CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 0000; **00**:1–22
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Cloud Computing in a Distributed e-Infrastructure using the Web Processing Service standard

Gianpaolo Coro*, Giancarlo Panichi, Paolo Scarponi, Pasquale Pagano

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" – CNR, Pisa, Italy

SUMMARY

New Science paradigms have recently evolved to promote open publication of scientific findings as well as multi-disciplinary collaborative approaches to scientific experimentation. These approaches can face modern scientific challenges but must deal with large quantities of data produced by industrial and scientific experiments. These data, so-called "Big Data", require to introduce new Computer Science systems to help scientists cooperate, extract information, and possibly produce new knowledge out of the data. E-Infrastructures are distributed computer systems that foster collaboration between users and can embed distributed and parallel processing systems to manage Big Data. However, in order to meet modern Science requirements, e-Infrastructures impose several requirements to computational systems in turn, e.g. being economically sustainable, managing community-provided processes, using standard representations for processes and data, managing Big Data size and heterogeneous representations, supporting reproducible Science, collaborative experimentation, and cooperative online environments, managing security and privacy for data and services. In this paper, we present a Cloud computing system (gCube DataMiner) that meets these requirements and operates in an e-Infrastructure, while sharing characteristics with state-of-the-art Cloud computing systems. To this aim, DataMiner uses the Web Processing Service standard of the Open Geospatial Consortium and introduces features like collaborative experimental spaces, automatic installation of processes and services on top of a flexible and sustainable Cloud computing architecture. We compare DataMiner with another mature Cloud computing system and highlight the benefits our system brings, the new paradigms requirements it satisfies, and the applications that can be developed based on this system.
Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Data Mining, Parallel Processing, Cloud Computing, Big Data Processing, Distributed Systems, e-Infrastructures, WPS, Science 2.0

1. INTRODUCTION

The progress and evolution of Information Technology have changed the way Science is approached [1]. The large amount of data produced by Web and mobile applications and by scientific and industrial experiments have required the introduction of new systems to process these data, extract information and generate new knowledge. These data are usually classified as Big Data [2] and are characterised by at least six "V"s: large Volume, high production (and requested processing) Velocity, Variability in terms of complexity, Variety of representation formats, untrustworthiness (Veracity) of the information, and high commercial or scientific Value of the extracted information. In order to manage and process Big Data, non-conventional Computer Science systems are required.

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

In the last decade, new Science paradigms are born and have evolved to manage also Big Data and to promote collaborative experimentation and publication of scientific findings. These paradigms, e.g. Open Science [3], e-Science [4] and Science 2.0 [5], have grown in time and today their definitions overlap on several points. Overall, they support the open publication of results, findings and documents (possibly involving Big Data) related to scientific research and they also promote Computer Science systems that foster collaborative approaches to solve current complex problems of Science. These new paradigms also indicate requirements for new Computer Science systems, which involve: (i) collecting, analysing, and representing data to ensure their longevity and reuse; (ii) open publication of all the elements of scientific research (processes, data, documents etc.); (iii) supporting the three "R"s of the scientific method: Reproducibility, Repeatability, and Re-usability.

Big Data processing is one important point among the requirements of new Science paradigms. In order to make Big Data processing affordable, distributed computing has been extensively used in many domains. This kind of computing approach parallelises the computation on several cores/processors or machines in a network of computers [6]. Flexible approaches exist to distributed computing that tend to meet new Science paradigms requirements, but the current realisations are far from this goal (Section 2). Many distributed computing systems can manage specific community requirements but are typically too tied to certain repositories and data formats, and do not support collaborative experimentation. Further, aspects like open publication of the results and the support of the three "R"s of the scientific method are seldom taken into account. Other crucial indirect requirements include: flexibility in deployment, sustainability of the overall computational system, management of heterogeneous community requirements (e.g. different programming language etc.), and quick import of community-provided processes. These aspects require new components to be added to standard distributed computing systems.

In this context, e-Infrastructures (e-Is) are Computer Science systems that can meet the requirements of Big Data and new Science paradigms, and can build on top of distributed computing systems. An e-I is a network of hardware and software resources (e.g. Web services, machines, processors, databases etc.) that allows users or scientists residing at remote sites to collaborate and exchange information in a context of data-intensive Science [4]. In an extended definition, an e-I includes the following facilities: (i) distributed storage systems and parallel (e.g. High Performance Computing [7]) or distributed processing (e.g. Cloud or Grid computing [8]) systems for Big Data; (ii) services to manipulate, publish, harmonise, visualise, and access data, which also manage heterogeneous policies and formats; (iii) catalogues of the data and the resources accessible through the e-I; (iv) security and accounting services; (v) systems to support collaborative scientific research, through data sharing and social networking services.

The distributed computing systems used by e-Infrastructures parallelise the computations on a variable number of machines connected to the e-I at the time of the computation. These computing systems should also satisfy a number of requirements once immersed in an e-Infrastructure. In particular, they should (i) support the publication of the enabled processes (e.g. scripts, compiled programs etc.) as-a-Service, in order to allow their programmatic exploitation by other services either provided by the e-I or by the communities using the e-Infrastructure; (ii) support processes implemented under several programming languages to facilitate the communities' exploitation and take over; (iii) be interoperable with other services of the e-I through a standard representation of the processes and of their parameters, in order to enable either complex workflows composition or simple chained executions; (iv) save the provenance of an executed experiment, i.e. the set of input/output data, parameters, and metadata, in order to enable any authorised user to reproduce and repeat the experiment; (v) support data and parameters sharing through collaborative experimental spaces to enact collaboration between users; (vi) be economically and technically sustainable by enabling easy porting and deployment on several partners machines; (vii) support federated authentication and authorisation and tailored accounting facilities, in order to promote integration of existing technologies without compromising policies management and enforcement. Although many flexible and easy-to-install implementations of computational systems exist [9, 10, 11], no one is currently able to satisfy all these requirements. Limitations include poor interoperability between services developed by different providers, difficulty to integrate processes written by non-experts in Computer Science (e.g. biologists, agronomists, students, Big Data analysts etc.), complexity to manage the too heterogeneous Big Data representations and a general scarce use of standard representations to describe processes and data.

The aim of this paper it to describe an open-source Cloud computing system (gCube DataMiner [12]) that shares many features with state-of-the-art computational systems and also aims at satisfying the requirements of new Science paradigms, and in particular the e-Infrastructures requirements listed above. The presented platform adds functionalities for collaborative experimentation, standard description of the hosted methods and of the executed processes, and flexible provisioning of resources. Thanks to these functionalities, gCube DataMiner is a fully working peculiar system, currently managing tens of thousands requests per month [13]. In particular, gCube DataMiner is able to (i) interoperate with the services of the D4Science e-Infrastructure [14]; (ii) use the Web Processing Service (WPS [15]) standard to publish the hosted processes; (iii) save the provenance of an executed experiment using the Prov-O representation [16]. Our system implements a Cloud computing Map-Reduce approach for Big Data processing and saves the output as well as the provenance information onto a collaborative experimentation space, which allows a user to share this information with other colleagues. The DataMiner deployment is fully automatic and is spread across different machines providers (including the European Grid Infrastructure Federated Cloud system [17]). We compare our system with another mature system used by the same D4Science e-I [18, 19] to execute processes provided by communities of practice in computational biology. We demonstrate the higher computing performance of our system and its better flexibility to meet the requirements of e-Is on a use case of computational biology.

The paper is organised as follows: Section 2 gives an overview on Big Data, distributed and parallel computational systems, and explains to what extent these currently meet the requirements of new Science paradigms. Section 3 describes our system and explains its approach to Cloud computing. Section 4 reports the performance of our system compared with another Cloud computing system. Finally, Section 5 reports about current and possible applications of our system and draws the conclusions.

2. OVERVIEW

Big Data processing has been widely addressed using distributed computing techniques, because by definition this kind of data requires non-standard computing resources to be managed and processed [20]. Applications of distributed computing to Big Data have been described in many scientific papers [21, 22]. Indeed, extracting information from Big Data is crucial in Business Intelligence [23, 24] and in many fields where data mining is needed [23, 25]. The extracted information may have high impact in several domains, including geospatial and biological data processing [25, 26], because of the large spectrum of possible stakeholders, ranging from students and scientists to decision makers [27]. Crucial topics related to Big Data processing systems are scalability, availability, data staging, transformation, quality assessment and data heterogeneity management [21, 28, 29]. Further, privacy and access policies are important especially when data are shared or exchanged between users [30, 31]. These topics still require research effort to be properly addressed [32], especially for what regards collaborative approaches to data analysis and the subsequent reasoning and actions to produce knowledge in a multi-disciplinary context [33].

Processing Big Data with a distributed computing system typically involves parallelisation of the computation (and possibly split of the data) on several available cores/processors or machines of a network of computers. These resources intercommunicate and coordinate their actions to achieve the common task of processing all the data. Distributed computing has been extensively used in many fields of Science involving Big Data. For example, in computational biology Open Modeller [34] uses distributed computing for species distribution modelling, and Lifemapper [35] uses Grid computing to predict the distribution of world's fauna and flora. Other initiatives, like the Map of Life [36], rely upon distributed computing for retrieving and archiving species information. Distributed computing is also used to process geospatial data, especially when the required spatial resolution is high or the analysis is made at global scale [37]. Examples involve (i) discovering and integrating multi-disciplinary data for large-scale ecological analyses [38], (ii) comparing large-scale geographical distributions of environmental parameters [39], (iii) analysing climate change impact on species habitats [37, 40], (iv) forecasting fishing activity [41], (v) forecasting dust storms [42] etc. Cloud Computing has been highlighted as one critical paradigm to support the requirements of geospatial Big Data processing [42]. An overall survey of the Cloud computing resources that tackle geospatial Big Data challenges is reported by Yang et al. [43]. For example, the elasticity demanded to Cloud computing systems may address the production velocity and the variety of Big Data, whereas the on-demand provisioning of resources enacts the implementation of *ad hoc* solutions to manage data veracity and variety [43].

General platforms for Big Data processing propose general-purpose high-throughput computing services [44, 45, 46] that offer long-time computing as a utility. Among these platforms, Cloud computing systems [47] have several deployment models (e.g. private, community, public, hybrid cloud) and service models to provide computational and storage resources at many levels (application, platform, infrastructure).

Although these systems are powerful enough to manage a large amount of communities requirements, they do not natively meet modern Science paradigms requirements. For example, domain-specific implementations like the ones mentioned above, are typically too tied to a certain repository and format and do not support neither data retrieval from many knowledge sources nor collaborative experimentation. As for the general purpose systems, many of them tend to meet modern Science paradigms but not completely. For example, several systems stress on low maintenance, low deployment costs, and high sustainability, granting reliability, easiness of installation, and usability at the same time. On the other hand, aspects like managing different programming languages and importing community-provided processes are often neglected, although they are crucial to meet the programming habits of the served communities [19]. Overall, there is no system satisfying all the requirements reported in Section 1, nevertheless examples can be given of flexible systems that may potentially meet them.

Flexibility in deployment, installation and support of different programming languages can be found in HTCondor [44], an open-source framework to implement high-throughput and high-performance computing systems. However, HTCondor requires expertise to be properly configured and does not natively support either reproducibility and repeatability of experiments or collaborative experimentation. Apache Hadoop [48] provides Cloud computing and storage facilities and supports Map-Reduce computations. This system can be paired with many Apache packages and promotes interoperability with other Apache services. Nevertheless, installing, configuring and integrating new processes is not easy for non-expert developers, although every programming language is potentially supported. Also, communication standards are not natively supported and only unofficial attempts have been proposed [49]. Simplification in Cloud computations programming is promoted by Apache Spark [11], which extends the Hadoop approach and features and allows building more complex workflows than Map-Reduce. Nevertheless, using interoperability standards is still not supported.

Apart from software that allows building Cloud computing systems, operational Cloud computing platforms exist that provide on-demand interoperable machines and software to build new Cloud services. Examples are Amazon EC2 [50], Google Cloud Platform [51], and Microsoft Azure [52]. These support all the provisioning approaches of Cloud computing (Software- , Platform-, and Infrastructure-as-a-Service) but still require large effort for a non-expert community of practice to build a Cloud computing system based on communication standards.

Usage of Cloud computing in a collaborative e-Infrastructure is not frequent but some implementations exist, e.g. Yabi [9] and other domain-oriented systems [53, 54, 55]. However, these do not publish their processes under recognized standards. On the other hand, services exist that use standards to publish processes, but these do not natively support easy building of Cloud computing systems. For example, 52North [56] and Zoo-project [57] support WPS publication and easy chaining of processes, but are more oriented to easily integrate non-distributed processes than to design Cloud computations. Services and software exist [58, 59, 60] that embed clients for remote

WPS services in order to demand large computations to these remote systems. These system would greatly benefit from WPS-based Cloud computing platforms.

3. METHOD

This section describes the components DataMiner relies upon and the details of the system. It also explains how DataMiner manages parallel and Cloud computations. Further, it describes a Web GUI that allows the users of an e-Infrastructure to interact with DataMiner.

3.1. Components

DataMiner (DM) is immersed in the D4Science distributed e-Infrastructure, an e-I built by using the open-source gCube software [12]. DM was born to be interoperable with the services of this e-I. Theoretically, every process hosted by DM can use all the resources of the D4Science e-I. In this section, we explain the main e-I components our system uses during and after the computations.

3.1.1. e-Infrastructure Systems

The Information System

The Information System (IS) is a central service of most e-Infrastructures [61, 62, 63] and of D4Science too. It provides the e-I services with updated information about the location and the status of the available and reachable resources (e.g. databases, internal and external services, applications etc.). For example, it provides information about the IP addresses, the Operating Systems and the machine architectures of the computational, storage, and geospatial services available in the e-I. These services possibly reside at different sites and are hosted by different providers (e.g. Windows Azure [64], the European Grid Infrastructure [17] etc.). The IS acts as a registry of the e-I resources and manages their assignment to certain communities of users. The D4Science IS is the main service that supports publication, monitoring, discovering, and access for the e-I resources. Internally, it uses a Resource Management service to allocate and deploy/undeploy resources on-the-fly for users and services. The D4Science IS automatically indexes Apache Tomcat Web services [65], after these have been endowed with appropriate non-invasive Java libraries of the gCube software [66]. As for the other resources, e.g. databases and non-gCube services, indexing can be manually done by fulfilling online forms that are transformed into IS resource profiles. The D4Science IS is indeed made up of a number of services that ensure high availability of the information and high performance on concurrent queries for on-the-fly information retrieval. We leave the technical details of the D4Science IS to [67].

The Distributed Storage System

A Distributed Storage Systems (DSS) is a network of services that allows storing a large quantity of data with a flexible management of information availability and storage capability [68]. DSSs are not based on rigid schemas (like relational databases), but use different technologies, e.g. column store (e.g. Cassandra [69]) or document store (e.g. MongoDB [70]), to manage even non-structured or semi-structured data. Typically, they use data partitioning techniques where one dataset is divided into several smaller packages and each is stored by one high-availability service. The drawback with respect to most relational databases is that it is impossible to guarantee availability and consistency (i.e. always retrieving the latest version of a stored dataset) of the information at the same time (CAP Theorem [71]). D4Science uses the MongoDB document store to host files and metadata and extends it by (i) adding smart management of metadata and using unique identifiers for the datasets, (ii) enhancing the interoperability of MongoDB with the e-I services, (iii) enhancing access security, and (iv) introducing temporary storage areas [72]. The D4Science DSS adopts a partitioning strategy that uses a horizontally scalable architecture, with backup services for each storage node. This is the base layer of the D4Science data sharing and computational services. Indeed, the storage system is used as a common storage area by the services and the users of the e-I. In particular, it hosts software packages, files metadata, files in tabular (e.g. CSV, Excel etc.) and binary formats etc.

Data Sharing and Collaboration Facilities

D4Science users and services are all endowed with an online *Workspace* area. The Workspace is a service that builds on the D4Science DSS and is mainly used to upload, publish, and share data. In particular, virtual folders can be created (like in a virtual online file system) and each folder and dataset can be shared with other specific users of the e-I or made publicly accessible through public URLs. Thus, the Workspace is a shared online file system that can act as a way for users to exchange data, metadata, software, parameters and computational results. Beside this system, social networking facilities are provided to users and services to foster collaboration. In particular, the D4Science social networking services produce continuously updated lists of events/news, posted by users and applications through Web interfaces. An emailing system allows exchanging messages between users and services (for example after a computation) and allows also users to invite other people to join the e-I [73].

Geospatial Data Services

Services to manage environmental data are today integral parts of many e-Is [74, 58, 75] Environmental data are generally offered by several providers under different formats. Usually, the formats promoted by the Open Geospatial Consortium (OGC) are used, but these allow different representations of the same data content (e.g. Web Coverage Service [76] and OPeNDAP [77] for raster data) and thus require services to manage several representations, in order to retrieve and publish information. D4Science includes services to discover (GeoNetwork OpenSource [78]) and access (Geoserver [58] and Thredds [79]) raster and vector geospatial datasets, e.g. environmental and biological data. These services can possibly harvest information from other OGC compliant services and can publish data under OGC formats to visualise (e.g. using the Web Map Service [80]) and to extract information. The D4Science geospatial metadata are stored, indexed and published on a GeoNetwork instance in ISO-19139 compliant format according to the INSPIRE directives [81], in order to maximise the re-usability of the information by other e-Is [75]. These metadata can be also discovered and retrieved using the CSW and OAI-PMH standards. A Web application allows the D4science users to search for data and visualise/overlay maps [18].

Virtual Research Environments

Based on an e-Infrastructure, it is possible to define Virtual Research Environments (VREs) [82]. VREs are online collaborative environments, each focussed on a specific topic or community and corresponding to a subset of resources of the e-I that are temporarily assigned to a group of users. D4Science allows creating VREs in short operational time, where users are endowed with processes, Web interfaces, maps, and data that are suited for their domain of study. For example, a VRE for marine biologists can contain Web applications, databases, and maps that allow them to study marine protected areas. Sharing data and information through the e-I services available in the VRE, fosters multi-disciplinary activities. This may result, for example, in the production of new degradation indicators for marine protected areas. A VRE is usually moderated by a domain expert, who can supervise and regulate the activities of the VRE participants by relying on proper e-I services.

3.1.2. Standards for services and executions descriptions

Web Processing Service

The Web Processing Service (WPS) is an XML-based standard promoted by the OGC [83]. In particular, it defines a standard to publish and discover processes made available as-a-Service. Processes can be algorithms, calculations, and models that possibly process geospatial data. Publishing a process via WPS, means assigning it a machine-readable information that describes the process metadata and the accepted parameters, the inputs and the outputs. This also allows building catalogues of processes using OGC-compliant cataloguing services (e.g. GeoNetwork [78]). A WPS service should be conceived to offer both simple and complex processes, because WPS descriptions are lightweight enough to enable very fast interactions. The data to feed the processes can be made available either via public HTTP URLs or sent directly inside an HTTP-POST request. A WPS response document is produced after a process execution and can contain other OGC documents, e.g. Geography Markup Language (GML) documents. WPS processes are also suited to be used in workflows that combine several of them, e.g. the ones of the Galaxy, Taverna, and Knime [84] Workflow Management Systems.

The operations offered by a WPS service are mainly three [15]: (i) **GetCapabilities**, which allows to get the list of processes hosted by the service (capabilities); (ii) **DescribeProcess**, which returns a description of a process, of its input parameters, and expected output; (iii) **Execute**, which allows executing a process by passing a list of parameters and input data, and getting the output embedded in an XML document.

The WPS standard presents limitations with respect to the requirements of e-Infrastructures. In particular, in an e-I the capabilities should be specific of a certain community of practice, i.e. the list of processes should change according to the context the service is serving (i.e. the VRE). Instead, WPS supports only one GetCapabilities operation per service. Further, WPS requires the output to be specified at process description phase, which is not always the case for user-provided algorithms and for complex processes whose output is determined during the computation [85]. Another e-I requirement is to support security and accounting per user, in order to trace the activities and to allow interactions between selected users only (e.g. the VRE participants) or services. Finally, the WPS specifications do not give indications on how to implement a Cloud computing system completely based on WPS services, i.e. WPS is still considered an interface to interact with non-distributed computational system, whereas several benefits would be gained if all the services constituting a distributed computational system relied on WPS (Section 3.3). The WPS specifications are independent on these considerations, nevertheless, these may affect the interaction with a service. In the next section, we will explain how our system overcomes these limitations although based on WPS.

Provenance of a Computation

Provenance is defined as the information about the entities, activities, and people involved in the production of a dataset or a “thing” [86]. This information allows assessing the quality and the reliability of the produced objects. When referring to computations and experiments, the provenance is the set of information about input and output data, parameters, and metadata that allow to repeat or reproduce an experiment [87]. Trusted datasets (or experiments) are likely to be reused in several contexts, propagating correct citations at the same time. In that respect, provenance information potentially maximises the longevity and re-usability of data and experiments [1]. The authoritative W3C consortium promotes the PROV-O ontological conceptual data model of provenance [16, 88]. An XML schema is also proposed (Prov-XML) to practically represent key concepts and produce provenance documents for data and processes. Indeed, the PROV-O model is flexible enough to be applied to several domains and objects. The main elements of the PROV-O model are (i) “entities” and “activities” with creation/usage/end times associated; (ii) composite entities; (iii) “agents” responsible for entities and activities; (iv) “bundles” that combine several provenance objects; (v) “properties” of entities; (vi) “collections” of entities.

3.2. Architecture

DataMiner (DM) is a service based on the 52North WPS implementation [56], but it extends this implementation in order to meet the D4Science e-Infrastructure requirements. DM is developed with Java as a Web service running on an Apache Tomcat instance [65], endowed with gCube libraries. Further, DM offers a development framework to integrate algorithms and to interact with gCube-based e-Is (Section 3.3).

The complete DM architecture is made up of two sets of machines (clusters) that operate in a Virtual Research Environment (Figure 1): the Master and the Worker clusters. In a typical deployment scenario, the Master cluster is made up of a number of powerful machines (e.g. Ubuntu 14.04.5 LTS x86 64 with 16 virtual CPUs, 16 GB of random access memory, 100 GB of disk) managed by a load balancer that distributes the requests uniformly to these machines. Each machine is endowed with a DM service that communicates with the D4Science IS to notify its presence and capabilities. The balancer is indexed on the IS and is the main access point to interact with the DMs. The machines of the Worker cluster have less local computational power (e.g. Ubuntu 14.04.5 LTS x86 64 with 2 virtual CPUs, 2 GB of random access memory, 10 GB of disk) and serve Cloud computations, as explained in the next section.

The Master and the Worker clusters are dynamically provisioned by the D4Science e-I through an orchestration engine based on the OpenStack platform for Cloud computing [89]. Both the DM clusters are based on pre-cooked application templates predefined and registered on the IS, which indicate: (i) the required resources, i.e. servers, floating IPs, volumes, and security context (the VRE) to be used by a cluster; (ii) the relations and connections between these resources, e.g. the assignment of a volume to a server. The orchestration engine is endowed with a management software that uses Ansible scripts [90] to configure multi-tier applications in a reliable and consistent manner. Autoscaling is planned to be released, based on the metrics collected by the OpenStack Telemetry service [91]. Changes to the clusters configurations are managed through the update of the templates on the IS and are made by an e-I manager.

When a WPS request comes to the Master cluster balancer, it is distributed to one of the cluster services (Master DM). The DMs host processes provided by several developers. In particular, two kinds of algorithms are hosted: “Local” and “Cloud” algorithms. Local algorithms are directly executed on the Master DMs and possibly use parallel processing on several cores and a large amount of memory. Instead, Cloud algorithms use distributed computing with a Map-Reduce approach and rely on the DMs in the Worker cluster (Cloud nodes). With respect to the standard 52 North implementation, DM adds a number of features. First, it returns a different list of processes according to the VRE in which the service is invoked. When an algorithm is installed on a DM, it is also indexed on the IS as a resource. Thus, an e-Infrastructure manager can assign it to a number of VREs. When invoked in that VRE, DM returns only the subset of hosted processes that have been assigned to the VRE. Thus, a VRE for agronomists will not report algorithms suited for marine biologists or mathematicians and vice-versa. On the other hand, one may also want to create multi-disciplinary VREs with algorithms belonging to different domains.

Although the load balancers are stateless services, the DMs are stateful services. In fact, when a computation starts, an asynchronous client can ask about the status of the execution. The WPS protocol solves this issue by embedding a direct URL to the DM in the WPS answer produced to acknowledge process acceptance. This URL points to the DM service that reports the status of the computation and the client will contact this service directly. The drawback of this approach is that the WPS service should be directly reachable by clients.

Using a WPS standard in a Cloud computing system allows a number of thin clients to use the processes. Third party software (e.g. the well-known QGIS [60] and ArcMap [92] for geospatial data manipulation) can be able to retrieve the capabilities of a WPS service and run remote processes. Further, the gCube framework already offers clients for R and Java [93] and the WPS service can manage HTTP-GET requests. Thus, a process can be also invoked using a common Web browser, which makes it easy to repeat the execution of an experiment. Indeed, for each execution DM releases an “equivalent HTTP-GET” request to repeat the experiment via Web browser. Finally, an OpenCPU [94] instance is provided in the D4Science e-I, which transforms WPS objects into Javascript objects and allows for fast building of HTML applications.

The DataMiner services rely on the security services of the D4Science e-I and require a *user token* [95] to be provided for each operation. This token is passed via basic HTTPS-access authentication, which is supported by most WPS and HTTP(S) clients. The token identifies both a user and a Virtual Research Environment and this information is used by DM to query the IS about the capabilities to be offered in that VRE, i.e. the processes the user will be able to invoke with that authorization.

The DataMiner computations can take inputs from the D4Science Workspace. Inputs can also come from Workspace folders shared among several users. This fosters collaborative experimentation already at the input selection phase. Inputs can also come from external repositories, because a file can be provided either as an HTTP link or embedded in a WPS execution request. The outputs of the computations are written onto the D4Science DSS and are immediately returned to a client at the end of the computation. Afterwards, an independent thread also writes this information on the Workspace. Indeed, after every successfully completed computation, a Workspace folder is created that contains the input, the output, the parameters of the computation, and a provenance document summarizing this information. This folder can be shared with other people and used to execute the process again. Thus, the complete information about the execution can be shared and reused. This is the main way by which DataMiner fosters collaborative experimentation.

The DM processes can access to the resources available in a VRE by querying the IS. For example, it is possible to discover geospatial services, maps, databases, and files. The DM Java development framework simplifies the interaction with the IS. Since the IS interface is HTTP-REST too, the interaction with the IS can be managed by the processes directly. Further, the DM development framework provides methods to transform heterogeneous GIS formats into a numeric matrix and thus simplifies the effort to process geospatial data [39].

DataMiner can also import processes from other WPS services. If a WPS service is indexed on the IS for a certain VRE, its processes descriptions are automatically harvested, imported, and published among the DM capabilities for that VRE. During a computation, DM acts as a bridge towards the external WPS systems. Nevertheless, DM adds provenance management, authorization, and collaborative experimentation to the original service.

3.3. Computing

DataMiner is able to execute processes written in several programming languages, e.g. Java, R, C etc. A process integration framework is provided to developers [96] along with automatic tools to execute other programming languages through Java wrappers [97]. Developing a DM process requires extending one between two Java interfaces: the first (StandardLocalAlgorithm) is for processes that will run on one Master DM machine (possibly using several cores), the second (NodeAlgorithm) is for processes that will use the Worker cluster with a Map-Reduce approach.

The StandardLocalAlgorithm interface requires implementing an initialisation method (init()), a processing method (process()), and a post-processing method (shutdown()). Inputs are declared in a specific method (setInputParameters()) with an associated type that is automatically translated by DM into a WPS object, i.e. a literal (integer, string, double etc.) or a complex input (CSV, XML, binary files etc.). The inputs are all passed to the process as strings, which may contain the path to a file, the name of a table on a database (along with database coordinates), the value of a literal input etc. DataMiner is responsible for preparing the objects for a process. For example, it transforms CSV files into tables if required, otherwise it automatically downloads input files and passes their local paths to the process. Outputs are declared in a separate method (getOutput()) where the developer specifies the type and the list of the outputs that will be produced. If output cannot be defined before the execution of a process (*non-deterministic* output), a Java Map object should be fulfilled with primitive or complex objects during the computation. This map will be transformed by DM into a WPS complex data type containing a GML document. This GML document contains GML "feature collection members" [98] for each output, with HTTP links associated to files (stored on the D4Science DSS) in the case of complex objects (Figure 2).

As for Cloud computing algorithms (NodeAlgorithms), the interface requires specifying (i) a preliminary setup phase of the algorithm (setup(), part of the Map process), (ii) a post-processing method (postprocess()) that concludes the Reduce phase, and (iii) a processing method that accepts indices referring to a portion of the input to process. These indices are based on the assumption that the algorithm should process a Cartesian product of two sets of elements (*Left* and *Right* sets). Thus, the algorithm should provide the number of Left elements (getNumberOfLeftElements()) and of Right elements (getNumberOfRightElements()) to process. DataMiner will autonomously establish which portion of these elements each Cloud node will process. The Left and Right indices associated to an input portion will be passed to the process method on the nodes. Thus, the Map phase of the Cloud computation is managed at service level (on a Master DM), not at process level. This allows refining or changing the DM Map algorithm as soon as new features and services are available in the e-I (e.g. to control or add resources), independently on the processes implementations. Currently, this algorithm depends on the number of nodes available at the start of the computation. At the beginning of a computation, a Master DM receives a processing request and executes the setup method of the process. Thereafter, the DM queries the IS for the current list of nodes and the list of potential additional nodes that the e-I could instantiate according to the policies governing the VRE and the user's computational quota. The Left X Right product is divided into chunks that are equally virtually assigned to the estimated overall number of nodes (Map phase). A WPS request is prepared for each chunk to be sent to the Worker cluster and the Master DM creates a thread pool for these requests. Each node receiving the request will execute the process method on the indicated Left and Right elements. However, the size of the thread pool at the start of the process is equal to the number of nodes that are really available. This number is less than the number of prepared threads, which is indeed equal to the number of chunks. The pool size establishes how many threads should be instantiated concurrently, whereas the other ones will be queued.

During the computation, some machine could be unavailable. In this case, the other DM nodes will process more chunks. An internal queue on each node limits the number of concurrent chunks that can be processed by the node (the default is four). The Master DM periodically queries the IS to get the list of available nodes and possibly adjusts the size of the thread pool. If the number of chunks to process is still high after a certain period, more machines are requested to the IS. In turn, the IS tries to instantiate new DMs through appropriate provisioning services. The Reduce phase is called after all the chunks have been processed and follows the implementation of the postprocess() method of the process.

Each Master DM thread that processes a chunk of the computation is endowed with retry mechanisms that take care of possible computation failures due to random factors, e.g. a URL that is temporary unreachable, a machine that exhausted the resources or that is busy etc. The maximum number of retries is set to two times the number of available resources, because within this range the load balancer will likely send the request to a fully working machine, if available.

The presented Cloud computing approach assumes that all the DMs machines have the same processes installed. This may seem a strong assumption, but it is managed by means of a fully automatic provisioning strategy (Section 3.2). In fact, a process running on DM is typically made up of a Java archive (JAR) containing the process code and other dependencies, e.g. auxiliary scripts, compiled programs, Linux packages, R packages etc. When a developer delivers a process, (s)he should specify the set of dependencies required by that process. Auxiliary files should be made available through the Workspace and packages should be indicated by fulfilling proper forms [97]. During the deployment phase on a DM instance, this information is transformed into a sequence of Ansible playbook scripts [90] that install the process and all the auxiliary packages on the DM machine. This mechanism can be also invoked during a computation: based on the error returned by a computation on a node, the Master DM can recognize that the node misses the process and then it invokes the Ansible installation scripts on that machine. This mechanism also makes the maintenance of the DMs easier.

At the end of a computation, the produced complex outputs (files, images etc.) are saved on a temporary area of the D4Science DSS and returned as HTTP links (possibly also embedded in the WPS answer if not too large). Afterwards, a separate thread creates a Workspace folder where the inputs, outputs, parameters, and provenance information are saved. A user will be able to retrieve a summary of the computation through the Workspace, where the computation folder will be a standalone, self-consistent information object containing all the information required to execute the experiment again or to be shared with other users. As for the provenance, a Prov-O XML document is produced for each computation in the Workspace folder (Figure 3). In this document, a process is represented as an "activity" with time and user information attached. The operator, the VRE, the inputs and the parameters are all represented as "entities". The output is an entity too, which contains a reference to the activity it was generated from. Thus, this information can be extracted from the provenance document and can be reused to make the output recognizable as belonging to a certain computation.

3.4. Web Interface

DataMiner offers a Web GUI to the users of a VRE (Figure 4). On the left panel, the GUI presents the list of capabilities available in the VRE, which are semantically categorised (the category is indicated by the process provider). For each capability, the interface calls the DescribeProcess operation to get the descriptions of the inputs and outputs. When a user selects a process, in the right panel the GUI on-the-fly generates different fields corresponding to the inputs (Figure 4-a). Input data can be selected from the Workspace (the button associated to the input opens the Workspace selection interface). The "Start Computation" button sends the request to the DM Master cluster, which is managed as explained in Section 3.3. The usage and the complexity of the Cloud computations are completely hidden to the user, but the type of the computation is reported as a metadata in the provenance file.

In the end, a view of the Workspace folders produced by the computations is given in the "Check the Computations" area (Figure 4-b), where a summary sheet of the provenance of the experiment can be obtained ("Show" button, Figure 4-c). From the same panel, the computation can be also re-submitted. In this case, the Web interface reads the Prov-O XML information and rebuilds a computation request with the same parameters. The computation folders may also include computations executed and shared by other users. Finally, the "Access to the Data Space" button allows obtaining a list of the overall input and output datasets involved in the executed computations (Figure 4-d), with provenance information attached that refers to the computation that either used or executed the dataset.

4. RESULTS

In this section, we report the performance of DataMiner with respect to another mature system used by the D4Science e-I for biodiversity computation experiments (Statistical Manager or StatMan, abbreviated) and presented in [18]. A comparison between the two architectures is displayed in Figure 5. In StatMan, a users' request is queued until one of the master machines is ready to process it. Further, the Cloud computing machines (Generic Workers) are separate services that execute processes as standalone, compiled software. Thus, this system requires a process to be prepared also in a standalone version. On top of the Cloud machines, a queue service hosts messages corresponding to the chunks of a computation, which are consumed by the Generic Workers one at time. This approach is common to several distributed computing systems [10, 44, 48], thus StatMan may represent also other systems to a certain extent. For the comparison presented in this section, we used the same hardware configuration for both the architectures (StatMan and DataMiner); this approach ensures a fair comparison between the two. In particular, the master machines in both the systems were Ubuntu 14.04.5 LTS x86 64 with 16 virtual CPUs, 16 GB of random access memory and 100 GB of disk, and the Cloud computing machines were Ubuntu 14.04.5 LTS x86 64 with 2 virtual CPUs, 2 GB of random access memory, and 10 GB of disk. Software versions updated to Nov. 2016 were used for StatMan (v. 2.2.0) and DataMiner (v. 1.4.0), both running within Java Virtual Machines v. 1.7.0_80. The machines communicated with direct 1 Gbps connections in both the compared systems and the machine provisioning mechanism was the one reported in Section 3.2).

We compared the performance of the two systems on the BiOnym taxonomic search process [99]. This algorithm searches for the best transcription of a species scientific name in a copy of the authoritative FishBase names repository [100] hosted by the D4Science DSS. This repository contains taxonomic and biological information for 33,500 species with a size of ~30GB (to Nov. 2016) and an increasing complexity and volume trend proper to Big Data. Each input species name may contain misspelling errors and the process uses a combination of an expert system (embedding taxonomic experts' knowledge) with string matching algorithms (e.g. the Damerau-Levenshtein distance [101]) to find the "best" transcription. In particular, given a species scientific name, BiOnym produces suggestions of (i) the most likely corresponding transcriptions to the input name, (ii) the related official taxonomic name authorships, and (iii) metadata of the found records in the names repository (e.g. the creation date, the last modification date, the belonging specimen collection etc.). We selected this process because it is very used by marine biologists with ~30,000 requests per month [13]. BiOnym has a "Local" version, which searches for one scientific name directly using a Master DM, and a "Cloud" version that uses Cloud nodes. The Cloud version accepts a list of species as input, which is divided into several subsets that are possibly processed by different machines. From a DM process perspective, this means that the Left set has size one and the Right set has size equal to the number of species names to process. The same rationale was used by StatMan to partition the input.

In the comparison here reported, BiOnym was used to process 1024 species scientific names using the process' Cloud version on both StatMan and DataMiner. The input data file size was 100kB and contained misspelled taxonomic names, whereas the output was a ~150MB file containing full taxonomic names suggestions. Default similarity threshold parameters [99] were used for all the experiments. The differences in the computational times are reported in Figure 6-a using from 1 to 20 nodes. Overall, DM gets faster and faster with respect to StatMan as more nodes are used: with 20 nodes, the time reduction is ~88%, whereas with one node it is ~73%. The reason is explained in Figure 6-b, which reports the time required by each step of the computational process in both the systems. The average single-node computation is faster on DM, because the process is not executed in a standalone fashion. Indeed, StatMan executes a Java process by instantiating a new Java Virtual Machine in which the process is executed, whereas DM re-uses the same Java Virtual Machine of the service. The DM post-processing phase is faster too, because the output is written on the D4Science DSS directly, whereas the Workspace is used only after the computation and independently. Further, the setup phase of the processes on StatMan first uploads the software with its dependencies on the DSS and then engages the nodes using a Java SOAP-based client. Instead, by using WPS and pre-installed software, DM skips this preparation phase and makes the nodes engaging phase faster through lightweight REST interactions.

One DM can concurrently manage up to a pre-configured number of processes (four by default). The other requests are queued using the thread pool of the Tomcat server that hosts the DM application. One "Local" BiOnym computation (i.e. not using the Cloud nodes) requires ~10 seconds to finish. Thus, running 20 simultaneous BiOnym "Local" requests on DM requires: ~12s to manage the first 4 computations; ~24s for the second 4 requests (that have waited 12s for the first group to finish); ~36s for the third group (12 overall computations); ~48s for the fourth group (16 overall computations); ~60s for the fifth group (20 overall computations). It is notable that the concurrency introduces some delay due to the use of shared machine resources.

The advantages brought by DM with respect to StatMan are mainly due to its architecture and to the full exploitation of the WPS standard. One major advantage is that all the DM services publish their capabilities using a standard, which enhances the interoperability with other external services and software with respect to using custom clients. Further, the Master and the Worker clusters are managed by fast load balancers that are able to dynamically add machines and to ignore them when offline. The Cloud nodes are exact replicas of the Master nodes with less computational resources. This allows using the Worker cluster directly from clients and fosters alternative usages of the Cloud computing system [102]. For example, external users of the e-I (authorised with proper tokens) may also implement their own Cloud computations by invoking the Worker cluster in custom workflows. Further, the Worker cluster can be used instead of the Master cluster to support a community of practice (i.e. a VRE), when low computational power is required but the number of requests is high. Thus, the DM architecture maximises the reuse of the machines, which is particularly useful for small e-Infrastructures.

5. CONCLUSIONS

In this paper, we have presented a peculiar computational system that goes towards the requirements of modern Science paradigms, especially because it blends together Cloud computing and e-Infrastructures. Our system adopts a collaborative approach, standards to describe both the hosted methods and the executed processes, and a flexible provisioning of resources. With respect to other distributed computing systems, it satisfies additional requirements brought by e-Infrastructures, and the benefits are many. First, the usage of a process is improved, thanks to the possible interoperability with data preparation and harmonisation services (e.g. [103, 104, 105]), which speed up the typical time-consuming phase of data preparation for an experiment. Further, providing a shared DSS and an experimentation Workspace area allows reusing the results of processes and also fosters multi-disciplinary experiments. Users could also be services or external machines (e.g. sensors) that produce experimental data at different frequencies and time scales, while other processes analyse these data and take decisions. In other words, it would be straightforward to build a blackboard-like system [106] using DataMiner. The same facilities are automatically offered to desktop software supporting WPS. Further, generating and storing provenance information improves the possibility to repeat and reproduce an experiment executed by other scientists. Finally, since processes and service installation is fully automatic, it is easy to deploy DataMiner on a number of machines providers.

Currently, DM is hosted by virtual machines of the European Grid Infrastructure (EGI), the Slovak Academy of Sciences (IISAS), the Göttingen University and the Max Planck Society (GWDG), the Supercomputing Center of Galicia (CESCGA), the Universitat Politècnica de València (UPV-GRyCAP), the University of Bari (ReCaS), the National Italian Institute of Nuclear Physics (INFN), the National Research Council of Italy (CNR), and the University of Athens. DataMiner is used in several European projects (e.g. BlueBRIDGE [107], ENVRIPlus [108], Parthenos [109], SoBigData.eu [110]) and by international organisations (e.g. the Food and Agriculture Organisation of the United Nations and the National Oceanic and Atmospheric Administration) in appropriate Virtual Research Environments, also for educational purposes. DM hosts ~200 processes and serves overall 55 VREs in D4Science[†], exploited by more than 2500 scientists in 44 countries. The hosted processes are written with the Java, R, Fortran, C, Octave, and Python programming languages and have been provided by developers with heterogeneous expertise (e.g. biologists, mathematicians, agronomists, physicists, data analysts etc.).

E-Infrastructures are valid Computer Science systems that may solve issues brought by Big Data and new Science paradigms if properly implemented. The DataMiner system goes in this direction, although further development is required to improve crucial points that do not depend on DataMiner only, e.g. native interoperability with services developed by other providers, management of many standards to access and retrieve data, staging Big Data on the computing machines and managing near-real time processing.

The focus on interoperability and standards makes DataMiner interesting also for industrial processes, e.g. for the fully automatic and interconnected processes of the smart cities and factories of the Industry 4.0 trend [111]. These systems could benefit from the domain-specific, expert-provided and state-of-the-art processes (e.g. Artificial Neural Networks, Support Vector Machines, Deep Learning processes, Monte Carlo methods etc. [112]) hosted by DataMiner and executed in short-time. Indeed, as also suggested by the modern Science paradigms, it is more and more evident that the isolation of Computer Science systems and domain experts is no more sustainable with respect to the complexity of modern scientific problems.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the BlueBRIDGE project (Grant agreement No 675680).

References

1. Hey T, Tansley S, Tolle KM, *et al.*. *The fourth paradigm: data-intensive scientific discovery*, vol. 1. Microsoft research Redmond, WA, 2009.
2. James M, Michael C, Brad B, Jacques B, Richard D, Charles R, Angela H. Big data: The next frontier for innovation, competition, and productivity. *The McKinsey Global Institute* 2011; .
3. EU Commission. Open science (open access) 2016. `https://ec.europa.eu/programmes/horizon2020/en/h2020-section/open-science-open-access`.
4. Andronico G, Ardizzone V, Barbera R, Becker B, Bruno R, Calanducci A, Carvalho D, Ciuffo L, Fargetta M, Giorgio E, *et al.*. e-infrastructures for e-science: a global view. *Journal of Grid Computing* 2011; **9**(2):155–184.
5. Waldrop MM. Science 2.0. *Scientific American* 2008; **298**(5):68–73.

[†]accessible after free registration at `https://services.d4science.org`

6. Attiya H, Welch J. *Distributed computing: fundamentals, simulations, and advanced topics*, vol. 19. John Wiley & Sons, 2004.
7. Yang LT, Guo M. *High-performance computing: paradigm and infrastructure*, vol. 44. John Wiley & Sons, 2005.
8. Pike RC, Thompson KL. Distributed computing system Apr 22 1997. US Patent 5,623,666.
9. Hunter AA, Macgregor AB, Szabo TO, Wellington CA, Bellgard MI. Yabi: An online research environment for grid, high performance and cloud computing. *Source code for biology and medicine* 2012; **7**(1):1.
10. Wang L, Tao J, Ranjan R, Marten H, Streit A, Chen J, Chen D. G-hadoop: Mapreduce across distributed data centers for data-intensive computing. *Future Generation Computer Systems* 2013; **29**(3):739–750.
11. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: cluster computing with working sets. *HotCloud* 2010; **10**:10–10.
12. National Research Council of Italy. The open-source gcube system to build distributed e-infrastructures 2016. `www.gcube-system.org`.
13. National Research Council of Italy. The gCube DataMiner usage statistics 2017. `https://gcube.wiki.gcube-system.org/gcube/Data_Mining_Facilities#Current_usage_statistics`.
14. National Research Council of Italy. The D4Science Distributed e-Infrastructure 2016. `http://www.d4science.org`.
15. Schut P, Whiteside A. OpenGIS Web Processing Service 2007. OGC project document `http://www.opengeospatial.org/standards/wps`.
16. Lebo T, Sahoo S, McGuinness D, Belhajjame K, Cheney J, Corsar D, Garijo D, Soiland-Reyes S, Zednik S, Zhao J. Prov-o: The prov ontology. *W3C Recommendation* 2013; **30**.
17. European Grid Infrastructure. The european grid infrastructure federated cloud 2016. `www.egi.eu/federation/`.
18. Candela L, Castelli D, Coro G, Pagano P, Sinibaldi F. Species distribution modeling in the cloud. *Concurrency and Computation: Practice and Experience* 2013; .
19. Coro G, Candela L, Pagano P, Italiano A, Liccardo L. Parallelizing the execution of native data mining algorithms for computational biology. *Concurrency and Computation: Practice and Experience* 2015; **27**(17):4630–4644.
20. Kaisler S, Armour F, Espinosa JA, Money W. Big data: Issues and challenges moving forward. *System sciences (HICSS), 2013 46th Hawaii international conference on*, IEEE, 2013; 995–1004.
21. Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Khan SU. The rise of big data on cloud computing: Review and open research issues. *Information Systems* 2015; **47**:98 – 115, doi:http://dx.doi.org/10.1016/j.is.2014.07.006. URL `http://www.sciencedirect.com/science/article/pii/S0306437914001288`.
22. Chen M, Mao S, Liu Y. Big data: A survey. *Mobile Networks and Applications* 2014; **19**(2):171–209.
23. Fernández A, del Río S, López V, Bawakid A, del Jesus MJ, Benítez JM, Herrera F. Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2014; **4**(5):380–409.
24. Chen H, Chiang RH, Storey VC. Business intelligence and analytics: From big data to big impact. *MIS quarterly* 2012; **36**(4):1165–1188.
25. Wu X, Zhu X, Wu GQ, Ding W. Data mining with big data. *ieee transactions on knowledge and data engineering* 2014; **26**(1):97–107.
26. Russom P, *et al.*. Big data analytics. *TDWI best practices report, fourth quarter* 2011; :1–35.
27. McAfee A, Brynjolfsson E, Davenport TH, Patil D, Barton D. Big data. *The management revolution. Harvard Bus Rev* 2012; **90**(10):61–67.
28. Ranjan R. Streaming big data processing in datacenter clouds. *IEEE Cloud Computing* 2014; **1**(1):78–83.
29. Tsuchiya S, Sakamoto Y, Tsuchimoto Y, Lee V. Big data processing in cloud environments. *Fujitsu Sci. Tech. J* 2012; **48**(2):159–168.
30. Demchenko Y, Grosso P, De Laat C, Membrey P. Addressing big data issues in scientific data infrastructure. *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, IEEE, 2013; 48–55.
31. Chaudhuri S. What next?: a half-dozen data management research goals for big data and the cloud. *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, ACM, 2012; 1–4.
32. Agrawal D, Das S, El Abbadi A. Big data and cloud computing: Current state and future opportunities. *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, ACM: New York, NY, USA, 2011; 530–533, doi:10.1145/1951365.1951432. URL `http://doi.acm.org/10.1145/1951365.1951432`.
33. Chen Y, Alspaugh S, Katz R. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *Proc. VLDB Endow.* Aug 2012; **5**(12):1802–1813, doi:10.14778/2367502.2367519. URL `http://dx.doi.org/10.14778/2367502.2367519`.
34. Santana F, Fonseca R, Saraiva A, Corrêa P, Bravo C, Giovanni R. openmodeller-an open framework for ecological niche modeling: analysis and future improvements. *World Conference on Computers in Agriculture and Natural Resources*, 2006.
35. 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):139–145.
36. 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.
37. Yang C, Goodchild M, Huang Q, Nebert D, Raskin R, Xu Y, Bambacus M, Fay D. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? *International Journal of Digital Earth* 2011; **4**(4):305–329.
38. 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.
39. Coro G, Pagano P, Ellenbroek A. Comparing heterogeneous distribution maps for marine species. *GIScience & Remote Sensing* 2014; **51**(5):593–611.

40. Coro G, Magliozzi C, Ellenbroek A, Kaschner K, Pagano P. Automatic classification of climate change effects on marine species distributions in 2050 using the aquamaps model. *Environmental and ecological statistics* 2016; **23**(1):155–180.
41. Coro G, Large S, Magliozzi C, Pagano P. Analysing and forecasting fisheries time series: purse seine in indian ocean as a case study. *ICES Journal of Marine Science: Journal du Conseil* 2016; :fsw131.
42. Yang C, Huang Q, Li Z, Liu K, Hu F. Big data and cloud computing: innovation opportunities and challenges. *International Journal of Digital Earth* 2017; **10**(1):13–53.
43. Yang C, Yu M, Hu F, Jiang Y, Li Y. Utilizing cloud computing to address big geospatial data challenges. *Computers, Environment and Urban Systems* 2017; **61**:120–128.
44. Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the condor experience. *Concurrency and computation: practice and experience* 2005; **17**(2-4):323–356.
45. Laure E, Edlund A, Pacini F, Buncic P, Barroso M, Di Meglio A, Prelz F, Frohner A, Mulmo O, Krenek A, *et al.*. Programming the grid with glite. *Technical Report*, CERN 2006.
46. Foster I, Kesselman C. Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications* 1997; **11**(2):115–128.
47. Mell P, Grance T. The nist definition of cloud computing. *National Institute of Standards and Technology* 2009; **53**(6):50.
48. Apache Software Fundation. Apache hadoop. *Online publication:* `http://hadoop.apache.org` 2011; .
49. Worldprogramming. Wps configuration for hadoop 2016. `https://www.worldprogramming.com/docs/wps/documentation/3.2/07%20interop_for_hadoop/WPS-Configuration-for-Hadoop.pdf/WPS-Configuration-for-Hadoop-en.pdf`.
50. Amazon Inc. Amazon elastic compute cloud (amazon ec2). *Amazon Elastic Compute Cloud (Amazon EC2)* 2010; .
51. Google Inc. Google cloud platform 2016. `cloud.google.com`.
52. Microsoft Inc. Microsoft azure platform 2016. `azure.microsoft.com`.
53. Hull D, Wolstencroft K, Stevent R, Globe C, Pocock M, Li P, Oinn T. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 2006; **1**(34):729–732.
54. Giardine B, Riemer C, Hardison RC, Burhans R, Elnitski L, Shah P, Zhang Y. Galaxy: a platform for interactive large-scale genome analysis. *Genome Research* 2005; **10**(15):1451–1455.
55. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Thiel K, Wiswedel B. Knime-the konstanz information miner: version 2.0 and beyond. *AcM SIGKDD explorations Newsletter* 2009; **11**(1):26–31.
56. 52North. The 52north wps service 2016. `http://52north.org/communities/geoprocessing/wps/`.
57. Zoo-Project. Zoo open wps platform 2016. `www.zoo-project.org`.
58. Steiniger S, Hunter AJ. Free and open source gis software for building a spatial data infrastructure. *Geospatial free and open source software in the 21st century* 2012; :247–261.
59. Degree. Open source software for spatial data infrastructures 2016. `www.deegree.org`.
60. QGIS. A free and open source geographic information system 2016. `http://qgis.org/en/site/`.
61. Pollock N, Williams R. E-infrastructures: How do we know and understand them? strategic ethnography and the biography of artefacts. *Computer Supported Cooperative Work (CSCW)* 2010; **19**(6):521–556.
62. Chapman A, Russell R. Jisc shared infrastructure services synthesis study: A review of the shared infrastructure for the jisc information environment. *Online Publication:* `http://opus.bath.ac.uk/17890/` 2006; .
63. Hey T, Trefethen AE. Cyberinfrastructure for e-science. *Science* 2005; **308**(5723):817–821.
64. Redkar T, Guidici T, Meister T. *Windows Azure Platform*, vol. 1. Springer, 2011.
65. Apache Software Fundation. Apache tomcat 2016. `http://tomcat.apache.org/`.
66. National Research Council of Italy. Smartgears - gcube system. java libraries to turn servlet-based containers and applications into gcube resources transparently 2016. `https://wiki.gcube-system.org/gcube/SmartGears`.
67. National Research Council of Italy. The d4science information system 2016. `https://wiki.gcube-system.org/gcube/Information_System`.
68. Dimakis AG, Godfrey PB, Wu Y, Wainwright MJ, Ramchandran K. Network coding for distributed storage systems. *IEEE Transactions on Information Theory* 2010; **56**(9):4539–4551.
69. Lakshman A, Malik P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review* 2010; **44**(2):35–40.
70. Banker K. *MongoDB in action*. Manning Publications Co., 2011.
71. Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News* 2002; **33**(2):51–59.
72. National Research Council of Italy. The d4science distributed storage management system 2016. `https://wiki.gcube-system.org/gcube/Storage_Management`.
73. Assante M, Candela L, Castelli D, Coro G, Lelii L, Pagano P. Virtual research environments as-a-service by gcube. *PeerJ Preprints* 2016; **4**:e2511v1.
74. Groot R, McLaughlin JD. *Geospatial data infrastructure: concepts, cases, and good practice*. Oxford university press Oxford, 2000.
75. Trumpy E, Coro G, Manzella A, Pagano P, Castelli D, Calcagno P, Nador A, Bragasson T, Grellet S, Siddiqi G. Building a european geothermal information network using a distributed e-infrastructure. *International Journal of Digital Earth* 2016; **9**(5):499–519.
76. Shekhar S, Xiong H. Web coverage service. *Encyclopedia of GIS*. Springer, 2008; 1255–1255.
77. Cornillon P, Gallagher J, Sgouros T. Opendap: Accessing data in a distributed, heterogeneous environment. *Data Science Journal* 2003; **2**:164–174.
78. Ožana R, Horáková B. Actual state in developing geonetwork opensource and metadata network standardization. *GIS Ostrava 2008* 2008; .

79. Bergamasco A, Benetazzo A, Carniel S, Falcieri F, Minuzzo T, Signell RP, Sclavo M. Knowledge discovery in large model datasets in the marine environment: the thredds data server example. *Advances in Oceanography and Limnology* 2012; **3**(1):41–50.
80. de La Beaujardiere J. Opengis® web map server implementation specification. *Open Geospatial Consortium Inc., OGC* 2006; :06–042.
81. INSPIRE. Generic conceptual model 2016. `http://inspire.ec.europa.eu/documents/Data_Specifications/D2.5_v3.4rc3_vs_3.4rc2.pdf`.
82. Fraser M. Virtual research environments: overview and activity. *Ariadne* 2005; **1**(44).
83. Lupp M. Open geospatial consortium. *Encyclopedia of GIS*. Springer, 2008; 815–815.
84. Deelman E, Gannon D, Shields M, Taylor I. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems* 2009; **25**(5):528–540.
85. Rogers H, Rogers H. *Theory of recursive functions and effective computability*, vol. 126. McGraw-Hill New York, 1967.
86. Freire J, Koop D, Santos E, Silva CT. Provenance for computational tasks: A survey. *Computing in Science & Engineering* 2008; **10**(3):11–21.
87. Simmhan YL, Plale B, Gannon D. A survey of data provenance in e-science. *ACM Sigmod Record* 2005; **34**(3):31–36.
88. Moreau L, Missier P. Prov-dm: The prov data model. *Online publication:* `https://www.w3.org/TR/prov-dm/` 2013; .
89. Sefraoui O, Aissaoui M, Eleuldj M. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications* 2012; **55**(3).
90. Ansible. Playbook documentation 2016. `http://docs.ansible.com/ansible/playbooks.html`.
91. OpenStack. Telemetry 2017. `https://wiki.openstack.org/wiki/Telemetry`.
92. ArcMap. Arcgis for desktop 2016. `http://desktop.arcgis.com/en/arcmap/`.
93. National Research Council of Italy. gcube wps thin clients 2016. `https://wiki.gcube-system.org/gcube/How_to_Interact_with_the_DataMiner_by_client`.
94. OpenCPU. Producing and reproducing results 2016. `https://www.opencpu.org`.
95. National Research Council of Italy. gcube token-based authorization system 2016. `https://gcube.wiki.gcube-system.org/gcube/Authorization_Client_Library`.
96. National Research Council of Italy. Dataminer process integration framework 2016. `https://wiki.gcube-system.org/gcube/How-to_Implement_Algorithms_for_DataMiner`.
97. Coro G, Panichi G, Pagano P. A web application to publish r scripts as-a-service on a cloud computing platform. *Bollettino di Geofisica Teorica e Applicata* 2016; **52**:article n. 51 53.
98. Burggraf DS. Geography markup language. *Data Science Journal* 2006; **5**:178–204.
99. Berghe EV, Coro G, Bailly N, Fiorellato F, Aldemita C, Ellenbroek A, Pagano P. Retrieving taxa names from large biodiversity data collections using a flexible matching workflow. *Ecological Informatics* 2015; **28**:29–41.
100. Froese R, Pauly D. *FishBase 2000: Concepts Designs and Data Sources*, vol. 1594. WorldFish, 2000.
101. Bard GV. Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric. *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*, Australian Computer Society, Inc., 2007; 117–124.
102. BlueBRIDGE European Project. The ichthyop model use case 2016. `https://support.d4science.org/projects/bluebridge/wiki/Ichthyop`.
103. National Research Council of Italy. The gCube Tabular Data Manager data preparation system 2016. `https://wiki.gcube-system.org/gcube/Tabular_Data_Manager`.
104. Candela L, Castelli D, Coro G, Lelii L, Mangiacrapa F, Marioli V, Pagano P. An infrastructure-oriented approach for supporting biodiversity research. *Ecological Informatics* 2015; **26**:162–172.
105. RStudio Inc. An online RStudio instance for data preparation of the D4Science Biodiversity Lab Virtual Research Environment 2016. `https://i-marine.d4science.org/group/biodiversitylab/r-studio`.
106. Nii HP. Blackboard application systems, blackboard systems and a knowledge engineering perspective. *AI magazine* 1986; **7**(3):82.
107. The BlueBRIDGE European Project. Project web site 2016. `www.bluebridge-vres.eu`.
108. The EnvriPlus European Project. Project web site 2016. `www.envriplus.eu`.
109. The Parthenos European Project. Project web site 2016. `www.parthenos-project.eu`.
110. The SoBigDataeu European Project. Project web site 2016. `www.sobigdata.eu`.
111. Lasi H, Fettke P, Kemper HG, Feld T, Hoffmann M. Industry 4.0. *Business & Information Systems Engineering* 2014; **6**(4):239.
112. National Research Council of Italy. The gCube DataMiner integrated processes 2016. `https://wiki.gcube-system.org/gcube/DataMiner_Algorithms`.

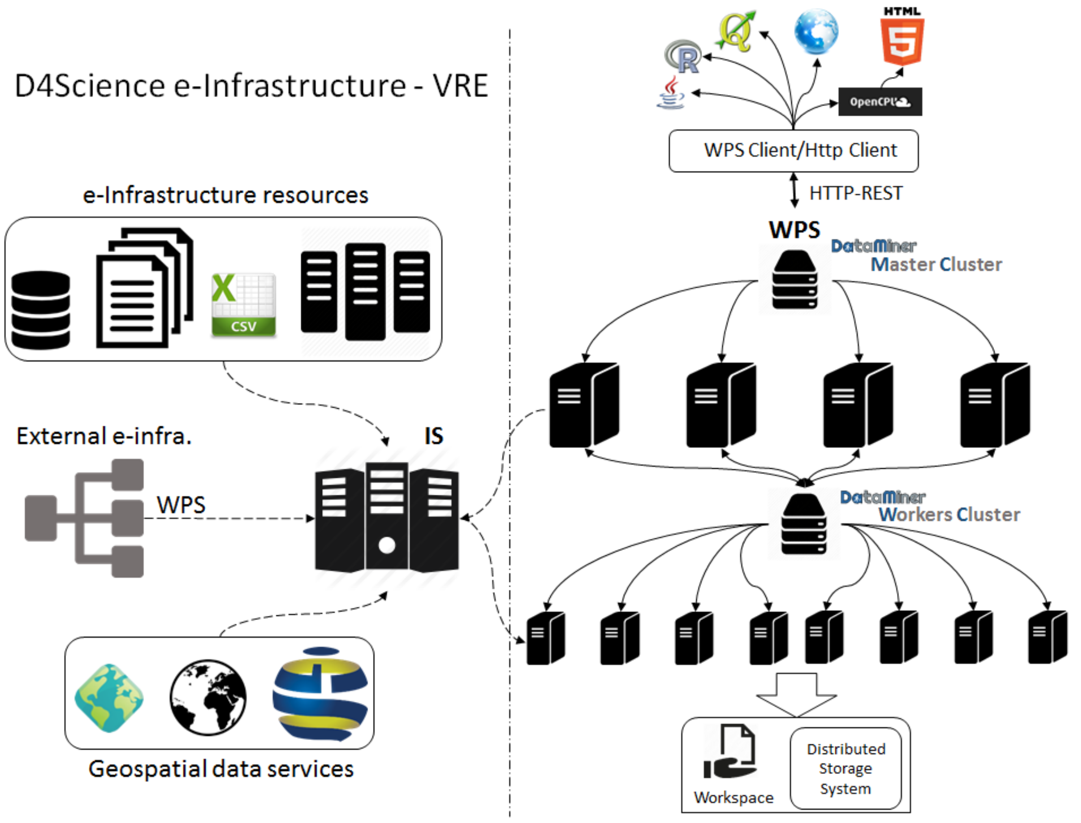


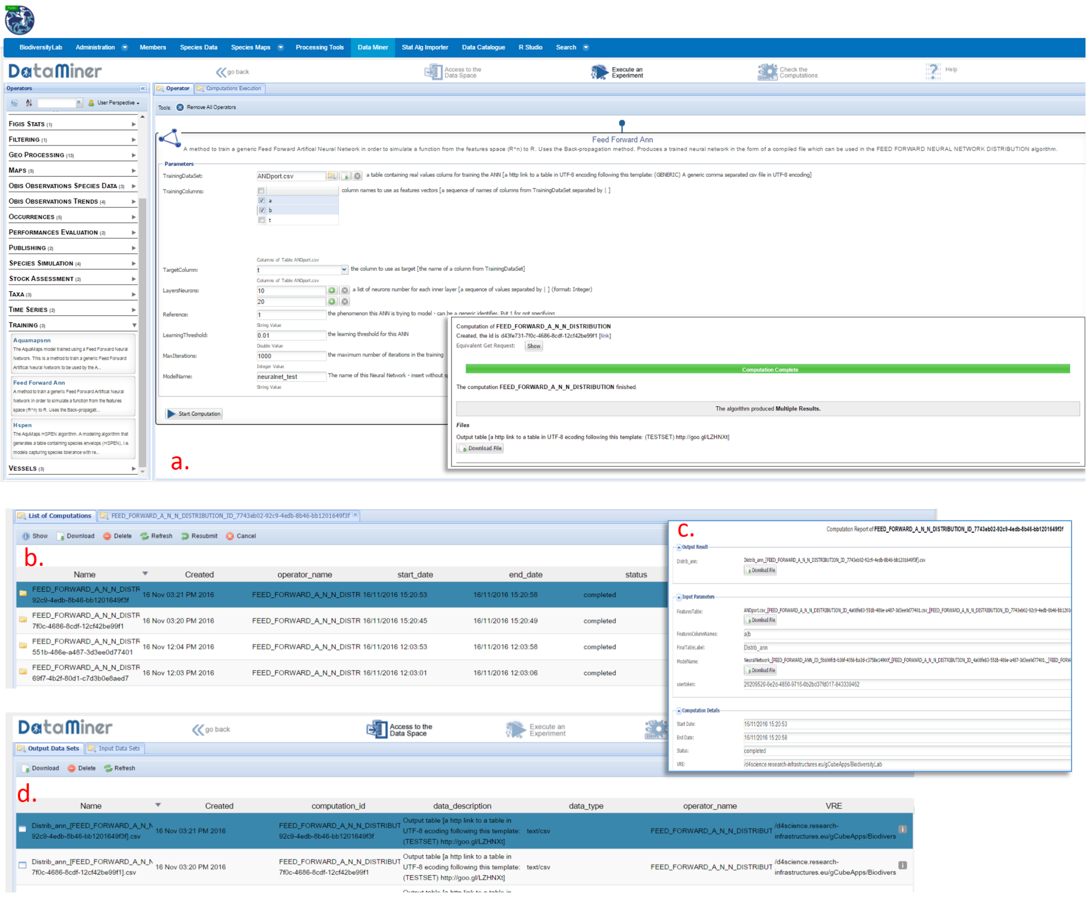
Figure 1. Architecture of the gCube DataMiner system. On the right side, the architecture of the computational system is reported, with two clusters of machines used for Local and Cloud computations respectively. Also, a number of clients is indicated that are able to natively interact with the system, i.e. Java, R, QGis, Web browser, HTML pages through an OpenCPU service instance. On the left side, e-Infrastructure resources are reported, which are indexed on the Information System (IS) along with the DataMiner services and other possible external WPS services. All the DataMiner services are able to write data on a Distributed Storage System and to organize/share the data through the D4Science e-Infrastructure Workspace.

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ExecuteResponse xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:xsi="http://www.w3.org
     /2001/XMLSchema-instance" xmlns:ows="http://www.opengis.net/ows/1.1" xsi:schemaLocation="
     http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute_response
     .xsd" serviceInstance="http://dataminer2-p-d4s.d4science.org:80/wps/WebProcessingService"
      xml:lang="en-US" service="WPS" version="1.0.0">
  <wps:Process wps:processVersion="1.1.0">
    <ows:Identifier>org.gcube.dataanalysis.wps.statisticalmanager.synchserver.mappedclasses.
         generators.FEED_FORWARD_A_N_N_DISTRIBUTION</ows:Identifier>
    <ows:Title>FEED_FORWARD_A_N_N_DISTRIBUTION</ows:Title>
  </wps:Process>
  <wps:Status creationTime="2016-11-16T15:20:53.965+01:00">
    <wps:ProcessSucceeded>Process successful</wps:ProcessSucceeded>
  </wps:Status>
  <wps:ProcessOutputs>
    <wps:Output>
      <ows:Identifier>OutputTable</ows:Identifier>
      <ows:Title>Output table [a http link to a table in UTF-8 ecoding following this
           template: (TESTSET) http://goo.gl/LZHNXt]</ows:Title>
      <wps:Data>
        <wps:ComplexData mimeType="text/csv">
      a,b,tvalue
      1,0,0.06122341
      0,1,0.056781072
      0,0,0.00084975874
      1,1,0.96765828
      </wps:ComplexData>
      </wps:Data>
    </wps:Output>
    <wps:Output>
      <ows:Identifier>non_deterministic_output</ows:Identifier>
      <ows:Title>NonDeterministicOutput</ows:Title>
      <wps:Data>
        <wps:ComplexData schema="http://schemas.opengis.net/gml/2.1.2/feature.xsd" mimeType="
             text/xml; subtype=gml/2.1.2">
          <ogr:FeatureCollection xmlns:ogr="http://ogr.maptools.org/" xmlns:gml="http://www.
               opengis.net/gml" xmlns:d4science="http://www.d4science.org" xsi:schemaLocation=
               "http://ogr.maptools.org/ result_8751.xsd">
            <gml:featureMember>
              <ogr:Result fid="F0">
                <d4science:Data>http://data.d4science.org/
                     RlNPc1lKS3R6ME5GZXRTa2prN1d2UnZRNmYrNzltNTJHbWJQNStIS0N6Yz0-VLT</
                     d4science:Data>
                <d4science:Description>Output table [a http link to a table in UTF-8 ecoding
                     following this template: (TESTSET) http://goo.gl/LZHNXt]</
                     d4science:Description>
                <d4science:MimeType>text/csv</d4science:MimeType>
              </ogr:Result>
            </gml:featureMember>
          </ogr:FeatureCollection>
        </wps:ComplexData>
      </wps:Data>
    </wps:Output>
  </wps:ProcessOutputs>
</wps:ExecuteResponse>
```

Figure 2. Output of a DataMiner process that reports the results of an Artificial Neural Network projection of the inputs (a,b) of an AND port. The results are reported both as an in-line WPS complex data and as a public HTTP link pointing to the D4Science Distributed Storage System and embedded in a GML document.

```
<?xml version="1.0" encoding="UTF-8"?><prov:document xmlns:d4s="http://d4science.org/#"
     xmlns:prov=http://www.w3.org/ns/prov#
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance">
   <prov:activity prov:id="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER_ID_ba09841f-6656-4f6d-b6f1-
       b8fd532d32e9">
      <prov:startTime>11/11/2016 15:05:53</prov:startTime>
      <prov:endTime>11/11/2016 15:05:55</prov:endTime>
      <prov:type xsi:type="xsd:QName">d4s:computation</prov:type>
      <prov:softwareAgent prov:id="d4s:dataminer.d4science.org"/>
      <prov:person prov:id="d4s:gianpaolo.coro"/>
      <prov:entity prov:id="d4s:operator_name">
         <prov:value xsi:type="xsd:string">FEED_FORWARD_NEURAL_NETWORK_TRAINER</prov:value>
      </prov:entity>
      <prov:entity prov:id="d4s:operator_id">
         <prov:value xsi:type="xsd:string">org.gcube.dataanalysis.wps.statisticalmanager.
              synchserver.mappedclasses.
      transducerers.FEED_FORWARD_NEURAL_NETWORK_TRAINER</prov:value>
      </prov:entity>
      <prov:entity prov:id="d4s:operator_description">
         <prov:value xsi:type="xsd:string">The algorithm trains a Feed Forward Artificial
              Neural Network using an online Back-Propagation
    procedure and returns the training error and a binary file containing the trained network<
         /prov:value>
      </prov:entity>
      <prov:entity prov:id="d4s:VRE">
         <prov:value xsi:type="xsd:string">/d4science.research-infrastructures.eu/gCubeApps/
              RPrototypingLab</prov:value>
      </prov:entity>
      <prov:entity prov:id="d4s:status">
         <prov:value xsi:type="xsd:string">100</prov:value>
      </prov:entity>
      <prov:entity prov:id="d4s:inputTable_[FEED_FORWARD_NEURAL_NETWORK_TRAINER_ID_ba09841f
           -6656-4f6d-b6f1-b8fd532d32e9].csv">
         <prov:value xsi:type="xsd:string">http://data.d4science.org/
              Tm5UMkwyellGY0tEa2RCMXpYd3JUZmdyWXYyb2NubjdHbWJQNStIS0N6Yz0</prov:value>
         <prov:activity prov:ref="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER_ID_ba09841f-6656-4
              f6d-b6f1-b8fd532d32e9"/>
         <prov:entity prov:ref="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER"/>
         <prov:entity prov:id="d4s:data_description">
            <prov:value xsi:type="xsd:string">inputTable</prov:value>
         </prov:entity>
         <prov:type xsi:type="xsd:QName">d4s:IMPORTED</prov:type>
         <prov:type xsi:type="xsd:QName">d4s:text/csv</prov:type>
      </prov:entity>
      <prov:entity prov:id="d4s:learningRate">
         <prov:value xsi:type="xsd:string">0.5</prov:value>
         <prov:activity prov:ref="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER_ID_ba09841f-6656-4
              f6d-b6f1-b8fd532d32e9"/>
         <prov:entity prov:ref="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER"/>
         <prov:entity prov:id="d4s:data_description">
            <prov:value xsi:type="xsd:string">learningRate</prov:value>
         </prov:entity>
      </prov:entity>
      <prov:entity prov:id="d4s:TrainedNeuralNetwork">
         <prov:value xsi:type="xsd:string">http://data.d4science.org/
              STlYeDRwWDFIYnlEa2RCMXpYd3JUY0JNWHZNTmN5VDlHbWJQNStIS0N6Yz0</prov:value>
         <prov:activity prov:ref="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER_ID_ba09841f-6656-4
              f6d-b6f1-b8fd532d32e9"/>
         <prov:entity prov:ref="d4s:FEED_FORWARD_NEURAL_NETWORK_TRAINER"/>
         <prov:entity prov:id="d4s:data_description">
            <prov:value xsi:type="xsd:string">Trained Neural Network</prov:value>
         </prov:entity>
         <prov:type xsi:type="xsd:QName">d4s:COMPUTED</prov:type>
         <prov:type xsi:type="xsd:QName">d4s:application/d4science</prov:type>
      </prov:entity>
   </prov:activity>
</prov:document>
```

Figure 3. Provenance of a DataMiner process that trained an Artificial Neural Network. The computation is modelled as an "activity" and "entities" are associated to inputs and output. The "entity" of the complex input (QName equal to d4s:IMPORTED) and output (QName equal to d4s:COMPUTED) have internal links to the "activity" so that they can be separated from the provenance document while keeping a reference to the computation that either used or produced them.



Aquamapsnn
The AquaMaps model trained using a Feed Forward Neural Network. This is a method to train a generic Feed Forward Artifical Neural Network to be used by the A...

Feed Forward Ann
A method to train a generic Feed Forward Artifical Neural Network in order to simulate a function from the features space (R^n) to R. Uses the Back-propagati...

Hspen
The AquMaps HSPEN algorithm. A modeling algorithm that generates a table containing species envelops (HSPEN), i.e. models capturing species tolerance with re...

Feed Forward Ann
A method to train a generic Feed Forward Artifical Neural Network in order to simulate a function from the features space (R^n) to R. Uses the Back-propagation method. Produces a trained neural network in the form of a compiled file which can be used in the FEED FORWARD NEURAL NETWORK DISTRIBUTION algorithm.

Parameters

TrainingDataSet:	ANDport.csv	a table containing real values colums for training the ANN [a http link to a table in UTF-8 encoding following this template: (GENERIC) A generic comma separated csv file in UTF-8 encoding]
TrainingColumns:	☐ ☑ a ☑ b ☐ t	column names to use as features vectors [a sequence of names of columns from TrainingDataSet separated by |]
TargetColumn:	Columns of Table ANDport.csv t	the column to use as target [the name of a column from TrainingDataSet]
LayersNeurons:	Columns of Table ANDport.csv 10 20	a list of neurons number for each inner layer [a sequence of values separated by |] (format: Integer)
Reference:	1 String Value	the phenomenon this ANN is trying to model - can be a generic identifier. Put 1 for not specifying
LearningThreshold:	0.01 Double Value	the learning threshold for this ANN
MaxIterations:	1000 Integer Value	the maximum number of iterations in the training
ModelName:	neuralnet_test String Value	The name of this Neural Network - insert without

Computation of FEED_FORWARD_A_N_N_DISTRIBUTION
Created, the id is d43fe731-7f0c-4686-8cdf-12cf42be99f1 [link]
Equivalent Get Request: Show

Computation Complete

The computation FEED_FORWARD_A_N_N_DISTRIBUTION finished.

The algorithm produced **Multiple Results.**

Files

Output table [a http link to a table in UTF-8 ecoding following this template: (TESTSET) http://goo.gl/LZHNXt]
Download File

List of Computations | FEED_FORWARD_A_N_N_DISTRIBUTION_ID_7743eb02-92c9-4edb-8b46-bb1201649f3f

Name	Created	operator_name	start_date	end_date	status
FEED_FORWARD_A_N_N_DISTR 92c9-4edb-8b46-bb1201649f3f	16 Nov 03:21 PM 2016	FEED_FORWARD_A_N_N_DISTR	16/11/2016 15:20:53	16/11/2016 15:20:58	completed
FEED_FORWARD_A_N_N_DISTR 7f0c-4686-8cdf-12cf42be99f1	16 Nov 03:20 PM 2016	FEED_FORWARD_A_N_N_DISTR	16/11/2016 15:20:45	16/11/2016 15:20:49	completed
FEED_FORWARD_A_N_N_DISTR 551b-486e-a487-3d3ee0d77401	16 Nov 12:04 PM 2016	FEED_FORWARD_A_N_N_DISTR	16/11/2016 12:03:53	16/11/2016 12:03:58	completed
FEED_FORWARD_A_N_N_DISTR 69f7-4b2f-80d1-c7d3b0e8aed7	16 Nov 12:03 PM 2016	FEED_FORWARD_A_N_N_DISTR	16/11/2016 12:03:01	16/11/2016 12:03:06	completed

Computation Report of **FEED_FORWARD_A_N_N_DISTRIBUTION_ID_7743eb02-92c9-4edb-8b46-bb1201649f3f**

Output Result

Distrib_ann:	Distrib_ann_[FEED_FORWARD_A_N_N_DISTRIBUTION_ID_7743eb02-92c9-4edb-8b46-bb1201649f3f].csv Download File

Input Parameters

FeaturesTable:	ANDport.csv_[FEED_FORWARD_A_N_N_DISTRIBUTION_ID_4a08fe83-551b-486e-a487-3d3ee0d77401.csv_[FEED_FORWARD_A_N_N_DISTRIBUTION_ID_7743eb02-92c9-4edb-8b46-bb120 Download File
FeaturesColumnNames:	a|b
FinalTableLabel:	Distrib_ann
ModelName:	NeuralNetwork_[FEED_FORWARD_ANN_ID_5b500fcb-b36f-4056-ba3d-c3758e14900f_[FEED_FORWARD_A_N_N_DISTRIBUTION_ID_4a08fe83-551b-486e-a487-3d3ee0d77401_[FEED_FORW Download File
usertoken:	26209520-6e2d-4850-9716-0b2bd37fd017-843339462

Computation Details

Start Date:	16/11/2016 15:20:53
End Date:	16/11/2016 15:20:58
Status:	completed
VRE:	/d4science.research-infrastructures.eu/gCubeApps/BiodiversityLab

Name	Created	computation_id	data_description	data_type	operator_name	VRE
Distrib_ann_[FEED_FORWARD_A_N_N 92c9-4edb-8b46-bb1201649f3f].csv	16 Nov 03:21 PM 2016	FEED_FORWARD_A_N_N_DISTRIBUT 92c9-4edb-8b46-bb1201649f3f	Output table [a http link to a table in UTF-8 ecoding following this template: (TESTSET) http://goo.gl/LZHNXt]	text/csv	FEED_FORWARD_A_N_N_DISTRIBUT	/d4science.research-infrastructures.eu/gCubeApps/Biodivers
Distrib_ann_[FEED_FORWARD_A_N_N 7f0c-4686-8cdf-12cf42be99f1].csv	16 Nov 03:20 PM 2016	FEED_FORWARD_A_N_N_DISTRIBUT 7f0c-4686-8cdf-12cf42be99f1	Output table [a http link to a table in UTF-8 ecoding following this template: (TESTSET) http://goo.gl/LZHNXt]	text/csv	FEED_FORWARD_A_N_N_DISTRIBUT	/d4science.research-infrastructures.eu/gCubeApps/Biodivers

Figure 4. Web interface of the gCube DataMiner system in a D4Science VRE (BiodiversityLab): (a) a training algorithm for an Artificial Neural Network is displayed; after the computation, the result (a trained network binary file) is reported under the completion bar, for downloading; (b) in the "Check the Computations" panel, the list of executed computations is displayed with the associated metadata; (c) by pressing the "Show" button, the metadata associated to a computation are reported in a user-friendly structured way, where inputs and outputs can be downloaded; (d) the "Access to the Data Space" panel allows retrieving the overall list of input and output data used in the computations, with medatata pointing to the associated computation.

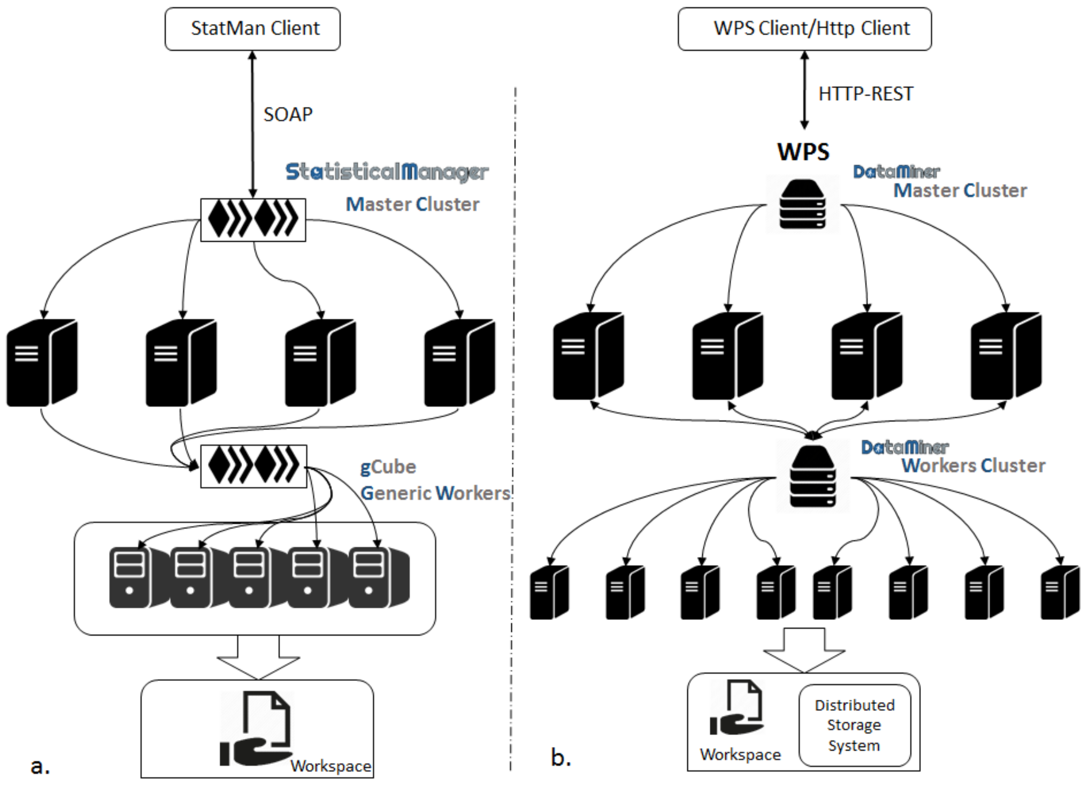


Figure 5. Comparison of the architectures of the (a) D4Science Statistical Manager (StatMan) and (b) DataMiner.

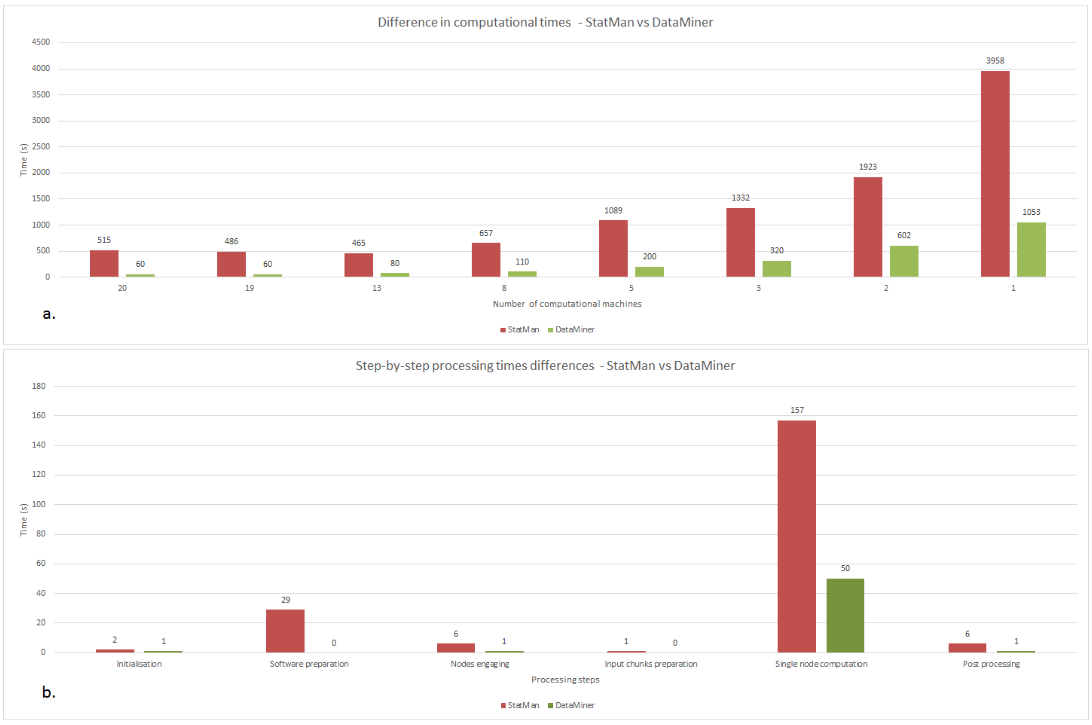


Figure 6. Difference in the performance of the same process (BiOnym [99], applied to 1024 species scientific names) running on DataMiner with respect to the Statistical Manager reported (a) at the variation of the number of a computational nodes and (b) per step of the computational process.