication channels. It provides an asynchronous communication mechanism where the sender and receiver of a message do not need to interact with the queue at the same time. Messages placed onto the queue remain stored until a consumer retrieves them. Direct message queues are used in several gCube cloud processing tasks, where a producer sends requests to a specific queue, and every consumer which consumes a message acts according to its content. Each service and application registered in the e-Infrastructure can act as producers on a message queue when need to send periodic status messages to a consumer that controls the processing status.

In the rest of this paper we will use the term *Worker node* to refer to an e-Infrastructure resource that takes part to a distributed computation by consuming messages from one assigned queue.

2.2.5. Executor This service is a key component to endow a gCube-empowered infrastructure with cloud processing. It acts as a container for gCube tasks, which are functionally unconstrained bodies of code that lack a network interface but can be dynamically deployed into a gCube service and executed through its interface. In particular, gCube tasks are designed, packaged, and deployed as plug-ins of the Executor component. An Executor plug-in is basically a piece of code that resides on a gCube service instance and can be invoked at any time. An instance of the Executor on a gCube node publishes descriptive information about the co-deployed tasks and can inform clients about the status of their execution. Clients may interact with the Executor tasks through a library offering high-level facilities which simplifies discovering of service instances by the IS and the execution of tasks available in those instances.

The *Generic Worker* plug-in is a task of the Executor which is exploited in cloud computation tasks. It is able to execute "processes", either binary executables or scripts, along with their dependencies in a sandbox. The Generic Worker: (*i*) creates the sandbox; (*ii*) provides a tightly controlled scratch space on disk and memory; (*iii*) prevent the process from inspecting the host system and read from input devices; and (*iv*) configures the environment to run the requested process.

The Generic Worker is written in Java language conforming to the standard interfaces of the gCube framework for the Executor plug-ins, but it executes code which is treated as a standalone external software. It is connected to an Active Message Queue instance for creating producers and consumers. A Generic Worker accepts as input a message from the IS notification system, which contains (*i*) the Active Message Queue IP address and encrypted credentials for accessing to it, (*ii*) the queue name from which messages have to be gathered, (*iii*) the queue name for sending status message, and (*iv*) a flag which states if the interaction with the queue has to be stopped.

2.2.6. Transformation Engine This service offers data transformation facilities, thus making it possible to produce data in "formats" different from their initial one. The service is manifestation and transformation agnostic by offering an intelligent, object-driven operation workflow. Each transformation-program is registered in the transformers registry and then used at run-time to process large (in batch) and small (in real-time) transformation scenarios. The Transformation Engine exploits the gCube Process Execution engine that manages the execution of the transformation-programs in a distributed e-Infrastructure. The execution is coordinated by a composite plan that defines the data dependencies among its actors. Such plan is based on a powerful, flow-oriented processing model that supports several computational middlewares without performance compromises. Thus, a transformation task can be designed as a workflow of invocation of code components, the transformation programs, assuming that prerequisite data are prepared and delivered to their consumers through the control of the flow of data.

2.2.7. Virtual Research Environment Management This service supports the definition, deployment, monitoring, and operation of virtual research environments [13]. Through VREs, groups of users have controlled access to selected, and optionally temporarily dedicated, data, services, storage, and computational resources integrated under a personalised web-based interface. A VRE facilitates cooperative activities such as: metadata cleaning, enrichment and transformation by exploiting mapping schema, controlled vocabularies, thesauri and ontologies; processes refinement and show cases implementation (restricted to a set of users); data assessment and processing; expert user validation of products generated through data elaboration or simulation; sharing of data and processes with other users. In essence, this facility for dynamically creating and maintaining VREs is a distinguishing feature of the D4Science infrastructure that simplifies the exploitation of the rest of the facilities. In particular, each VRE provides its users with a ready-to-use, comprehensive and integrated web-based environment catering for the scientific tasks it has been designed for.

Other key systems belonging to the gCube framework and exploited by the D4Science HDI are presented in the following sections. In particular, we focus on (*a*) the set of facilities and services specifically conceived to support the species distribution modeling, namely those facilities giving access to environmental data (cf. Sec. 2.3) and those supporting the distributed processing of such data (cf. Sec. 2.4); and (*b*) how such facilities are deployed and which benefits are resulting when the above services are exploited in a cloud-oriented setting (cf. Sec. 2.5).

2.3. Environmental Data Discovery and Access Facilities

An infrastructure aiming at properly supporting Species Distribution Modeling has to manage environmental data that are generally offered by a large number of different data providers.

D4Science offers a service for discovering and accessing to distributed environmental data and maps having the architecture and constituents depicted in Figure 3. This service relies on maps stored on several GeoServer [19] instances. A set of PostGIS 1.5 [30] databases store the concrete values and geometries and the GeoServer distributes them according to standard Open Geospatial Consortium (OGC)[§] protocols like Web Map Service (WMS), Web Coverage Service (WCS) and Web Feature Service (WFS). A GeoNetwork[¶] instance indexes layers according to their content, name, title and URL of the GeoServer which stores them. GeoNetwork provides a Lucene [31] based indexing system which stores metadata and, by means of specific connectors, can parse the protocols adopted by common geo-spatial storage systems (e.g., GeoServer, Thredds [32]). Simple metadata forms compilation is moreover possible for indexing non-geographical data.

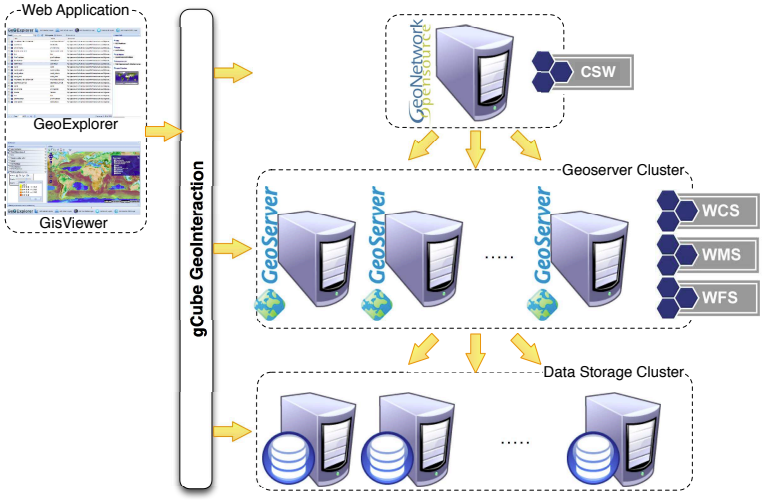

Figure 3. The distributed geo-spatial architecture in D4Science. It consists of (*a*) a GeoNetwork instance federating a number of GeoServer services and offering a unifying catalogue of the items these services contain; (*b*) a number of GeoServer instances offering geospatial items. GeoServer instances can be either natively deployed on D4Science infrastructure resources or pre-existing (e.g., the FAO GeoServer); (*c*) a number of storage servers supporting data disseminated via GeoServer. These storage servers can be used by many GeoServer instances.

GeoNetwork is endowed with an OGC CSW [33] based search engine which allows for retrieving meta-information. For the GeoServer GIS layers, metadata are registered by the clients that create new layers or are automatically read from a GeoServer in the case of static layers. The URL indication of the GeoServer instance which stores the information is mapped on the URL field in the GeoNetwork's default metadata set. A client can use a CSW request via HTTP protocol in

[§]http://www.opengeospatial.org/
[¶]http://geonetwork-opensource.org/

order to retrieve, for a certain layer, the URL of the GeoServer on which this is stored. From that moment on, the client can interact directly with that GeoServer. A set of methods for interacting with the above architecture is available in gCube and is offered also through a web application.

The web application is partitioned into two logical components, namely the GeoExplorer (cf. Fig. 4) and the GisViewer (cf. Fig. 5). The former offers an analytic view of all the layers indexed on the GeoNetwork. It consists of two panels (cf. Fig. 4), i.e., a tabular paging view of the layers, and a "layer details" contextual panel. The latter exposes a summary of layers' metadata, like the layer title, name and the URL of the GeoServer that stores it.

The GeoExplorer portlet (cf. Fig. 4) is a web application that allows users to interactively navigate, organize, analyze, search and discover GIS layers that are either internal to the D4science infrastructure or hosted by external services. It interacts with the above described GeoNetwork based architecture in order to discover layers residing in a gCube-based distributed e-Infrastructure.

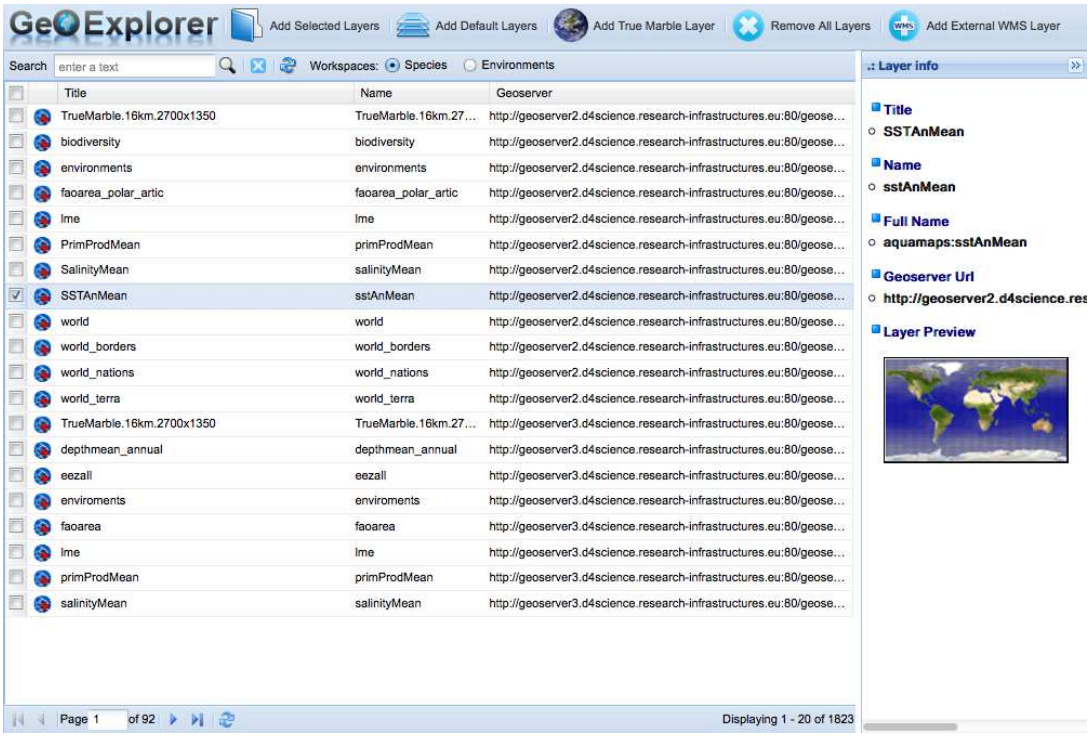

Figure 4. The GeoExplorer portlet interface.

The GeoExplorer portlet provides the following facilities:

1. layers exploration (preview, sorting, filtering, multi-selection);
2. visual correlation analysis of several overlying layers;
3. CQL visual filter of one or more layers;
4. tabular spatial data retrieval of one or more layers;
5. transect charts production;
6. a rich set of base layers;
7. external WMS layers visualization.

The GisViewer (cf. Fig. 5) is a widget based on a map container for visualizing the GIS layers chosen by means of the GeoExplorer. A graphical map-based representation is crucial for enabling users to analyze geo-spatial data and to retrieve auxiliary information.

The main functionalities of the GIS Viewer, which can be sparsely found in other tools for GIS layers visualization, can be summarized as:

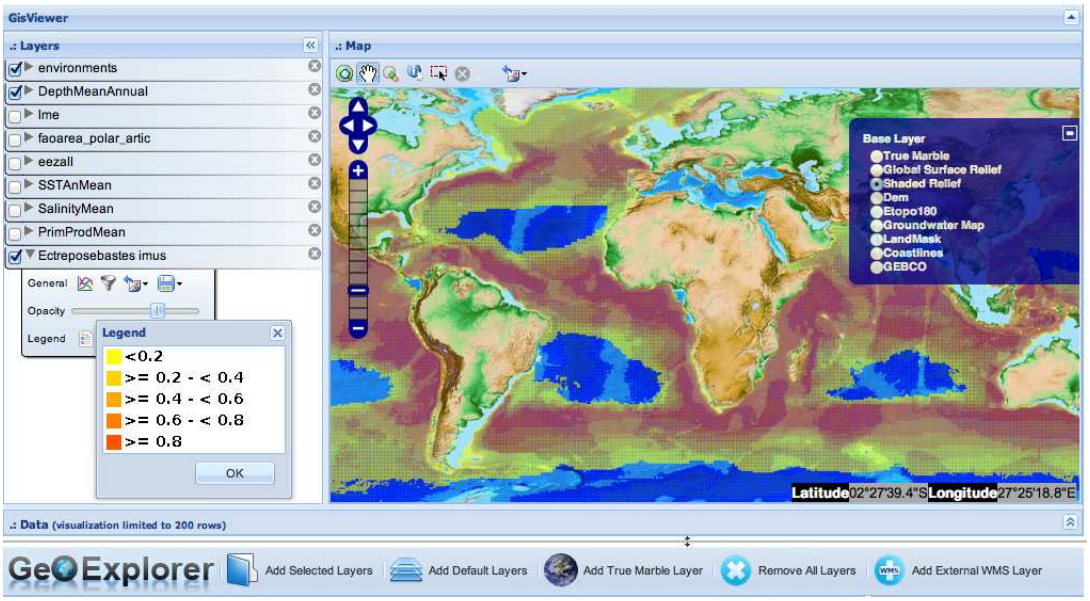

Figure 5. The GisViewer widget interface.

1. use of the "overlay" concept to enable users to create a personal view on maps;
2. dynamic sortable list of layers, where the sorting establishes how the layers are overlaid and how they are shown on the map;
3. layers opacity dynamic setting;
4. alternative layers style selection;
5. CQL filters application;
6. layer snapshot saving onto the local file system or on the distributed D4Science storage system;
7. transect function production and display, along a line traced on the map;
8. complete data retrieval and display for a certain zone or point manually selected on the map;
9. selection of less or more accurate background maps.

2.4. Distributed Data Mining Facilities

The idea under the distributed cloud computation for training and projecting species distributions algorithms is to overcome common limitations that may be encountered when using such statistical procedures. Possible limitations are (*i*) the training and projection procedure time, (*ii*) the linear or non-linear time increase when the number of species or the input resolution increases, (*iii*) the multiple runs needed for reducing *overfitting* or *local minima* problems, (*iv*) the multiple model topologies to be evaluated for assessing the optimal model's configuration. All these issues are due to the demanding computational requirements of the experiments. A scientist usually needs to evaluate the results and to combine the outcomes of different experiments. If the training session of a model requires too much time, then a complete experiment can take a very long period. Some experiments have prefixed deadlines or scientists could require results in short time. This is especially the case of disease transmission modeling or of the evaluation of the consequences of natural disasters on species presence. The production of the optimal model could go over the prefixed time of the experiment. Such contingency can imply to accept sub-optimal models or to accelerate the qualitative evaluation phase of the results, which is a fundamental step of the experiment. In many cases the number of experiments performed by the scientists is reduced, which can result in incomplete models. Furthermore, when using a local desktop tool for species distribution modeling, the data to be used as inputs have to be collected and prepared for the algorithm to apply.

The main advantages in using a distributed e-infrastructure endowed with a cloud computation procedure can be resumed by looking at the features provided by D4Science:

1. efficiency and time saving in computations;
2. availability of a set of data sources containing environmental features;
3. quality and reliability of the features;
4. certification of compliancy between the data hosted by the infrastructure and the algorithm inputs\outputs;
5. import of users' own files;
6. sharing of results and users' files.

For species distributions modeling applications, gCube provides software for building (*i*) data sharing mechanisms, (*ii*) import facilities for uploading files into multi-tenant workspaces, (*iii*) data assessment procedures for helping users to certify their datasets, (*iv*) development tools for new algorithms according to standard interfaces.

In particular, the Statistical Manager (SM) has been specifically conceived to offer data mining operations (species distribution modeling falls in this category) by benefitting from a distributed computing infrastructure like D4Science.

2.4.1. The Statistical Manager This is a cross-usage service that provides users and services with tools for performing data mining operations. Specifically, it offers a unique access to perform data mining and statistical operations on heterogeneous data, which may reside either on the client side, in the form of csv files, or be remotely hosted, possibly in a database. The Statical Manager (SM) service is able to take inputs and execute the operation requested by a client by invoking the most suited computational facility from a set of available computational resources. Executions can run either on multi-core machines or on different computational platforms, like D4Science, Windows Azure and other different private and commercial Cloud providers.

The SM Service is a container of algorithms which are implemented as plug-ins based on the Dependency Injection programming pattern [34]. In this paper we will focus on species distributions modeling algorithms. These reside on SM and can be invoked by infrastructural or external clients according to a public WSDL interface. The requests are managed asynchronously and the client can monitor the status of the computation at any time.

Each species distribution modeling procedure is made up of two parts: a *computation manager* (Generator) and an *algorithm core*.

The algorithm core is the set of procedures that calculate the probability that a species can live in a specific area, according to certain environmental conditions. It provides methods to retrieve species and area information. For example it can provide database queries to select a range of species or *c-squares*.

The Generator is a procedure responsible for (*i*) communicating with a computational platform (e.g., D4Science, Windows Azure) and controlling its available resources, (*ii*) monitoring the computation, and (*iii*) distributing the probability calculations, provided by the algorithm core, on a computational platform. Many algorithm cores can share the same Generator, as many adopt a simple map-reduce approach [35] on the species names dimension. For example, the Aquamaps and Neural Network projection algorithms for marine species share the same Generator. The Generator associates the processing of a pair *(species, c-squares)* to each atomic step of the computation. In other cases the Generator is more specific to the algorithm, especially when the parallelisation is dependent on the core part of the algorithm.

When a client asks for a computation, it sends a request to the SM service for executing an algorithm with proper inputs. An internal procedure of the SM service (*SM Selector*) is responsible for choosing the most suitable Generator for the selected algorithm. The choice of the Generator depends even on which computational resources are available at the moment. For example, if no computational resources (i.e., no Worker nodes) are available in D4Science, then the computation is outsourced to external providers like Windows Azure or to a dedicated single multi-core machine. In such case, the SM Selector uses a different Generator from the previous one. Figure 6 shows a schema of such selection mechanism.

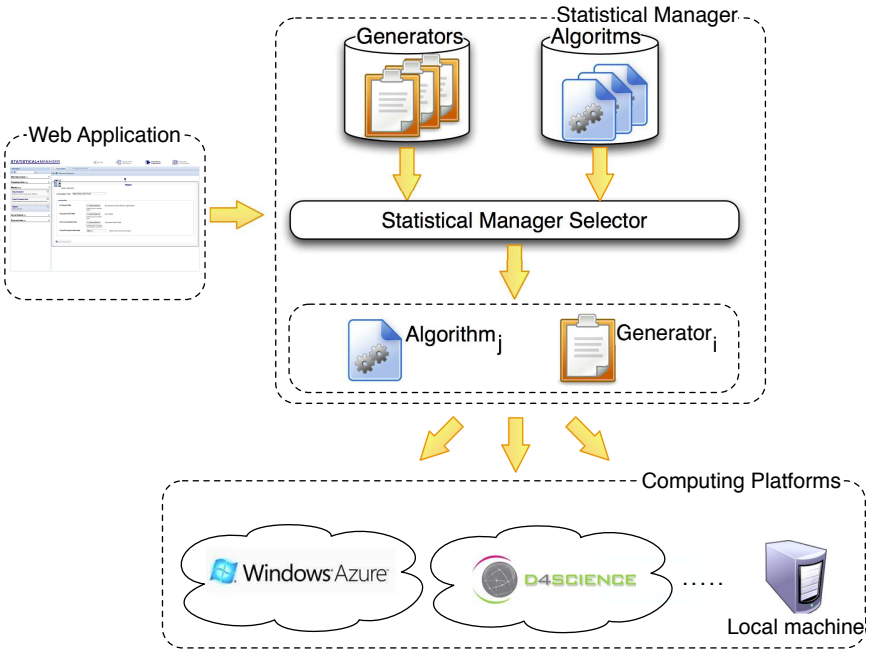

Figure 6. The Statistical Manager Architecture.

The Statistical Manager as well as Generators and algorithms are developed in Java. Generators and algorithms are plug-ins of the Statistical Manager. This means that adding a new Generator or algorithm does not require to modify the SM. Furthermore they can be developed separately. Indeed, we built up a framework which allows to easily create new Generators and algorithms by extending well-documented general interfaces. Thus, a developer with knowledge in the Biodiversity domain can extend the SM with specialized algorithms, while an IT-specialist can contribute with new Generators for new classes of algorithms.

The implementation of a new algorithm does not imply the creation of a new Generator. An algorithm suited to parallel processing is compliant with an interface called *DistributionAlgorithm*. According to this interface, the algorithm declares the kind of processing it performs (ALGORITHM_PROP), taken from a finite number of possibilities. Species modeling algorithms, for example, include those that (*i*) calculate species distributions on geo-referenced areas (GEOAREA_VS_SPECIES), (*ii*) consider only the environmental features vector space (ENVIRONMENTAL_FEATURES_VS_SPECIES) or (*iii*) calculate correlation between some phenomenon and species presence (PHENOMENON_VS_SPECIES). The ALGORITHM_PROP allows us to get a complete overview of the kinds of calculations the SM can perform. On the other side, this programming limitation gives us more control on the algorithms that can be plugged to the SM by external developers. The indication on the ALGORITHM_PROP is used by the *SM Selector* process, which identifies the best Generator for that algorithm. An algorithm only declares how to take sets of information objects from a source and how to produce a probability. These objects contain information to process, for example they could be related to species or to environmental features. They can be stored either on local files, or on a database, or on platform dependent storage systems. For example, Aquamaps Suitable implements a DistributionAlgorithm with an ALGORITHM_PROP of kind GEOAREA_VS_SPECIES. It relies on a database. By implementing the DistributionAlgorithm interface, it indicates how to retrieve a range of species and geo-referenced environmental features from the database. Clearly, it also implements a method to produce a probability for one species record in a geographical space.

On the other side, the Generators class conform to the *Generator* interface. Such interface is developed relying only on the DistributionAlgorithm interface, without identifying a

specific implementation. Thanks to the Dependency Injection pattern, the compliancy with the DistributionAlgorithm interface allows the real algorithm to be instantiated at runtime. Furthermore, a Generator has to declare the list of ALGORITHM_PROPs it supports and the computational platforms it can exploit. The implementation of a Generator is (*i*) independent on the specific algorithm implementation, (*ii*) may run multiple kinds of algorithms, and (*iii*) depends on the computational platform. For example, a Generator implementing a distributed computation for GEOAREA_VS_SPECIES algorithms, will own the logic for dividing the area and species dimensions in several intervals. Furthermore, it will distribute the probabilities calculations for the intervals on several machines. Every available computational platform is registered on the D4Science Information System along with its connection parameters. The list of platform is maintained by an administrator of the infrastructure. The Statistical Manager is responsible for discovering the platforms list and to give a Generator the access to the platform it requires.

Each platform is statically ranked according to the number of resources it can potentially offer and to the degree of parallelization it supplies. According to the current setup, local processing has the lowest rank, while D4Science has the highest. The others fall in the middle, according to the number of resources they offer to the e-Infrastructure. When a request for an algorithm execution is sent to the SM, this runs the *SM Selector* process which (*i*) finds the algorithm, (*ii*) takes its ALGORITHM_PROP and (*iii*) finds a list of Generators which support the algorithm. Once a ranked list of Generators has been produced, the SM Selector takes the best Generator. Then the SM asks to the Generator to check for the availability of the platform resources. We chose to put this check among the Generator capabilities because the Generator owns the logic to interact with the platform. If no resources are available at that moment, the next Generator is chosen. Otherwise, the Generator is executed and the algorithm is indicated as the DistributionAlgorithm to use. If all the resources are occupied on all the platforms, the request is queued.

2.5. Species Distribution Modeling with a Cloud Computing Approach

On the basis of the services and technologies discussed in the previous sections, we can describe the cloud computing approach exploiting the computational resources of D4Science.

We will take the Aquamaps Suitable algorithm as an example. Let's suppose a client asks for the projection of a set of marine species on the world oceans. The Statistical Manager is in charge of satisfying the request and selects a Generator and a computational platform which is suited for the Aquamaps algorithm: D4Science is one of these platforms. As the computation of a distribution for a single species cannot be parallelized, the distribution acts on the species set by splitting the set in several chunks and calculating several species distribution concurrently.

The following sections describe the steps a Generator of the Statistical Manager executes in order to produce the set $\{P(s,c)\}$ of probability distributions associated with the involved species. An overall picture of the whole process is in Figure 7.

2.5.1. Setup and Planning Phase In the setup phase of a D4Science processing, the Generator sends a message to the Information System in order to get (*i*) the list of Executor nodes containing the Generic Worker plug-in, hereafter called Worker nodes, and (*ii*) the location of an available Active Message Queue instance. The number of available nodes can be increased in a very straightforward way by means of the gCube enabling technology[‖]. Once the D4Science Worker nodes list has been gathered, the Generator sends a "wake-up" message to them. After that, the Generator performs the following operations:

1. create a queue named *D4Science* (Main-Queue) – if it does not exist – and a response queue with a unique dynamic name (Response-Queue), e.g., *D4Science-22-30-2012-efb1234-egh17*;
2. activate a queue producer on the Main-Queue and a queue consumer on the Response-Queue;

[‖]An Infrastructure administrator can add nodes via a web-based interface according to the expected load and to the privileges assigned to a community acting in a certain virtual research environment.

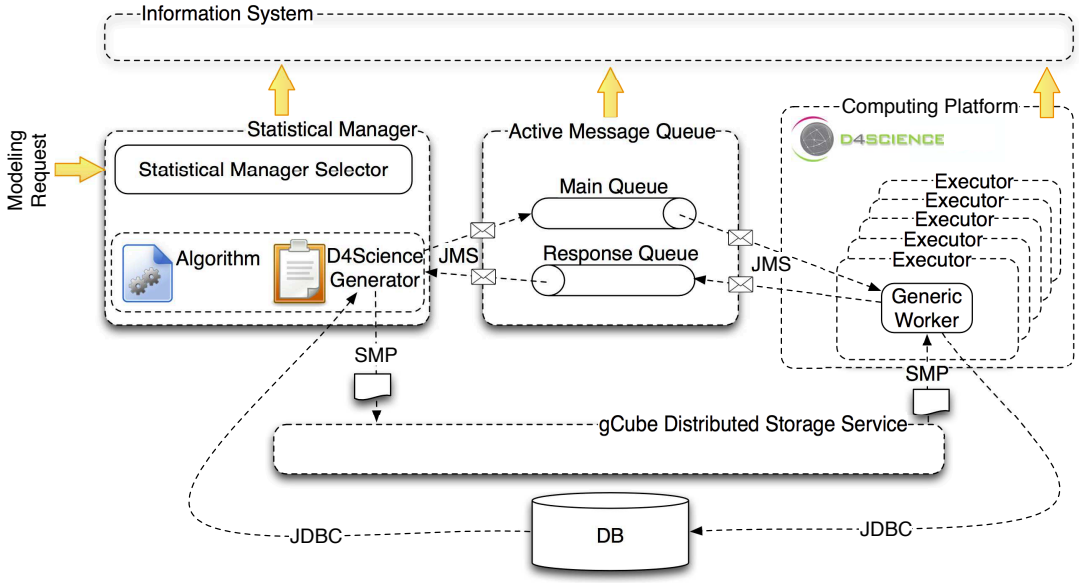

Figure 7. Representation of the D4Science Cloud Processing for the Aquamaps Suitable algorithm. The protocols are specified on the connection lines. The Generic Workers execute the species distribution algorithm on a data subset. In this particular case the algorithm writes directly on a database table. The Generator invokes the Reduce operation of the algorithm at the end of the procedure. The Reduce acts again on the database. The usage of a database is not mandatory. The computation relies on a Distributed Storage system for distributing the algorithm code. The communication protocol with the Distributed Storage is a gCube proprietary protocol called SMP.

3. configure the Generic Worker instances by means of the Executor client library. Each instance is asked to activate a queue consumer on the Main-Queue and a producer on the Response-Queue. The Active Queue IP address and port information is passed too.

The Worker nodes are now ready to consume messages and perform calculations.

At this point the Generator has to organise the actual computation by defining the "tasks" to be performed by the Worker nodes. In this phase the Generator will use (*i*) the *Algorithm package*, i.e., a standalone software package prepared by the SM and containing the script and libraries to be remotely executed; and (*ii*) the algorithm core and the expected input parameters.

In addition to that, the Generator caters for "data staging" as defined by each algorithm. In general, the choice of the system to use for storing input ad output is on the algorithm side. In some cases, the algorithm might have the requirement to use a database (this is the case of the computation depicted in Fig. 7). Alternatives can include the Distributed Storage system, or a partitioned table. The algorithms procedures that use the database are invoked by the Worker nodes during the processing and by the Generator in the setup phase. The usage of a database by the algorithms is not the optimal choice, as it can be a bottleneck for the whole processing. Anyway, in some cases the overhead due to the database operations can be disregarded, because such operations take much less time respect to the processing. The parallelization of the computation can considerably reduce the processing time respect to a sequential approach, even if a database is used a support. This is the case of Aquamaps Suitable algorithm, in which the probability calculations for a large number of species take the 80% of the overall computation time.

In the case of the Aquamaps Suitable algorithm a database is used for storing the species distributions. The algorithm creates a table during the setup phase, which is then used by the distributed computation. The algorithm part which runs on the Worker nodes uses that table to write the probability values. During the reduce phase, the Generator invokes a reduce method of the algorithm, which relies on the table content and checks for errors and duplicates.

The "tasks" preparation consists in defining the set of messages characterising the steps of a computation, e.g., a message might contain the portion of data to be processed by a Generic Worker. Assuming that the projection area has already been transformed in c-squares – activity that can be performed by exploiting the Transformation Engine (cf. Sec. 2.2.6) – the Generator retrieves the number of species and c-squares. The species set is then partitioned in n parts, where n is an external parameter related to the computational load which will be assigned to each node. A queue message is prepared for each of the n parts in which the species range to process is indicated. The species chunks are then arranged in a message sequence which is sent to the queue. In the case of the Aquamaps Suitable algorithm, the algorithm portions running on the Worker nodes are responsible for automatically reading the needed environmental features data directly from a data source in the infrastructure, which currently is a PostgreSQL instance. This means that the Generator has to include an encrypted file containing database connection information into the *Algorithm package*. The Generator has not direct connection to the database. Indeed it invokes a setup phase of the Aquamaps algorithm core, which builds such file. The Aquamaps core asks the Generator for database connection parameters and creates a connection file. The Generator includes this in the Aquamaps *Algorithm package* as a general configuration file and sends it to the Worker nodes.

To sum up, the setup phase consists in alerting the Worker nodes and creating the communication channels between the Generator and each Worker node while the planning phase consists in creating a number of messages entirely characterising a computation "task" to be performed by a Worker node.

2.5.2. Computation Start Once the package is ready, it is distributed to all the identified D4Science Worker nodes. The Distributed Storage system is useful to this scope: the Generator uploads the package on an area in the Storage, and sets the credential for accessing to such files. Only the D4Science Worker nodes will be allowed to download them. At this stage the Generator provides each queue message to be sent with the information about the package location on the storage. The sequence of messages is then ready and contains all the information needed by the nodes. By using the producer for the Main-Queue, the Generator adds messages to the queue and waits for those to be processed.

2.5.3. Request and Status Queues Every node that has been activated begins to consume queue messages. The content of each message instruct the Generic Worker to:

1. download a package from a certain location on the Distributed Storage;
2. focus on a certain script in the package;
3. pass the auxiliary information file to the script;
4. pass a range of species as input to the script;
5. execute the script in a sandbox location on the machine, with strong limitations on the maximum memory to use and permissions to write on the disk. As explained in Section 2.5.8 such limitations are calculated by the Generator on the basis of the requirements of the algorithm.

At the same time, a procedure is started which sends information about the status of the process by means of the producer on the Response-Queue. The possible status are:

1. STARTED: when the computation has started;
2. PROCESSING: when the node is computing;
3. FINISHED: when the node has correctly finished the computation;
4. FATAL-ERROR: when an unrecoverable error occurs;
5. RECOVERABLE-ERROR: when a recoverable error occurs and the Generator can overcome it by sending the message again.

During this phase the consumer on the Generator side, which is the only listener on the Response-Queue, will receive the status messages and will be then able to understand which messages are

being or have been processed. In the case of a RECOVERABLE-ERROR included in a certain message, the Generator sends that message again along with the indication to delete any previous value produced for that message. At the same time, knowing the sender of the status message, the Generator is able to calculate how many Worker nodes are really processing its messages.

In the case of the Aquamaps Suitable algorithm, each node produces several records on a table, where each row represents the probability $P(s, c)$ for the species s to have the c-square c as potential environment. Each message contains the instruction to process m species over all the k c-squares. The computation stops when all the $m \times k$ probabilities have been calculated.

2.5.4. Queue Information Broadcasting Once the computation has started (i.e., at least one node is processing), a daemon is created which periodically asks the IS for waking those Worker nodes that contain inactive Generic Worker plug-ins. This mechanism ensures that if a node starts after the computation had already begun or if a node has previously been busy, it will participate in the computation. Furthermore, it will enhance the processing parallelization from that moment on. The broadcast message produced by the daemon is the same as the initial setup message.

2.5.5. Reduce Phase When all the messages have been processed, the Generator closes the computation and checks for the correctness of the result. In this phase it sends a broadcast message to the Worker nodes for asking them to destroy both producers and consumers if no more messages are present on the respective queues. Then the Response-Queue is erased and the algorithm core is asked for running a *reduce* operation on the expected output. The algorithm implements the reduce phase as only it knows how to manage the outputs of the nodes. In this phase it checks for the presence of the expected outputs, which might be files on the storage system or tables etc. and eventually applies a merging phase for producing the final output.

2.5.6. Runtime Behavior The behavior of the whole computation described so far can be defined *dynamic* because the combined usage of the infrastructure broadcast instructions and queue messages permits a non-exclusive usage of the infrastructure Worker nodes. A gCube node can host several applications other than the Generic Worker plug-in, and this means that it may be busy when the computation starts. Furthermore, as D4Science is a community-based infrastructure, new Worker nodes can be activated at any time by some partner in the world, or by an infrastructure administrator. The Generic Worker plug-in is embedded in the standard gCube Hosting Node distribution. This means that, right after the deployment of such a software package on a server, a gCube Service can instantaneously participate to the computation. The same dynamic behaviour happens if a node crashes, because the control mechanisms discussed in the next section are able to recover the messages that have only partially been processed.

2.5.7. Control Mechanisms A set of control mechanisms are distributed between the Generator and the Worker nodes. Their aim is to lower the probability that a computation failure happens. This is achieved by means of retry and timeouts procedures placed at several points in the processing. Generally speaking, the number of systems involved in the computation implies that a failure might happen at various stages. The following is a list of retry mechanisms in D4Science which lower the probability that a random error could stop the computation:

1. Executor client invocations: 3 errors are allowed in contacting the IS for distributing the wake-up messages;
2. Storage upload / download: 3 errors are allowed in the upload to (at the Generator side) and download from (at the Worker nodes side) the Distributed Storage system;
3. Recoverable Errors: $3 \times totalnumberofmessages$ containing such error are allowed;
4. Fatal Errors: 2 fatal errors are allowed. After each fatal error the computation is started from the beginning;
5. Queue Connection: 3 errors are allowed for connecting to the Active Message Queue instance.

The timeout controls in the following list are able to avoid possible deadlocks in the computation:

1. Computation Start: 3 hours are allowed to pass before the computation starts. At the timeout the computation is treated as a fatal error occurred;
2. Queue Inactive: on the Worker nodes side the Main-Queue is declared to be inactive if it has been erased or if 1 hour has passed without receiving a message. In this case the node will destroy the listener;
3. Processing Messages: once a computation has started, the status messages are expected within a predefined period (30s by default). A timeout of 5 minutes declares a node to be inactive thus causing the message under processing be sent again.

2.5.8. Multi Client Behavior We use the term *cloud computation* for the process described so far, as the number of Worker nodes is variable and the requests for new computations are asynchronous.

In fact, once the computation has started, the species chunks are translated into messages which lie on a queue. The system does not require a continuous polling to know the status of the computation. The algorithm run by a Generator keeps information on how large an atomic computational step must be. Such step is represented by a single message sent to the queue. In the D4Science species processing, the algorithm defines a $MaxNumberOfSpeciesPerNode$ parameter, which sets a limit to the maximum number of species that can be involved in the processing of one message. It is to the algorithm developer to set such limit, in order not to overload the nodes in each computation step. Such parameter is then used by the Generator in preparing the messages. In the Aquamaps case, the parameter is set to 20, then each message sent to the queue contains indications to produce probability distributions for only a set of 20 species. Furthermore, the Generator defines a $SequenceLength$ parameter that accounts for concurrent processing. The parameter sets the length of a sequence of messages after which the Generator must wait for the processing to finish, before sending the next sequence. The Generator has then an intermittent behaviour, that avoids to fulfil the message queue when the number of messages is high. This situation can happen when many species must be processed.

The described mechanism is useful when multiple Generators concurrently ask for a computation. Suppose that two Generators wanted to perform a long computation at the same time. Each of them would use the described intermittent behaviour. Assuming one of the two is faster, the message queue receives $SequenceLength$ messages from the first Generator and then other $SequenceLength$ messages from the second. When the first sequence has been processed, the faster Generator sends a second sequence, which is enqueued to the one previously sent by the slower Generator. This means that the Worker nodes will alternately process sequences belonging to the two Generators. Such behaviour can be generalized by stating that the Worker nodes alternate sequences of messages sent by concurrent Generators. This allows for short computations to be managed even if they are preceded by long computations. It does not replace the possibility to associate priorities to job, but complement it by ensuring that each job starts almost immediately even if hundreds of other jobs are already started and are competing for the same set of computational resources. Thus, the capability of the Worker nodes to participate in more than one computations differentiates the presented approach from many high-throughput distributed computing systems that instead allocate a set of resources to a single job until the job is completed.

Finally, it is worth noticing that differently from many commercial platforms, D4Science does not allocate a potentially huge and indefinite number of Worker nodes to a specific computation. Workers are either hosted at partners' site (and their reliability is not granted) or are allocated to commercial cloud. In the latter case the privileges associated to a given community define quotas on the maximum number of resources concurrently exploitable. Worker nodes accessible through the infrastructure participate to several computations at the same time and dynamically accept new jobs even if other assigned jobs are still active.

2.5.9. Maps Publication Once the set of probabilities $\{P(s,c)\}$ has been produced and stored, e.g., as a tabular data source, the Statistical Manager can be asked to produce and publish a map for each species. This operation is achieved by exploiting the gCube GeoInteraction software library which

	Elapsed time (hours)
CLOUD (10 workers)	2.5
SINGLE-MULTI-CORE (10 cores)	3.5
SINGLE-MULTI-CORE (4 cores)	4.52
CLOUD (4 workers)	5.44
SEQUENTIAL	7.35

Table II. Computation time comparison of three approaches on the Aquamaps Suitable projection algorithm.

interfaces with the geospatial system described in Section 2.3. This library is invoked by the SM service which performs the following steps:

1. transform each c-square into a GIS geometry and store the $\{P(s,c)\}$ set on a table;
2. contact the GeoNetwork and retrieve the URL of the GeoServer containing the lowest number of layers (most unloaded GeoServer);
3. for each species s build a new GIS layer, based on the distribution $P(s,c)$;
4. update the GeoNetwork with the information about the layer's name, title and GeoServer URL.

By means of this procedure each species distribution will be retrievable by querying the GeoNetwork through CSW calls. Once information about the GeoServer that hosts the layer has been retrieved, then a standard OCG protocol can be used to visualize the map. The operation is automatically performed by the GeoExplorer portlet, while external clients can perform the same requests without using the gCube client libraries.

2.5.10. Performances In this section we will compare performances on a potential niche projection based on the Aquamaps Suitable algorithm using three different approaches:

1. SEQUENTIAL: the algorithm is applied without parallelization;
2. SINGLE-MULTI-CORE: the algorithm is parallelized on a single machine using several CPU cores;
3. CLOUD: the algorithm uses the D4Science Generator described in Section 2.4.

The first comparison is presented in Table II reporting on the generation of a probability distribution for 11,549 marine species belonging to the FishBase [36] repository. Local performances were calculated on a machine with the following characteristics: CentOS 5.7 x86_64, 16 CPUs, 8 GB of RAM, 500 GB of disk. D4Science Worker nodes, instead, were CentOS 5.7 x86_64 with 2 CPUs, 2 GB of RAM, 10 GB of disk. Data were stored on an external PostgreSQL database endowed with PostGIS [30]. All the machines resided at the same data center (CNR Pisa, Italy) and on the same LAN. The distribution produced by the 3 approaches was checked for generating the same results. The effectiveness is then the same in all the three cases.

It is evident that parallel processing is able to appreciably lower the computational time. The overhead due to the uploads on the Distributed Storage and the setup of the nodes is one of the reasons for which the SINGLE-MULTI-CORE is faster when the number of Worker nodes is less than 6. The overhead influence decreases as the number of Worker nodes increases, while the SINGLE-MULTI-CORE procedure suffers from the exploitation of many resources on the same machine for long time.

Figure 8 shows the trend of the single computation with respect to the cloud computation when the number of Worker nodes and cores increases. There is a point, around 6 nodes / cores, in which the cloud setup overhead is compensated by the parallelization on independent Worker nodes. From that point on, using a cloud computation on D4Science is more convenient with respect to a single machine computation. For sake of comparison with an execution on a multi-core server, only the execution on 10 D4Science Worker nodes has been reported. However, the same architecture has been exploited by communities having access to hundreds Worker nodes and it proved to be scalable and robust under heavy load.

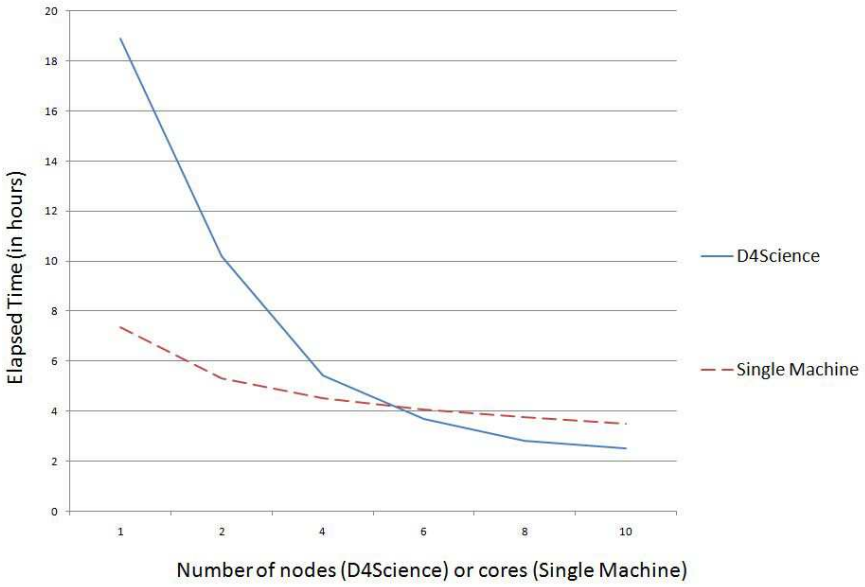

Figure 8. Performances comparison between the D4Science and Single Multi-Core machine computations at the variation of the number of Worker nodes and cores. The computation ran the Aquamaps Suitable Algorithm on 11549 marine species.

3. RELATED WORKS

Given the extent of the topics discussed in this paper, there are a lot of previous initiatives and works in fields ranging from tools specifically conceived to support the production of species distribution models, to initiatives offering access to the needed data and to federate infrastructures. In this section we provide an overview of some of these initiatives.

openModeller [37] is an open source software realising a single computing framework capable of handling different data formats and multiple algorithms that can be used in potential distribution modeling. The motivations leading to its development result from the willingness to provide scientists with a unified environment for species' distribution modeling. openModeler systematises the steps related to data management and to the algorithms exploited for modeling. It may exploit distributed computing resources through Condor but it requires tailored worker nodes specifically equipped with openModeller libraries. Moreover, it does not provide any facility for data discovery. It is expected that the users acquire data by themselves and transform them to the supported formats. The current version supports the following algorithms: ANN, AquaMaps, Bioclim, GARP, Climate Space Model, SVM, and Envelope Scores. Its functionality is offered via a command line tool, a desktop tool as well as via a Web Service.

Lifemapper [38] is a predictive electronic atlas of the Earth's biological biodiversity. The primary goal is to develop an up-to-date and comprehensive database of species maps and predictive models. It leverages a form of distributed computing inspired by SETI@Home [39], i.e., it relies on a screensaver version of the GARP genetic algorithm installed on volunteers PCs to produce species maps. The models are developed using specimen data from distributed museum collections and an archive of geospatial environmental correlates.

The Map of Life [6] aims at realising a global and web-based infrastructure for storing, sharing, producing, serving, annotating and improving diverse types of species distribution information. By combining and integrating diverse data types the aim is to mitigate the limitations they individually have, e.g., spatial or temporal grain, false positives or false negatives, global uniformity. To do this, they envisage an IT infrastructure comprising four major components: (*i*) an upload and storage

mechanism supporting a rich array of data including checklists, expert range maps, modelled distributions, focal species point records, allied species point records, area inventories, survey and atlas data, habitat preferences, species dependencies, dispersal capacity and related traits, phylogenetic relatedness, and detectability; (*ii*) a workbench supporting user-defined or semi-automatic data integration and modeling tasks to produce estimates of species distributions; (*iii*) a user interface supporting human users while uploading, searching and visualizing data as well as editing, commenting and voting on produced maps; and (*iv*) a number of APIs allowing the programmatic exploitation of the realized facilities. At the time of writing this paper, no details on the technical solutions to be exploited to realize the envisaged facilities are available.

Nativi *et al.* [40] discuss a service-oriented framework aiming at enabling scientists to do large-scale ecological analysis. Such a framework is conceived to realize an interoperability infrastructure compliant with the Global Earth Observation System of Systems (GEOSS) principles to enable the discovery and integration of multi-disciplinary data. In particular, they discuss an architecture consisting of: (*a*) a Biodiversity Data Provider which interfaces with GBIF; (*b*) a Climatological Data Provider which interfaces with NCAR GIS portal; (*c*) a Catalog which supports the discovery of occurrence points and environmental datasets; (*d*) a Model Provider which integrates OpenModeller and (*e*) a GUI and a User Scenario Controller which take care of realizing a typical biodiversity scenario aiming at predicting shifts in the spatial distribution of species' presence as a consequence of climate change. Recently, Nativi *et al.* [41] have discussed a "broker-oriented" approach as an effective strategy to be exploited when serving a multi-disciplinary data science.

Boyd and Foody [42] survey recent developments in GIS and remote sensing and their impact on species distribution modeling. In particular they highlight how GIS has been used as a component in many studies because of the flexibility it offers to analysts in relation to how data are used and what analytical criteria are employed in studies. For instance, Graham *et al.* [43] propose a Global Organism Detection and Monitoring system that relies on a GIS to allow users to immediately see a map of their data combined with other user's data. Users might display maps of invasive species distributions or abundances based on various criteria including taxonomic classification, a specific project, a range of dates, and a range of attributes (e.g., percent cover, age, height, sex, weight).

A lot of services aiming at offering web-based and user-friendly interfaces for visualising environmental data have been developed in a series of initiatives including, e.g., [44, 45, 6, 46]. None of them offers a comprehensive set of features as those offered by the GeoExplorer and GisViewer components (cf. Sec. 2.3).

High-throughput computing is a well-consolidated approach to provide large amounts of computational resources over long periods of time. It is worth to cite Condor [47] whose lifetime spans the last thirty years. Condor provides a job management mechanism by implementing scheduling policy, management of priorities, and resource monitoring. Differently from other batch management systems, Condor specifically provides support for opportunistic computing through the ability to use computing resources whenever they become available and without requiring dedicated worker nodes. At time of writing this paper, Condor is not exploited by D4Science. This is mainly due to the dispersed computing resources that lie in diverse administrative domains, span multiple countries, and are offered to the D4Science e-Infrastructure according to heterogeneous exploitation policies. However, single computational resource providers could adopt Condor to manage their resources and those resources could be registered into the D4Science e-Infrastructure. In this way they would become part of the D4Science trusted network. Specific Generators then should be added to the Statistical Manager to exploit the Condor pools.

Although the visibility cloud computing has acquired, it is commonly recognised that such a concept is not completely new and there are a number of connections with paradigms aiming at providing users with computational capabilities beyond those of their own computers like grid computing, utility computing and distributed systems in general [12, 48]. As a consequence of this, a number of distributed computing platforms and systems have been developed in the diverse domains. Sytems like gLite [49] and Globus [50] have been developed in the context of Grid Computing to support the development of Grid infrastructures. BOINC [51] is a software

system – resulting from the SETI@Home [39] experience – conceived to enable distributed computing by relying on resources belonging to the general public. All these initiatives share the common goal of offering computing as a utility. However, cloud goes beyond the provision of pure computing to promote an "as-a-Service" capability delivery model, e.g., [52, 53]. It is characterized by its elasticity and almost infinite scalability granted by commercial cloud providers. Besides optimizing the local usage and costs, Cloud providers might reach the infinite scalability goal by establishing agreements with other clouds to dynamically complement their local capacity, i.e., realising a *Federated Cloud*. Gomes *et al.* [54] discuss the mutual benefits resulting from the sharing of resources between clouds in federated clouds and propose a market-oriented mechanism to coordinate such a sharing.

Leveraging a federated model, D4Science offers a comprehensive set of computational resources that are then exploited by different services. The Statistical Manager service presented in this paper is just an example of such a service. It offers a common behaviour on heterogeneous computational systems, spanning from high-throughput computing systems to commercial cloud systems. Furthermore, it implements – across the diverse systems – the same policies for job checkpoint, migration, and prioritization of jobs and realizes a specific cloud computing approach (cf. Sec. 2.5).

4. CONCLUSIONS, CHALLENGES, AND OPPORTUNITIES

Species' distribution modeling is an effective tool for supporting studies aiming at predicting and understanding the distribution of species. It relies on the representation of the ecological requirements for a given species to survive. Although the capability to predict species' distribution is commonly considered a successful approach in many areas, there are some factors that prevent this facility to be a *commodity* at community of practice fingertips. Among the limiting factors there is the lack of a flexible and integrated environment supporting the entire process of species distribution modeling, from seamless data identification to maps production and dissemination.

In this paper, we have described how such an environment has been realised to support the species distribution modeling task in accordance with the modern ways of doing science and by leveraging on the offerings of an *Hybrid Data Infrastructure*. This Hybrid Data Infrastructure federates resources – ranging from hardware to data and software – of existing infrastructures (including Cloud). Its aim is to provide diverse communities of practice with innovative services for data management via dedicated *virtual research environments*.

In the next future our work will concentrate on another aspect underlying the species distribution modeling infrastructure described so far, i.e., the production of feature values at the expected input resolution. The assumption we made in the cloud processing description is that the geo-spatial features the user needs for the experiments are already available at the needed resolution. Obviously this is not always the case, especially when information is available only in certain zones, is not continuous or the available resolution is not the desired one. In these cases the geo-spatial data available in the infrastructure need some processing to be produced at novel points or simulated at lower or enhanced resolution. Our future plans will implement geo-spatial data processing by means of cloud computing facilities, endowed with a WPS protocol interface. Calculations required by high resolution simulations or kriging operations will surely benefit from the usage of cloud computing facilities. In such a vision the geo-spatial data provided by the infrastructure will be divided in (*i*) those yet ready to be distributed at the wanted resolution and (*ii*) those which are virtually available and will be produced on demand by the Statistical Manager. In the latter case, the SM will exploit WPS and use the same cloud infrastructures as in the case of the species distributions modeling algorithms.

ACKNOWLEDGEMENTS

This work has been partially supported by the VENUS-C project, co-funded within the 7th Framework

Programme by the GÉANT & e-Infrastructure Unit, Information Society & Media Directorate General of the European Commission, Grant No. 261565. It has been also partially supported by the *D4Science-II* project (FP7 of the European Commission, INFRA-2008-1.2.2, Grant No. 239019), the *iMarine* project (FP7 of the European Commission, FP7-INFRASTRUCTURES-2011-2, Grant No. 283644), and the *EUBrazilOpenBio* project (FP7 of the European Commission, ICT-2011.10.1.5, Grant No. 288754).

References

1. Guisan A, Thuiller W. Predicting species distribution: offering more than simple habitat models. *Ecology Letters* September 2005; **8**(9):993–1009.
2. Elith J, Leathwick JR. Species distribution models: Ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics* 2009; **40**:677–697.
3. Townsend Peterson A, Soberón J, Pearson RG, Anderson RP, Martínez-Meyer E, Nakamura M, Araújo MB. *Ecological Niches and Geographic Distributions*. Princeton University Press, 2011.
4. de Souza Dias BF. Towards Linking Ecosystems and Ecosystem Services to Economic and Human Activity. *Statement*, Convention on Biological Diversity 2012.
5. Guisan A, Zimmermann NE. Predictive habitat distribution models in ecology. *Ecological Modelling* 2000; **135**(2-3):147 – 186.
6. Jetz W, McPherson JM, Guralnick RP. Integrating biodiversity distribution knowledge: toward a global map of life. *Trends in Ecology & Evolution* 2012; **27**(3):151 – 159.
7. Hey T, Tansley S, Tolle K. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.
8. Boulton G, Campbell P, Collins B, Elias P, Hall DW, Laurie G, O'Neill O, Rawlins M, Thornton DJ, Vallance P, *et al.*. Science as an open enterprise. *Technical Report*, The Royal Society June 2012.
9. Candela L, Castelli D, Pagano P. Managing big data through hybrid data infrastructures. *ERCIM News* 2012; (89):37–38.
10. Atkins DE, Droegemeier KK, Feldman SI, Garcia-Molina H, Klein ML, Messerschmitt DG, Messina P, Ostriker JP, Wright MH. Revolutionizing science and engineering through cyberinfrastructure. Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure January 2003.
11. Thanos C. Global research data infrastructures: The GRDI2020 vision. GRDI2020 Booklet January 2012.
12. Foster I, Zhao Y, Raicu I, Lu S. Cloud Computing and Grid Computing 360-Degree Compared. *Grid Computing Environments Workshop, 2008. GCE '08*, 2008.
13. Candela L, Castelli D, Pagano P. Making Virtual Research Environments in the Cloud a Reality: the gCube Approach. *ERCIM News* October 2010; (83):32–33.
14. Candela L. Data Use - Virtual Research Environments. *Technological & Organisational Aspects of a Global Research Data Infrastructure - A view from experts*, Ashley K, Bizer C, Candela L, Fergusson D, Gionis A, Heikkurinen M, Laure E, Lopez D, Meghini C, Pagano P, *et al.* (eds.). GRDI2020, 2012; 91–98.
15. D4Science Consortium. D4Science: a Hybrid Data Infrastructure. URL `http://www.d4science.org`.
16. Kaschner K, Watson R, Trites AW, Pauly D. Mapping world-wide distributions of marine mammal species using a relative environmental suitability (RES) model. *Marine Ecology Progress Series* July 2006; **316**:285–310.
17. Kaschner K, Ready JS, Agbayani E, Rius J, Kesner-Reyes K, Eastwood PD, South AB, Kullander SO, Rees T, Close CH, *et al.*. AquaMaps: Predicted range maps for aquatic species. `http://www.aquamaps.org/` 2008.
18. Froese R, Pauly D. *FishBase 2000 Concepts, Design and Data Sources*. International Center for Living Aquatic Resources Management (ICLARM), 2000.
19. Deoliveira J. GeoServer: Uniting the 'geoweb' and Spatial Data Infrastructures. *Tenth International Conference for Spatial Data Infrastructure (GSDI-10*, 2008.
20. Rees T. "C-Squares," A New Spatial Indexing System and its Applicability to the Description of Oceanographic Datasets. *Oceanography* 2003; **16**(1):11–19.
21. Candela L, Castelli D, Pagano P. gCube: A Service-Oriented Application Framework on the Grid. *ERCIM News* January 2008; (72):48–49.
22. European Community. European Union Public Licence (EUPL) 1999. URL `http://joinup.ec.europa.eu/software/page/eupl`, version 1.1.
23. Heery R, Patel M. Application profiles: mixing and matching metadata schemas. *Ariadne* 2000; **25**.
24. Andreozzi S, Burke S, Ehm F, Field L, Galang G, Konya B, Litmaath M, Millar P, Navarro J. GLUE Specification v. 2.0. *Recommendation*, Open Grid Forum 2009.
25. Christensen E, Curbera F, Meredith G, Weerawarana S. Web Services Description Language (WSDL) 1.1. *Note*, W3C 2001.
26. Cattell R. Scalable sql and nosql data stores. *SIGMOD Rec.* May 2011; **39**(4):12–27, doi:10.1145/1978915.1978919. URL `http://doi.acm.org/10.1145/1978915.1978919`.
27. Anderson JC, Lehnardt J, Slater N. *CouchDB: The Definitive Guide*. O'Really, 2009.
28. SNIA. Cloud Data Management Interface (CDMI) v1.0.2. *Technical position*, SNIA 2012.
29. Snyder B, Bosanac D, Davies R. *ActiveMQ in Action*. Manning Publications Co., 2011.
30. Obe R, Hsu L. *PostGIS in Action*. Manning Publications Co., 2011.
31. McCandless M, Hatcher E, Gospodnetič O. *Lucene in Action, Second Edition*. Manning Publications Co., 2010.
32. Caron J, Davis E, Ho Y, Kambic R. UNIDATA's THREDDS Data Server. *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*, 2006.
33. Nogueras-Iso J, Zarazaga-Soria F, Béjar R, Álvarez P, Muro-Medrano P. OGC Catalog Services: a key element for the development of Spatial Data Infrastructures. *Computers & Geosciences* 2005; **31**(2):199 – 209, doi:10.1016/j.cageo.2004.05.015.
34. Prasanna D. *Dependency Injection*. Manning Publications Co., 2009.

35. Lee KH, Lee YJ, Choi H, Chung YD, Moon B. Parallel data processing with mapreduce: a survey. *SIGMOD Record* 2011; **40**(4):11–20.
36. Froese R, Pauly D. Fishbase 2011. Www.fishbase.org.
37. de Souza Muñoz ME, De Giovanni R, de Siqueira MF, Sutton T, Brewer P, Pereira RS, Canhos DAL, Canhos VP. openmodeller: a generic approach to species' potential distribution modelling. *GeoInformatica* 2011; **15**(1):111–135.
38. Stockwell DR, Beach JH, Stewart A, Vorontsov G, Vieglais D, Pereira RS. The use of the GARP genetic algorithm and internet grid computing in the lifemapper world atlas of species biodiversity. *Ecological Modelling* 2006; **195**(1-2):139 – 145, doi:10.1016/j.ecolmodel.2005.11.016.
39. Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D. Seti@home: an experiment in public-resource computing. *Commun. ACM* Nov 2002; **45**(11):56–61, doi:10.1145/581571.581573. URL `http://doi.acm.org/10.1145/581571.581573`.
40. Nativi S, Mazzetti P, Saarenmaa H, Kerr J, Tuama ÉÓ. Biodiversity and climate change use scenarios framework for the GEOSS interoperability pilot process. *Ecological Informatics* 2009; **4**(1):23–33.
41. Nativi S, Craglia M, Perlman J. The Brokering Approach for Multidisciplinary Interoperability: A Position Paper. *International Journal of Spatial Data Infrastructures Research* 2012; **7**:1–15.
42. Boyd D, Foody G. An overview of recent remote sensing and GIS based research in ecological informatics. *Ecological Informatics* 2011; **6**(1):25 – 36. Special Issue: 5th Anniversary.
43. Graham J, Newman G, Jarnevich C, Shory R, Stohlgren TJ. A global organism detection and monitoring system for non-native species. *Ecological Informatics* 2007; **2**(2):177 – 183.
44. Stocks K. SeamountsOnline: An Online Resource for Data on the Biodiversity of Seamounts. *Seamounts: Biodiversity and Fisheries*, *Fisheries Centre Research Report*, vol. 12, Morato T, Pauly D (eds.). Fisheries Centre, University of British Columbia, Canada, 2004; 13–24.
45. Bachman S, Moat J, Hill A, de Torre J, Scott B. Supporting Red List threat assessments with GeoCAT: geospatial conservation assessment tool. *Zookeys* 2011; **150**:117–126.
46. Grassle J. The Ocean Biogeographic Information System (OBIS): an on-line, worldwide atlas for accessing, modeling and mapping marine biological data in a multidimensional geographic context. *Oceanography* 2000; **13**(3):5–7.
47. Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience* 2005; **17**(2-4):323–356.
48. Zhang Q, Cheng L, Boutaba R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications* 2010; **1**:7–18, doi:10.1007/s13174-010-0007-6. URL `http://dx.doi.org/10.1007/s13174-010-0007-6`.
49. Laure E, Grandi C, Fisher S, Frohner A, Kunszt P, Krenek A, Mulmo O, Pacini F, Prelz F, White J, *et al.*. Programming the grid with glite. *Computational Methods in Science and Technology*, 2006.
50. Foster I, Kesselman C. Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications* 1997; **11**(2):115–128.
51. Anderson D. Boinc: a system for public-resource computing and storage. *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, 2004; 4 – 10, doi:10.1109/GRID.2004.14.
52. Allen B, Bresnahan J, Childers L, Foster I, Kandaswamy G, Kettimuthu R, Kordas J, Link M, Martin S, Pickett K, *et al.*. Software as a service for data scientists. *Commun. ACM* Feb 2012; **55**(2):81–88, doi:10.1145/2076450.2076468. URL `http://doi.acm.org/10.1145/2076450.2076468`.
53. Howe B, Cole G, Souroush E, Koutris P, Key A, Khoussainova N, Battle L. Database-as-a-service for long-tail science. *Proceedings of the 23rd international conference on Scientific and statistical database management*, SSDBM'11, Springer-Verlag: Berlin, Heidelberg, 2011; 480–489. URL `http://dl.acm.org/citation.cfm?id=2032397.2032436`.
54. Gomes ER, Vo QB, Kowalczyk R. Pure exchange markets for resource sharing in federated clouds. *Concurrency and Computation: Practice and Experience* 2012; **24**(9):977–991.