Parallelising the Execution of Native Data Mining Algorithms for Computational Biology

Gianpaolo Coro*, Leonardo Candela, Pasquale Pagano,
Angela Italiano, Loredana Liccardo

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

SUMMARY

Data mining is more and more used in biology. Biologists adopt prototyping languages, like R and Matlab, to facilitate the application of data mining algorithms to their data. A side effect is that their scripts become more and more complex and also require frequent updates. Application to large datasets becomes impractical and the time-to-paper increases. On the other side, high performance computing systems usually require procedures to be translated into specific languages or adapted to a certain computing platform. Such modifications account for speeding up the processing. The translation is not automatic, especially in complex cases and can require large programming effort and accurate validation. In this paper, we propose an approach to parallelise data mining procedures coming in the form of compiled software or R scripts developed by biologists' communities of practice. Our approach is not invasive and possibly does not alter the original code. Furthermore, it allows for fast updating when a new version is ready. We clarify the constraints and the benefits of our method and report a practical use case to demonstrate such benefits with respect to a standard execution. Our approach relies on a distributed network of web services and eventually exposes the algorithms as-a-Service, to be invoked by remote thin clients.

Received . . .

KEY WORDS: Data Mining, Parallel Processing, Cloud Computing, Computational Biology, Distributed Systems, R

1. INTRODUCTION

Data mining techniques have become fundamental in computational biology. Applications in this field range from predicting the effect of climate change on species distributions [1, 2, 3], to the investigation about fundamental traits in life history [4, 5, 6]. Machine learning techniques allow the extraction of relationships from observed phenomena and help biologists in overcoming the limitations imposed by manual analysis. Also, biologists have learned to develop prototypes in scripting languages like R and Matlab, which provide a large set of *precooked* state-of-the art techniques as well as a *lightweight* programming style. One side effect of this, is that several issues arise in terms of (*i*) managing scripts updates and ownership, (*ii*) applying parallel processing techniques, (*iii*) managing several interpreters versions and execution environments, (*iv*) overcoming licensing issues for not free of use languages. More in details, prototypes usually pay poor attention to the efficiency of the computation, which ends in impractical applications to large datasets. Furthermore, scripts developed by biologists need frequent updates. Biologists would also

*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

like to keep the ownership and the control on every step made by the procedures. This scenario makes it difficult to communicate with the world of Cloud Computing, which usually needs the scripts to be adapted or translated into some high efficiency language. Furthermore, automatic translation is not effective in the case of complex scripts. Aside of efficiency issues, other difficulties arise while meeting biologists' requirements. We classify such difficulties in (*i*) issues related to the usage of scripting languages and (*ii*) issues related to the usage of compiled programs. Scripting languages are either not free of use (e.g. Matlab) or are strongly dependent on the version of the interpreter (e.g. the R scripting language). If a service or an integrated development environment (IDE) has to execute R scripts developed by multiple communities of practice, it shall manage issues related to the interpreters versions and to the Operating Systems in which the scripts have been developed. It is very difficult to impose the usage of one single R interpreter version to all the developers. On the other side, downgrading a complex script to an earlier version of the R interpreter, requires the script to use earlier versions of the depending packages. For several data mining libraries this could end in machine learning models that do not converge, given the same input data and parameters. In other words, an R script that was tested against a certain version of the R interpreter, should be run using the same version. This consideration applies also to many other scripting languages. For compiled programs, instead, the requirement is more strict, because these need specific hardware to be properly executed. Without having access to the source code, one possible way to parallelise the procedure is to execute the same code on different disjoint subsets of the input space [7]. In this case, the compiled program is executed simultaneously on portions of the input space, where the union of these portions is the input space itself. These disjoint subsets shall be prepared before executing the parallel processes. This approach imposes constraints on the classes of algorithms that can be brought on a Cloud computing platform without modifying the source code. In our experience on the processes for marine species [**?**, 8] as well as for plants [9], we found that many of these processes satisfy such constraints. Such processes usually require the application of a sequence of data mining algorithms, each of which can be parallelised on the input space in the most cases.

Complex processes can be built as workflows using these algorithms in cascade. For example, the generation of a distribution map by means of the niche model of a species [10] usually requires (*i*) a data preparation phase, (*ii*) a probability calculation phase and (*iii*) a map generation process. In the first phase, tables are produced on the basis of environmental information collected at locations where the species presence/absence assessment was performed. In the second phase, the model calculates a presence probability value for each location, according to its environmental characteristics. The locations are usually analysed independently on the others. Finally, a map is produced by transforming each assessed location into a GIS geometry. Each of the steps in this specific workflow is a process that can be parallelised on the number of locations to be assessed, because these are treated as independent elements. Other examples of workflows that can be parallelised, are those processes that search in taxa names repositories [11], where a large number of scientific names must be checked to be possible correct transcriptions for an input scientific name.

In this paper, we present a system (*gCube Statistical Manager* or simply *Statistical Manager*) that meets the requirements of this scenario. For both R scripts and compiled programs, it executes the original software coming from biologists (native programs) in parallel fashion. In the case of R scripts, the system selects the most appropriate interpreter. In the case of compiled Java or C programs, it selects the best platform to execute them, among the available ones in a Cloud computing infrastructure. Thus, it allows the reuse of native programs just in the environments where they have been developed and tested. Our system comes with a development framework that allows one to distribute and parallelise the execution of programs in an easy way. The Statistical Manager is based on a network of web services and on a Cloud computing architecture. This system eventually supplies the integrated algorithms as-a-Service, for use by thin clients.

The paper is organized as follows. Section 2 gives an overview of multi-user systems that apply data mining to biology datasets using parallel processing. It also reports on systems for developing prototypes and on platforms supporting the integration of multiple programming languages. The section highlights the complementarity between our system and other ones. Section 3 describes

our platform and the development framework we provide to enable parallel processing on native programs, along with their limitations and benefits. Section 4 reports a practical use case, on the parallelisation of an R script to estimate the relationships between the lengths and weights of a set of marine species. It demonstrates the practical benefits of our solution in this case. Finally, Section 5 draws the conclusions.

2. OVERVIEW

In this section, we report an overview of systems that address the parallelisation of procedures developed by a certain community of practice. First, we mention approaches aiming at enhancing R scripts processing. Then we present more general purpose approaches that manage the execution of software written in several programming languages (including R), along with the issues they experienced. Further, we report about platforms enabling Cloud and Grid computing for algorithms developed by heterogeneous communities of practice, with a particular attention to those accepting also R scripts. We also mention solutions for building complex algorithms that can combine parallel processing algorithms. Finally, we clarify the complementary aspects of our system with respect to the reported ones.

Most approaches to the parallelisation of procedures developed by communities of practice, face the problem from the point of view of the communities themselves. They usually try to approach the problem in the minimal invasive way. This is the case, for example, of R libraries that enable parallel processing. The work by Schmidberger et al. [12] reports several approaches in this direction. The authors highlight the increasing interest around high performance computing with R, especially in Bioinformatics. The most reported approaches use wrappers for running scripts on parallel processing infrastructures [13, 14]. Schmidberger et al. also report techniques to improve R code. These involve either optimizing code efficiency [15] or making the scripts interact with (or be executed on) external Grid or Cluster computing systems compliant with R [16, 17, 18]. In particular, R scripts can invoke external computing systems to parallelise some steps of a process, otherwise the scripts can be entirely executed on a Grid or on a Cloud computing system. However, these solutions are usually able to execute R scripts only and cannot be used for programs developed with other programming languages. Furthermore they are tight to a certain version of the R interpreter.

A more general purpose solution is reported in [19]. The authors describe the MGET multi-platform and framework. This is a Python-based set of tools to integrate software written in several languages, for applications to marine and geo-spatial processing. The main aim of this system is to reduce the effort and knowledge required to a biologist to integrate tools that account for geo-spatial processing and data mining. On the other side, MGET presents several limitations. It is a desktop software that is not natively integrated with parallel processing platforms. Furthermore, the authors report that developers experienced difficulties in running non-Python software, especially in the case of R. In particular, difficulties were naturally found in managing different versions of the R interpreter and of the packages linked by the scripts. The authors were forced to translate most R scripts into Python, to achieve full working integration.

Many works highlight the importance to integrate R scripts, even by non R-native platforms. This is due to the many packages developed for R, and many communities of users adopt this language for prototyping. Platforms integrating multiple programming languages are present too. One example is RapidMiner [20], which embeds R as well as external programs in a desktop platform.

Several platforms are also available to make communities of practice use Cloud or Grid computing in an easy way. For example, Yabi [21] is a Python based architecture that is able to plug-in execution and data backends. It enables seamless and transparent access to heterogeneous HPC environments. It also provides workflows managements tools. Yabi also pays attention to usability issues and is strongly oriented to users which are not familiar with Grid or Cloud computing setup. Usability is a key feature for attracting communities of practice.

Systems addressing easy parallelisation of software adopt various approaches. For example, Apache Hadoop [22] is able to distribute external code onto several nodes. The parallelisation

adopts a Map-Reduce approach. The input space is usually split into subsets and these are passed to computational nodes. However, using Hadoop requires experience in setting up the Map-Reduce process. Furthermore, it does not account for the suitability of the platform to the native software by default. For example, if the Hadoop nodes run an R 2.14.0 interpreter, they will not likely be able to execute scripts developed with R 3.0.0. This coherence check is not natively managed by Hadoop. In a similar way, other cloud computing systems exist, which support software parallelisation with minimal code adaptation [23]. However, they usually require that system integrators understand the contents of the native code or even to translate it.

Other systems, like Workflow Management Systems (WMSs), are able to combine algorithms into workflows in a flexible way. WMSs can be a valid solution to manage the complex algorithms mentioned in the previous section. They strongly separate algorithms invocations from the interaction among these algorithms. WMSs allow users with basic programming experience to combine algorithms and perform complex analyses. The algorithms are the atomic steps of the workflows and can be executed by remote systems, possibly using parallel processing. Formal definitions of input and output types and other metadata allow users to understand and reuse algorithms in other workflows. Examples of WMSs with applications to computational biology are Taverna [24] and Galaxy [25].

Our proposal presents complementary aspects with respect to the above systems. It adopts the point of view of the communities of practice. In the same way as Hadoop, our system parallelises the process on the input space using a Map-Reduce approach. However, we spent effort in making this as easy as possible. We provide simple procedures to alter the native software, on-the-fly and during a computation, when necessary and possible. In the case a script had to be altered, our system provides basic methods to inject code, in order to make it run on a subset of the input data. Furthermore, our system is Java-based and can be installed on several operating systems. This means that the choice of the platform on which the user's software will be delivered is flexible. As explained in the next section, we introduce a mechanism to select the most suitable machine, given standardized specifications about the native software to execute. For example, this allows R scripts to be executed by a suitable interpreter. In a similar way to Yabi, our system distributes the native software that has been integrated as-a-Service, by means of a distributed network of web services. We paid attention to the usability of our system when building a web-based user interface for non-programmers. This user interface is self maintained and dynamically generated (cf. Fig. 2). Furthermore, we provide a thin client and a SOAP [26] interface for integrations by external programs. In particular, the SOAP protocol [27] allows a client and a web server to exchange structured information and to embed XML documents containing structured objects and callable methods definitions. SOAP is a common choice for a protocol to interact with web services and can be used by different transport protocols.

Last but not least, the Statistical Manager is a distinguishing and integral component of the gCube Software System [28, 29]. gCube enables to build and operate Hybrid Data Infrastructures [30] enabling Virtual Research Environments (VREs) [31], the Statistical Manager permit to add an open and configurable data analytic environment to every VRE. These are the main novelties of our approach, which are detailed in Section 3.

3. METHOD

In this section we describe the details of our solution. In Section 3.1 we explain the system architecture based on a network of web services. In Section 3.2, we clarify how to expand the system capabilities by introducing new algorithms. Section 3.3 describes how the platform enables parallel processing for native language algorithms, and how it accounts for matching the most appropriate running platform. Section 3.4 explains how the system provides algorithms as-a-Service. Finally, Section 3.5 clarifies the limitations and the benefits of our solution.

3.1. System architecture

The Statistical Manager (SM) architecture was formerly introduced in [7] as a Cloud computing system and here we present an enhanced version. It is depicted in Figure 1.

The SM is conceived to work as a dynamic cluster of replicated SM service instances, i.e. new SM instances can be added to the cluster on the fly and all the instances have equal capabilities in terms of hosted algorithms. Clients interact with the SM cluster via a message queue and each message (e.g. retrieving the list of algorithms or the computations products) is managed by the service instance that is not busy. Request for computing tasks can be executed locally to the SM instance or dispatched to a distributed computing platform. An Information System (IS) is used to provide the service instances forming the cluster with up to date information about their siblings presence and status as well as about the rest of services forming the computational platform and the underlying infrastructure. From the SM perspective, the IS stores information about (*i*) the IP addresses of the services, (*ii*) their roles in the computations, (*iii*) the computational environment offered by the respective machines. More in details, some machines are purely *Worker* nodes, which only execute programs as external programs. They can lay in a gCube based computational e-Infrastructure, otherwise they can be either Hadoop or Microsoft Windows Azure [32] nodes. *The nodes can be endowed with different Operating Systems, R interpreters and execution environments for software.*

With respect to the system in [7], we added the possibility for the IS to be aware of the operating system and of all the software running on the machines. On top of this, we built a matching mechanism that is able to supply a program with the appropriate platform to be executed on. Such information is specified on the IS by the system administrator, when indexing a new available node. Information includes (*i*) the version of the R interpreter running on the machine, (*ii*) the version of the Java Runtime Environment, (*iii*) the machine architecture and the installed operating system. The system administrator specifies such characteristics on the basis of a common vocabulary defined in the framework specifications [33].

The SM adopts a Map-Reduce approach for computing. A queue-based messaging system dispatches information about the computation to the nodes. Such information is made up of a sequence of messages, each including the following: (*i*) the location of an accessible storage area containing the software to execute, (*ii*) indication on how to take a chunk of the input data space, i.e. the portion of the input to process, (*iii*) an accessible location for the data to process, (*iv*) the algorithm name and its parameters. The software and the data are then downloaded by each node. Workers are data and software agnostic, which means that when ready to perform a task, they consume information from the queue and execute the software in a sandbox, passing the experimental parameters as input. This allows to manage security and memory issues. Eventually, the outputs of the algorithms are stored as tables or files by relying on the shared storage facilities. One PostgreSQL database, shared among the services, is used to store low-size tables. For large datasets and for all kinds of files, a distributed storage system based on MongoDB [34] is used, which ensures high availability and scalability.

During the computation not all the workers are employed. As explained in the next section, when a new algorithm is deployed, the developer specifies the kind of execution environment on which it can run. This information can include (*i*) a specific version of the R interpreter, (*ii*) a compatible version of the Java Runtime Environment, (*iii*) a compatible computer architecture and operative system on which the program can run. When the Statistical Manager starts the processing phase, it first prepares the software to be distributed to the nodes and the information required to build the computation messages. Later, it asks the IS for a list of workers supporting a suitable platform for the algorithm. If there are not suitable Worker nodes available, the SM does not execute the computation and alerts the client about a lack of resources. Otherwise, it accepts the request and attaches platform indications to the messages in the execution queue. The Worker nodes consume the queue messages, check if they meet the indicated platform requirements and possibly start the processing. At the end of the computation, the output is stored either on the central PostgresSQL instance (in the case of low-size tables) or on MongoDB (for files and large datasets).

Our system implements a Map-Reduce approach. It can typically manage cases in which one algorithm has to process the product of two data matrices. The SM assumes that it is possible to parallelise the computation by dividing one of the matrices into chunks. This allows the reuse of original programs in the environments where they were developed and tested. Thus, parallelisation occurs at the level of the input space. If the algorithm can be natively applied to a subset of the input space, the benefits of this approach are straightforward. Otherwise, code injection methods shall be applied when source code is available. This comes as part of the SM development framework, as explained in Section 3.3.

3.2. Adding new algorithms

The SM allows to add new algorithms as plugins. One enhancement with respect to the system described in [7], is that these algorithms can be also non-Java programs. The system comes with a development framework which is part of the gCube framework [28]. A software developer who wanted to contribute to extend the service capabilities, would have to develop a Java class implementing one basic Java Interface. For a new algorithm, this defines its inputs and outputs as well as its platform requirements.

In the case of non-Java software, e.g. R scripts or C programs, the Java class acts as a wrapper. In this case, the framework helps developers by providing functionalities to invoke external programs, with algorithms parameters passed as inputs. The wrapper is the bridge between our framework and the native program. The developer must invoke the external program indicating the location of the input data. As explained before, the SM and the Worker nodes are responsible for materializing the data in the folder where the wrapper and the native software will be executed. During the computation, the wrapper will be delivered to the nodes along with the native program and with a subset of the input space.

To help the SM in distributing the computation, three indications are essential when implementing the wrapper: (*i*) the wrapper must specify which are the inputs and the outputs of the algorithm, according to a set of predefined input types, (*ii*) it must specify which input data matrix the SM has to divide into chunks, (*iii*) it must include suitable running platform indications, according to a controlled vocabulary. The Java Interface to implement suggests methods to provide such information to the SM. The Interface also forces to specify what happens in the Reduce phase of the process. The wrapper requires users to specify a method named *postProcess*, in which the output by the computing nodes can be merged or revised, for example to delete duplicates in a table. The Statistical Manager framework allows to retrieve files possibly produced by the Worker nodes, using the D4Science e-Infrastructure facilities. In particular, the distributed storage system mentioned in the previous section also acts as a shared file system for output files. Otherwise, an algorithm can retrieve connection parameters to the shared PostgresSQL database, in order to access an output table.

From the point of view of the SM, and in very general terms, an algorithm can fall in one of 3 cases: non-parallelisable, parallelisable and parallelisable after modifications. In Section 3.3, we explain how an algorithm is managed when it falls in one of the two last cases. If the algorithm cannot be parallelised, the SM system can be forced to execute it on one single machine. As explained in Section 3.3, one limitation of our approach is that the user shall know *a priori* if the algorithm can be parallelised by means of a Map-Reduce approach. On the other hand, the SM framework is endowed with a set of tools that help the user in such task. During the development phase, the user is alerted about the possible contraindication of forcing parallel processing on a sequential process [33]. Furthermore, the Statistical Manager framework is endowed with a local testing suite that allows users to verify the correctness of the output of the algorithm running on the Cloud computing system [35]. This local testing suite is able to distribute the algorithm from a local-computer development environment, based on the Eclipse IDE, onto a set of testing machines. The testing machines are currently part of the D4Science e-Infrastructure [36]. This preliminary testing phase is important for the developer to understand if the output of the parallel process is the same as the one of the sequential execution. Afterwards, since the SM is part of the D4Science e-Infrastructure, the algorithm publication process shall pass through the D4Science deployment

workflow [29, 37] in order to be published through the e-Infrastructure. This software engineering approach relies on the Etics [38] automatic integration, building and testing system. The Etics instance running on D4Science checks if the algorithm and its dependencies are compliant with the e-Infrastructure services that will execute the process. Furthermore, automatic testing can be invoked through Etics to check for the coherence of the output. These *a posteriori* tests can be developed in Java language and can be specified in the Etics building configuration. They are executed at the end of the building and integration phase. The D4Science deployment workflow also contemplates a pre-production user testing phase. In this phase, the developer of the algorithm can verify that the integration works and that the outputs are coherent with the expected ones.

3.3. Parallelisation of native software

Following the indications in Section 3.2, a Java wrapper must be developed to pass input parameters to native programs. A software developer is helped by the SM framework tools, which ensure files and parameters to be available in the folder where the program will run. We are assuming that the algorithm processes a matrix product of elements, which is common in Map-Reduce systems. If the program is yet able to accept a subset of the input space, building the wrapper is straightforward. Otherwise, the developer can either decide not to subdivide the input space or to inject code into the native program if possible. In the former case, the process will be managed by only one Worker node entirely. In the latter case, the SM framework provides simple facilities to overwrite code lines inside scripts, by performing strings substitutions. In Section 4, we show one example in which we inject an R line into a script, to make it process only a subset of the input data. As stated in Section 3.1, the SM framework endows users with a set of testing tools along with a sophisticated deployment phase, in order to understand if the parallelises version of the algorithm produces the same results as the original sequential version.

The platform selection system of the SM allows to detect the best platform to run the script, among the available ones. For what regards compiled programs, the system selects only machine architectures which can support the program. We remind that such compliance is specified by the Java wrapper on the algorithm side, and by the Information System on the Worker nodes side. For Java programs, only Worker nodes supporting a runtime environment equal or higher to the one in which the algorithm was compiled, will be selected. For R scripts, instead, having a full range of interpreters would have been impractical. Thus, we decided to cluster the R interpreters according to our experience with R. We noticed that in the history of the development of the R language, some macro steps in the enhancements of the language can be identified. To our experience, these correspond to major compatibility issues between scripts, libraries and interpreters. We clustered R interpreters in 3 intervals, according to their version: (*i*) lower than 2.15.0, (*ii*) between 2.15.0 and 3.0.0 (excluded), (*iii*) higher or equal to 3.0.0. We decided to run scripts developed with interpreters in the first range, by means of an R 2.14.1 interpreter. Scripts in the second range are executed with R 2.15.3, and scripts in the third range with R 3.0.3. To assess such clustering, we made tests on 20 complex scripts, coming from 5 different research institutes. The tests were successful in spite of the approximation: the behaviours of the algorithms were the same as in their original environments.

3.4. Algorithms as-a-Service

The Statistical Manager is based on a network of web services, which communicate via SOAP protocol among them and with external clients. The platform currently hosts both general and special purpose algorithms implementations for statistical computing and data mining in Computational Biology. SM currently supplies about 100 algorithms implementations including: clustering of environmental characteristics associated to species habitats, ecological niche modelling, climate change impact on species distributions, Bayesian models for life history traits [39].

The system distributes the hosted algorithms as-a-Service. It offers web-based APIs for invoking algorithms executions via thin clients. Furthermore, it is endowed with a dynamically generated user interface that allows final users to select the algorithms, run them and monitor their execution via a plain web browser. The interface interacts with the SM and retrieves the descriptions and the parameters of the processes. Furthermore, it is able to summarize the output of a computation when

it has finished. Figure 2 depicts the user interface to invoke an algorithm. On the left side, the list of capabilities is presented, which are grouped according to a user-oriented perspective. In the right panel, the dynamically generated form for the selected algorithm is presented, which enables to compile algorithm expected inputs according to the proper format. In the figure, a Feed Forward Artificial Neural Network (FFANN) is asked to process some biological data in tabular format. The *StartComputation* button starts the execution.

The interaction design by thin clients, including the web interface, follows very closely the standard OGC WPS specifications [40]. The difference with respect to WPS services is that SM is more flexible in terms of algorithms plugability and in the dynamism of the web interface. The SM is also runnable as part of an e-Infrastructure, built with the gCube framework [28]. This integration allows to add the concept of Virtual Research Environments (VREs) to the SM algorithms. Thus, each algorithm can be published by the SM as an e-Infrastructure resource, and a VRE manager can assign or hide each of them to the VRE participants.

3.5. Limitations and Benefits

Our procedure is applicable to problems that can be solved by means of a Map-Reduce approach. Adaptation of algorithms code is possible in the case of R scripts, while in the case of compiled processes only the input space can be adapted. Modifications are performed by means of a Java wrapper. All the problems that fit these limitations can be managed by means of our procedure. To our experience, this is the case of many experiments in Computational Biology. Such experiments, use several data mining procedures in cascade, where many of these atomic steps can use parallelisation with Map-Reduce. In the case of non-parallel processes, the Statistical Manager executes the complete process on one single powerful machine. The mechanism that matches the correct environment to execute the algorithm is valid also in this case. The mentioned cascade of algorithms can be implemented by using an external workflow management system (e.g. Taverna [24]), which can be connected to the Statistical Manager by using either a thin Java client or the SOAP protocol.

Thus, the above limitations define the set of problems our system can solve. The link to the Computational Biology domain is due to a long experimentation in such domain and to a tight connection with the D4Science e-Infrastructure [41]. This currently host datasets that regard biodiversity and climate for the most part. The applicability to other domains than Computational Biology is under investigation. Some cases fall under our constraints yet, but others would need to manage different kinds of parallelisation processes. For example, they could require to manage Direct Acyclic Graphs or multi-temporal granularities.

On the other side, the benefits of our solution with respect to other ones are in the aggregations of facilities that concretely address scientists' needs. These needs can be summarized as (*i*) importing and executing native programs with minimal modifications and possibly in their native environment, (*ii*) automatic generation of a user interface for the algorithms, (*iii*) publishing mechanisms to distribute the algorithms as-a-Service, (*iv*) direct access to biodiversity and climate datasets managed by an e-Infrastructure. The use case presented in the next section gives an example of application of this method, which concretely reduced the time-to-paper for a group of scientists.

4. USE CASE

In order to demonstrate the benefits our platform can bring to a sequential execution, we selected a non-trivial use case. We applied our method to the R script that produced the results in [5]. The script uses a Bayesian hierarchical model to estimate the relationships between the lengths and the weights (LWR) of 32,000 species (520 families) of fishes. The procedure uses the JAGS modelling framework for R, which implements a Monte Carlo Markov Chain method. The JAGS package is pre-installed by default on all the Worker nodes in our SM installation, along with a large set of Bayesian modelling and data mining packages. The script was developed with R 2.15.1. It uses existing LWR studies to derive species-specific LWR parameters. In the case of data-poor species,

the analysis relies on LWR studies of closely related species with the same body shape. The outcome of the study allows practitioners to transform fishes weights into lengths and vice versa, for a large set of species. The analysis requires, for each species, to collect statistics (*i*) about all the species in the same family and (*ii*) about all the species, outside the family, that have similar shape. This means that the algorithm must process each single species against a very large amount of other species. The native process focuses on one family at time.

The species length-weight parameters are contained in an input table, along with indications about the belonging families. Thus, the Statistical Manager could parallelise by splitting either the column containing the list of species or the column containing the list of families. The native script was conceived to apply the analysis sequentially to each family, thus we decided to parallelise on the family dimension. This means that each node had to process each species in a family against all the species of the same family and also against other ones having the same shape. Thus, the native script had to be limited to run only on one specific family on each Worker node. The native script accepted only one entire dataset as input and could not work on a subset because it required to collect statistics on all the species. Thus, the easiest approach to limit the process, was to modify the code on the fly. This was achieved by injecting a simple line in the R script by means of the Java wrapper.

The native script reads a file containing 32,000 species parameters, copies the resulting dataset ($Fam.All$) into two variables ($Families1$ and $Families2$) and then cycles on the families in the first dataset. For each family, it analyses each species against the content of $Families2$. On the other side, the wrapper receives the family to focus on, as input from the SM message. Thus, by means of the wrapper, we simply forced the first dataset to contain only that family. More in details, the code injection for the family to process (e.g. $Acanthuridae$) consisted in substituting the R assignment $Families1 <- Fam.All$ with the assignment $Families1 <- Fam.All[Fam.All == "Acanthuridae"]$. The function $substituteInScript(StringtoSubstitute, StringsubstitutiveString, StringscriptName)$ of the SM framework, facilitated the job. Note that also $scriptName$ is passed as input parameter to the Java wrapper. We leave the technical details on how to use these functions to the SM framework guide about R scripts integration [35]. The native script writes the output on a database table. The database connection parameters are among the input parameters of the procedure and are the same for each script executed by the SM Worker nodes. Each execution of the script enriches the same table with new rows containing species length-weight parameters. Thus, the Reduce phase of this example, which takes place at the end of the Cloud process, is trivial since output information is yet gathered on the produced table. For such reason, we will not take the Reduce phase into account in efficiency comparison.

In order to explain the benefits this application brought, we report a performance comparison in Figure 3 and in Table I. We compared the time (in hours) required by a sequential run of the length-weight estimation with two different parallel processes. The former parallel process executed the procedure on one multi-core machine. Scripts were started in parallel on different chunks of the input dataset. Each chunk was stored in a different file. The number of chunks was equal to the number of cores to employ. Each data chunk contained a set of species families to process. The latter parallel process was managed by the SM, with a different number of nodes used in each test. The chart in Figure 3, shows that the SM multi-machine system becomes more and more efficient when the number of machines increases. In the sequential case, higher time for the SM is due to the overhead in data staging. The reason is that each machine in the SM system has to download data and software. Thus, in the case of one single machine the dataset is very large and this requires more disk writing activity and higher network traffic. Such overhead is absent for the multi-core machine, where data are on the disk yet. On the other hand, the overhead is lower when several machine are used, because the datasets downloaded by the machines are smaller.

As the figure shows, higher efficiency is gained by the multi-core case, when less than 12 cores, with respect to 12 SM machines, are used. Also, the time reduction decreases exponentially at the increase of the number of machines or cores. The different behaviour between the multi-core and multi-machine runs is due to the sharing of machine resources by the cores. This becomes more

evident when the number of used cores approaches the maximum number of exploitable cores. The SM Worker nodes machines were CentOS 5.7 x86 64 with 2 CPUs, 2 GB of RAM, 10 GB of disk. The multi-core machine was a Windows Server 2008 Enterprise 64-bit, with 8 GB or RAM and 24 cores. The reduction in the time required by the computation on 21 Worker nodes with respect to the sequential run was about 98%. This means that the authors could run the script several times to double check the results. Furthermore, they still periodically update the procedure and data, getting 100% match with the results of the sequential run. This benefit allowed them to produce the paper in much shorter time.

There are several differences between our approach and a parallel processing package for R scripts. Enabling multi-core processing for an R script can be a matter of few lines of code using the proper framework. On the other hand, using Cloud computing still requires either to strongly change the script or to make the developer lose control on the environment in which the script will be deployed (see Section 2). This means that the developer is not sure that the script will run in the environment in which it was tested, and also that testing cannot be immediate after the developer has changed the script. On the other hand, we have demonstrated that our Cloud computing system is more efficient than parallel processing on one single machine. Furthermore, our approach makes code modifications easy when the script is parallelisable. We also add a framework to apply modifications in few steps and a testing suite to verify the correctness of the output just after script modifications and before the final deployment. Moreover, the code is executed within an environment that is compliant with the environment in which the script has been tested. As explained in Section 1, after the deployment the script is automatically endowed with a user interface and is published as-a-Service. Finally, our system is flexible enough to apply the same approach to compiled programs that can be parallelised with a Map-Reduce approach, although without the possibility of modifying the code.

5. CONCLUSIONS

We have presented a system that is currently used by a Computational Biology community of practice through both the i-Marine[†] and the D4Science[‡] web portals. The system was designed to meet the requirements of biologists, who wanted to enhance the efficiency of their algorithms. At the same time, they did not want to lose the possibility to update them periodically and to preserve the step-by-step behaviour of the procedures. We have described the technical steps required by the integration of a native algorithm. Furthermore, we have clarified our assumptions and limitations for parallelising the algorithms, as these must be compliant with a Map-Reduce approach. On the other side, our experience is that this fits many cases in Computational Biology, where a sequence of data mining algorithms is usually applied in sequence, and each step can be parallelised using a Map-Reduce approach.

We have shown a practical application to a non-trivial model, in which an R script was adapted to be parallelised. The benefits to our final users were the possibility to apply periodic modifications and tests, which was impractical with the native algorithm. Furthermore, our system reduced the time-to-paper for the authors. Other advantages are that (*i*) our system grants that the algorithm will be executed in a compliant environment, (*ii*) it automatically endows the algorithm with a user interface, (*iii*) it finally provides the algorithm as-a-Service. Our architecture is part of an e-Infrastructure for biodiversity and environmental data [41]. This allows to organize the provided algorithms according to the Virtual Research Environment paradigm. We are experimenting our approach also outside the Computational Biology field, searching for new domains or single use cases, where algorithms have the same requirements and constraints.

[†]i-Marine web site `www.i-marine.d4science.org`
[‡]D4Science web site `www.d4science.org`

ACKNOWLEDGMENTS

The reported work has been partially supported by the i-Marine project (FP7 of the European Commission, INFRASTRUCTURES-2011-2, Contract No. 283644).

References

1. Hijmans RJ, Graham CH. The ability of climate envelope models to predict the effect of climate change on species distributions. *Global Change Biology* 2006; **12**(12):2272–2281, doi:10.1111/j.1365-2486.2006.01256.x. URL http://dx.doi.org/10.1111/j.1365-2486.2006.01256.x.
2. Pearson RG, Dawson TP. Predicting the impacts of climate change on the distribution of species: are bioclimate envelope models useful? *Global ecology and biogeography* 2003; **12**(5):361–371.
3. Thomas CD, Cameron A, Green RE, Bakkenes M, Beaumont LJ, Collingham YC, Erasmus BF, De Siqueira MF, Grainger A, Hannah L, *et al.*. Extinction risk from climate change. *Nature* 2004; **427**(6970):145–148.
4. Stearns SC. The evolution of life history traits: a critique of the theory and a review of the data. *Annual Review of Ecology and Systematics* 1977; **8**(1):145–171.
5. Froese R, Thorson JT, Reyes RB. A bayesian approach for estimating length-weight relationships in fishes. *Journal of Applied Ichthyology* 2014; **30**(1):78–85, doi:10.1111/jai.12299. URL http://onlinelibrary.wiley.com/doi/10.1111/jai.12299/abstract.
6. Claudet J, Osenberg C, Domenici P, Badalamenti F, Milazzo M, Falcón J, Bertocci I, Benedetti-Cecchi L, García-Charton J, Goñi R, *et al.*. Marine reserves: fish life history and ecological traits matter. *Ecological applications* 2010; **20**(3):830–839.
7. Candela L, Castelli D, Coro G, Pagano P, Sinibaldi F. Species distribution modeling in the cloud. *Concurrency and Computation: Practice and Experience* 2013; :n/a–n/adoi:10.1002/cpe.3030. URL http://onlinelibrary.wiley.com/doi/10.1002/cpe.3030/abstract.
8. Candela L, Castelli D, Coro G, De Faveri F, Italiano A, Lelii L, Mangiacrapa F, Marioli V, Pagano P. D4Science facilities for managing biodiversity databases. *Technical Report 2013-TR-035*, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR 2013.
9. Amaral R, Badia RM, Blanquer I, Braga-Neto R, Candela L, Castelli D, Flann C, De Giovanni R, Gray WA, Jones A, *et al.*. Supporting biodiversity studies with the EUBrazilOpenBio Hybrid Data Infrastructure. *Concurrency and Computation: Practice and Experience* 2014; **n/a**:n/a, doi:10.1002/cpe.3238.
10. 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.
11. Vanden Berghe E, Bailly N, Coro G, Fiorellato F, Aldemita C, Ellenbroek A, Pagano P. Bionym: a flexible workflow approach to taxon name matching. *Technical report*, CNR 2014. Technical Report 2014-TR-022, Istituto di Scienza e Tecnologie dellInformazione A. Faedo, CNR.
12. Schmidberger M, Morgan M, Eddelbuettel D, Yu H, Tierney L, Mansmann U. State-of-the-art in parallel computing with r. *Journal of Statistical Software* 2009; **47**(1).
13. Li MN, Rossini A. RPVM: Cluster statistical computing in R 2001. Http://cran.uib.no/doc/Rnews/Rnews_2001-3.pdf#page=4.
14. Yu H. Rmpi: Parallel statistical computing in R. *R News* 2002; **2**(2):10–14.
15. Gentleman R. *R programming for bioinformatics*. CRC Press, 2008.
16. Rossini A, Tierney L, Li N. Simple parallel statistical computing in r 2003. UW Biostatistics Working Paper Series. Working Paper 193. http://biostats.bepress.com/uwbiostat/paper193.
17. Wegener D, Sengstag T, Sfakianakis S, Ruping S, Assi A. Gridr: An r-based grid-enabled tool for data analysis in acgt clinico-genomics trials. *e-Science and Grid Computing, IEEE International Conference on*, IEEE, 2007; 228–235.
18. Hill J, Hambley M, Forster T, Mewissen M, Sloan TM, Scharinger F, Trew A, Ghazal P. Sprint: A new parallel framework for r. *BMC bioinformatics* 2008; **9**(1):558.
19. Roberts JJ, Best BD, Dunn DC, Treml EA, Halpin PN. Marine geospatial ecology tools: An integrated framework for ecological geoprocessing with arcgis, python, r, matlab, and c++. *Environmental Modelling & Software* 2010; **25**(10):1197–1207.
20. RapidMiner, Inc. The Rapid Miner Software 2014. Http://rapidminer.com/.
21. Hunter A, 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.
22. Bhandarkar M. Mapreduce programming with apache hadoop. *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, IEEE, 2010; 1–1.
23. Tejedor E, Ejarque J, Lordan F, Rafanell R, Alvarez J, Lezzi D, Sirvent R, Badia RM. A cloud-unaware programming model for easy development of composite services. *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, IEEE, 2011; 375–382.
24. Oinn T, Greenwood M, Addis M, Alpdemir MN, Ferris J, Glover K, Goble C, Goderis A, Hull D, Marvin D, *et al.*. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience* 2006; **18**(10):1067–1100. URL {http://dx.doi.org/10.1002/cpe.993}.
25. Goecks J, Nekrutenko A, Taylor J, *et al.*. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol* 2010; **11**(8):R86.
26. Mitra N, Lafon Y, *et al.*. Soap version 1.2 part 0: Primer. *W3C recommendation* 2003; **24**:12.
27. Curbera F, Duftler M, Khalaf R, Nagy W, Mukhi N, Weerawarana S. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing* 2002; **6**(2):86–93.

28. Candela L, Castelli D, Pagano P. gcube: a service-oriented application framework on the grid. *ERCIM News* 2008; **72**:48–49.
29. Candela L, Castelli D, Pagano P. gcube v1. 0: A software system for hybrid data infrastructures. *Technical Report*, Technical Report 2008-TR-035, Istituto di Scienza e Tecnologie dellInformazione A. Faedo, CNR 2008.
30. Candela L, Castelli D, Pagano P. Managing big data through hybrid data infrastructures. *ERCIM News* 2012; (89):37–38.
31. Candela L, Castelli D, Pagano P. Virtual research environments: an overview and a research agenda. *CODATA Data Science Journal* 2013; **12**:GRDI75–GRDI81, doi:10.2481/dsj.GRDI-013.
32. Jennings R. *Cloud Computing with the Windows Azure Platform*. John Wiley & Sons, 2010.
33. Coro G, Italiano A. Statistical manager developer's guide 2012. Http://gcube.wiki.gcube–system.org/gcube/index.php/How–to_Implement_Algorithms_for_the_Statistical_Manager.
34. MongoDB. an open-source, document-oriented database designed for ease of development and scaling 2014. Www.mongodb.org.
35. Coro G. Statistical manager developer's guide: Integrating r scripts 2012. Http://gcube.wiki.gcube–system.org/gcube/index.php/How–to_Implement_Algorithms_for_the_Statistical_Manager#Integrating_R_Scripts.
36. Candela L, Castelli D, Pagano P. Making virtual research environments in the cloud a reality: the gcube approach. *ERCIM News* 2010; **2010**(83):32.
37. Ellenbroek A. Building an e-infrastructure for capture statistics and species distribution modeling: the d4science approach. *Book of Abstracts*, 2010; 174.
38. Begin ME, Sancho GDA, Di Meglio A, Ferro E, Ronchieri E, Selmi M, Zurek M. Build, configuration, integration and testing tools for large software projects: Etics. *Rapid Integration of Software Engineering Techniques*, Springer, 2007; 81–97.
39. Coro G, Candela L. gCube statistical manager: The algorithms. *Technical Report 2014-TR-027*, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR 2014.
40. Lanig S, Schilling A, Stollberg B, Zipf A. Towards standards-based processing of digital elevation models for grid computing through web processing service (wps). *Computational Science and Its Applications–ICCSA 2008*. Springer, 2008; 191–203.
41. Candela L, Castelli D, Pagano P. D4science: an e-infrastructure for supporting virtual research environments. *IRCDL*, 2009; 166–169.

Number of Cores or Machines	Computation Time (hours) Multi-core Machine	Computation Time (hours) SM (Multi-machines)
1 (Sequential run\1 Working node)	480	500
5	240	250
12	48	36
21	25	11

Table I. Details of the computation times in the comparison between the sequential run of an R script for marine species length-weight relationship estimation, against a parallel process on one multi-core machine and on several machines. The process estimates parameters for 520 families of marine species containing a total amount of 32,000 species. The parallelisation is made on the number of families. The second column refers to a computation on one multi-core machine, while the third column contains the performance of a Cloud computing system based on the Statistical Manager service.

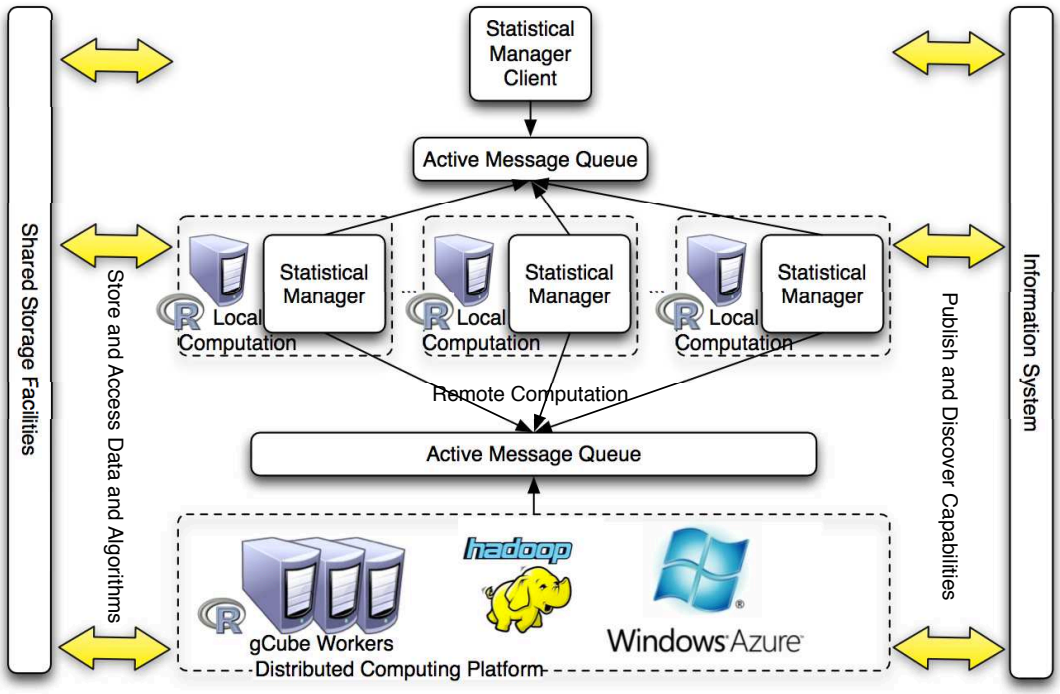

Figure 1. Architecture of the gCube Statistical Manager.

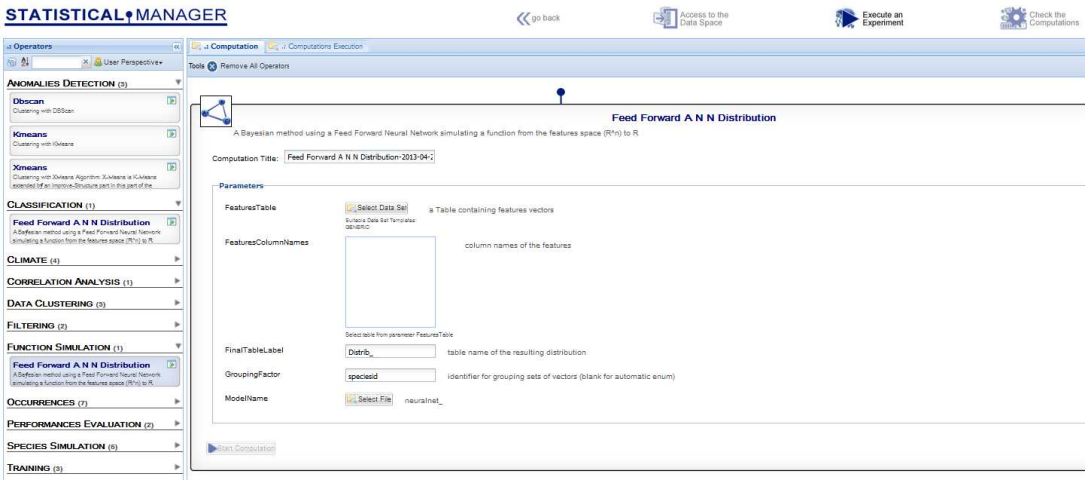

Figure 2. Web-based user interface of the Statistical Manager. The left side shows the grouped list of available algorithms. The right panel reports the interface of a Feed Forward Artificial Neural Network algorithm.

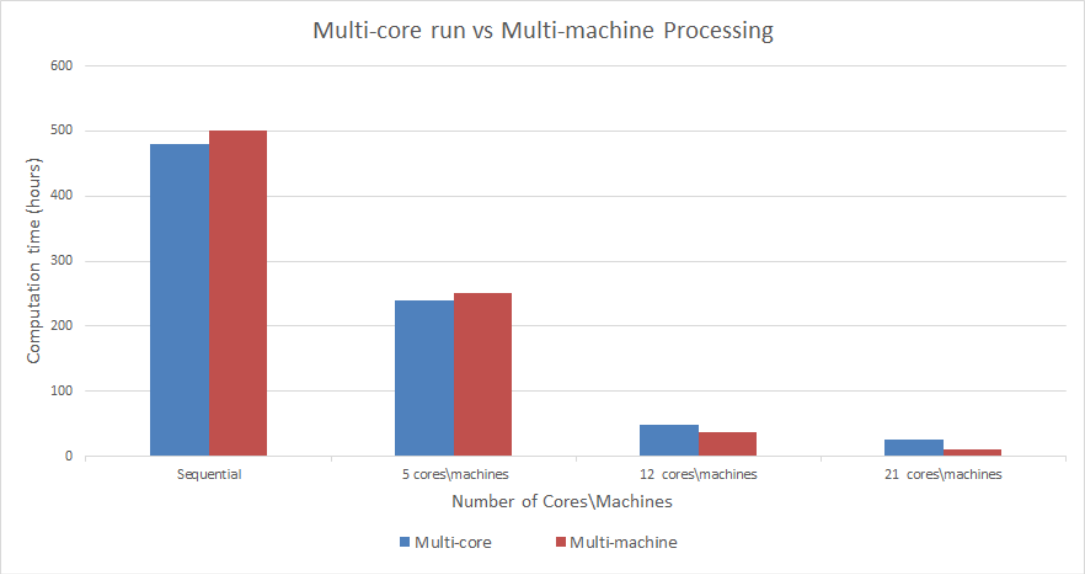

Figure 3. Comparison between the sequential run of an R script for marine species length-weight estimation, against parallel processing on one multi-core machine and on several machines.