OMWS: A WEB SERVICE INTERFACE FOR ECOLOGICAL NICHE MODELLING

RENATO DE GIOVANNI[1], ERIK TORRES[2], RAFAEL B. AMARAL[3], IGNACIO BLANQUER[2], VINOD REBELLO[3], VANDERLEI P. CANHOS[1]

[1]*CRIA - Centro de Referência em Informação Ambiental, Av. Dr. Romeu Tórtima, 388, Campinas, SP, Brazil;* [2]*Institute of Instrumentation for Molecular Imaging (I3M), Universitat Politècnica de València, Camino de Vera s/n, Valencia, Spain;* [3]*Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brazil*

Abstract.—Ecological niche modelling (ENM) experiments often involve a high number of tasks to be performed. Such tasks may consume a significant amount of computing resources and take a long time to complete, especially when using personal computers. OMWS is a Web service interface that allows more powerful computing back-ends to be remotely exploited by other applications to carry out ENM tasks. Its latest version includes a new operation that can be used to specify complex workflows in a single request, adding the possibility of using workflow management systems on parallel computing back-end. In this paper we describe the OMWS protocol and compare its most recent version with the previous one by running the same ENM experiment using two functionally equivalent clients, each designed for one of the OMWS interface versions. Different back-end configurations were used to investigate how the performance scales for each protocol version when more processing power is made available. Results show that the new version outperforms (by a factor of two) the previous one when more computing resources are used.

Key words.—workflow, high-throughput computing, openModeller

Studies involving ecological niche modelling (ENM) sometimes require hundreds (see Segurado & Araújo 2004, Marmion *et al.* 2009, Feeley & Silman 2010, Lorena *et al.* 2011), thousands (see Farber & Kadmon 2003, Elith *et al.* 2006, Wisz *et al.* 2008) or even millions (see Diniz-Filho *et al.* 2009) of models and related procedures to be carried out. The reason is that models can be created for multiple species using different algorithms with different sets of parameter values. There can also be many replicates for model tests, as well as many model projections based on different environmental scenarios. Depending on the experiment, performing such tasks on personal computers can be a troublesome experience, if not prohibitive. Even relatively simple experiments can take significant time to complete or, in some cases, exhaust machine resources when computing-intensive algorithms and high-resolution environmental layers are used. For this reason, ENM is a typical field where software applications can greatly benefit from parallelization techniques and high-throughput computing by outsourcing the execution of tasks to more a powerful pool of computing resources.

Over the Internet, different strategies can be used to remotely exploit such computational infrastructures (Kai *et al.* 2011). In all scenarios, an entry point, which can be an application directly accessible to end users or an intermediate Web service, is used to communicate with the larger system. Web services are web-based applications that support dynamic interactions with other software applications using open standards that include data formats like eXtensible Markup Language (XML) (World Wide Web Consortium, 2006) to transmit data over a network via Internet-based protocols. Web services provide a standard means of interoperating between different software applications running on a variety of platforms and/or frameworks (World Wide Web Consortium, 2004a). For this reason, Web services currently permeate Web development, as can be seen by the fact that most major Web sites provide some mechanism for other programs to access their data or use their functionality. This allows all kinds of user interfaces and other applications to be developed on top of standard Application Programming Interfaces (APIs) created for remote calls, making Web services important components of complex cyberinfrastructures (see Stein 2008 and Amaral *et al.* 2014 for examples). Moreover, service-based applications can free users from the burden of configuring more specific computing resources.

Over the years, many tools were developed for ENM, mostly as Desktop applications. To our knowledge, the first Web application for ENM was released in 1994 (Boston & Stockwell 1995), while the first Web service for ENM appeared ten years later as part of openModeller (Muñoz *et al.* 2011), and it was recently named the openModeller Web Service (OMWS). The first OMWS prototype was created for the Biodiversity World project (Pahwa *et al.* 2006) so that scientific workflows for ENM could be built using the Triana workflow management system (Taylor *et al.* 2003). This prototype was improved and soon became officially part of the openModeller toolbox. More recently, other initiatives such as the LifeMapper project[1] and eHabitat (Skøien *et al.* 2013) also created Web services for ENM.

Although the first version of OMWS managed to cover the most important ENM tasks by means of individual operations, any complex experiment would require a client program to send a large number of requests and manage any dependencies between them. This could lead to inefficient bandwidth use, preventing back-end implement-tations to fully exploit more sophisticated tools and parallelization techniques. In this paper we present the latest version of OMWS that includes new operations and additional changes, allowing complex ENM experiments to be fully specified in a single request without losing the possibility of using the previous individual calls that remained in the protocol. This paper presents an overview of the protocol and its general design, as more specific details of all operations and the corresponding input/output parameters can be found in the official documentation[2]. The capabilities of OMWS are then demonstrated with a typical ENM experiment implemented for both protocol versions. The experiment is performed against the same server hosting both versions of the service. Different back-end configurations were used to measure speedup and saturation point (performance limit) when more processing power was made available.

[1] http://lifemapper.org.

[2] http://openmodeller.sf.net/web_service_2.html.

[2] http://openmodeller.sf.net/web_service_2.html.

OMWS Scope and General Design

OMWS was designed to cover essential ENM tasks, such as model creation, testing and projection, without including other pre or post processing operations like data cleaning, species occurrence data retrieval from other sources or raster aggregation. Although frequently used together with ENM procedures, such additional tasks would make the protocol significantly more complex and would produce overlaps with other more specific protocols. Due to the potentially long duration of the ENM tasks, OMWS was deliberately designed to be a processing protocol with asynchronous operations, generating outputs that are temporarily stored on the server side to be retrieved later by clients. There are no operations for explicitly storing or manipulating objects on the server, such as occurrence points, algorithms, models or environmental layers. OMWS should therefore only be seen as a remote niche modelling engine service, not a repository. Model repositories, such as the EUBrazilOpenBio niche modelling application (Amaral *et al.* 2014) or BioGeo[3], can use OMWS behind the scenes, but they need to provide additional functionality related to authentication, storage and search capabilities not covered by OMWS.

Technically, OMWS is currently built on top of the Simple Object Access Protocol (SOAP) (World Wide Web Consortium, 2007): a generic messaging framework created to facilitate communication between applications running on different platforms with different technologies. By using SOAP, the whole protocol is programmatic-cally defined in a Web Service Definition Language (WSDL) (World Wide Web Consortium, 2001) document specifying all operations, inputs and outputs. Through WSDL, existing SOAP frameworks for different programming languages can automatically generate proxy code to interact with the service. For maximum interoperability and better performance, OMWS follows the SOAP Document/Literal style, which means that

[3] http://biogeo.inct.florabrasil.net.

messages are exchanged between client and server as plain XML documents fully encoded according to XML Schema (World Wide Web Consortium, 2004b) serialization rules.

Since OMWS was originally created as a web service for the openModeller toolbox, its main XML definitions are all based on the openModeller XML Schema[4]. Although this clearly facilitates using openModeller tools in the background, by no means it prevents other niche modelling tools to be used. New server implementations could be developed, or the existing standard server implementation could use a plugin approach, translating inputs and outputs of different ENM tools to the expected data structures.

Main Data Types

Like openModeller, OMWS follows the ENM correlative approach (Soberón & Peterson 2005) by relating species occurrence points with spatially explicit environmental variables so that the corresponding environmental data can be used by an algorithm to generate a niche model. Therefore, environmental layers, occurrence points (presence, absence or background), algorithms, models and model projections are the fundamental data types used by the protocol.

Since environmental layers can frequently be very large raster files, they are referenced in the protocol by an identifier. There are no restrictions or assumptions for such identifiers other than being unique and resolvable by the service. They can refer to files stored on the server side or even point to remote resources. Environmental scenarios are specified as a sequence of environmental layers followed by an optional mask also referenced by an identifier. With this approach, rasters that are frequently used can be previously stored on the server and advertised through the *getLayers* operation. This way, clients can browse the available options and use appropriate identifiers for each selected layer when building requests. Remote raster sources can also be used, either as files directly available through HTTP or FTP, or as more formal raster repositories exposed through services like the Web Coverage Service (WCS)[5]. Any of these options enables different mechanisms to be set up so that users can provide their own layers if necessary (e.g., by granting upload access to a specific directory on the server, or by configuring the server to access a remote resource where users have full control over the available rasters). However, for flexibility and simplicity, such mechanisms are not covered by the protocol, which also does not put any constraints on the content of layer identifiers. This approach makes the protocol more generic, but at the same time requires any additional mechanisms, capabilities or restrictions to be documented and communicated by service providers.

Occurrence points are often used in low numbers in ENM, although there can be situations where thousands of points are available for the species. Additionally, the same set of occurrence points is seldom reused by other ENM experiments. For this reason, occurrence points are always completely included in requests. Each point contains two mandatory attributes for the coordinates and an optional attribute for the corresponding environmental values, in which case the service is relieved of the task of reading environmental data from the corresponding layers. Each point in a request must have its coordinates expressed in the same spatial reference system specified in Well-Known Text (Herring 2011) for the whole set of points.

OMWS advertises available algorithms through the *getAlgorithms* operation. There is no fixed or standardised set of algorithms, so each service is free to decide which algorithms can be used. Algorithm metadata includes name, description, bibliography, authors, developers and parameter metadata (name, data type, domain and description), besides algorithm and parameter identifiers that are used by the other operations when specifying algorithm and parameter values.

Each different algorithm in ENM produces a completely different kind of model. For instance, the result of Bioclim (Nix 1986) is a series of envelopes comprised by minimum, maximum, mean and standard deviation values for each variable, while Random Forests (Breiman 2001) produces a set of decision trees, and Artificial Neural Networks (Tarassenko 1998) produces a system of interconnected neurons with activation functions and weights for each interconnection. OMWS deals with such diversity by allowing each algorithm to have its own particular XML representation for models, without imposing any

[4] http://openmodeller.cria.org.br/xml/2.0/openModeller.xsd.
[5] http://www.opengeospatial.org/standards/wcs.

kind of validation. Regardless its representation, models can be reused in subsequent calls for testing or projection purposes.

Model projections are rasters produced in a specific format (e.g., GeoTiff) based on a specific template raster that indicates the resolution and spatial reference system to be used. Each projection is based on a given environmental scenario. Projections can be retrieved from a URL obtained by calling the *getLayerAsURL* operation when the procedure is finished.

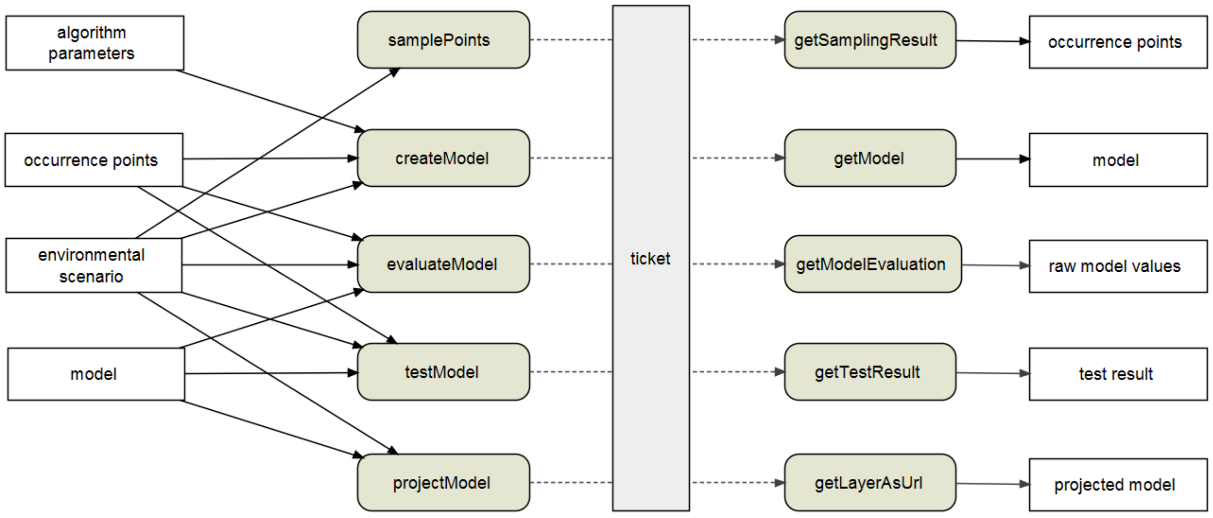

Figure 1: Paired individual asynchronous operations with their main inputs and outputs. Each asynchronous operation generates a ticket that is used as input by its counterpart operation to retrieve results later. Note: the first column shows only the main types of input for each operation (additional parameters are available).

OPERATIONS

OMWS includes different kinds of operations, starting with a simple *ping* operation that can be used to monitor service status. Two other operations, *getAlgorithms* and *getLayers* are used to advertise algorithms and environmental layers available, respectively. The remaining operations can be divided into three groups. The first one is a set of individual asynchronous operations related to the main ENM tasks (Figure 1): *createModel*, *testModel*, *projectModel* and, in the most recent version of the protocol, *samplePoints* and *evaluateModel*. All these operations return a ticket that can be used later to call the corresponding operation for retrieving results: *getModel*, *getTestResult*, *getLayerAsURL*, *getSamplingResult* and *getModelEvaluation*. For model projections, an additional operation called *getProjection-Metadata* can be used to retrieve more information about a projection, such as the number of cells predicted present for a given threshold. Despite the similar names, model testing and model evaluation are different operations in OMWS. The former is used for typical threshold-dependent (confusion matrix) or threshold-independent (ROC curve) calculations, while the later is used to calculate raw model values for each given occurrence point in a given environmental scenario.

The second group of operations was included in the latest version of the protocol, allowing complex experiments to be specified in a single call and then processed in an optimized way on the server. A *runExperiment* request may contain any number of ENM jobs with or without dependencies between them (Figure 2). Each job contains its own set of parameters where each parameter either points to a fixed value specified in the first section of the request or to the output of another job. Frequent situations such as generating models for multiple species using multiple algorithms and then testing or projecting results into different environmental scenarios can be expressed with this new kind of request.

The *runExperiment* operation is also asynchronous, returning a set of individual tickets for each job. The corresponding operation *getResults* can be used to fetch sets of results given one or more tickets.

Finally, the last group of operations is used for job management after any asynchronous call: *getProgress* returns the status of one or more jobs, *getLog* returns the job log and *cancel* can be used to abort one or more jobs.

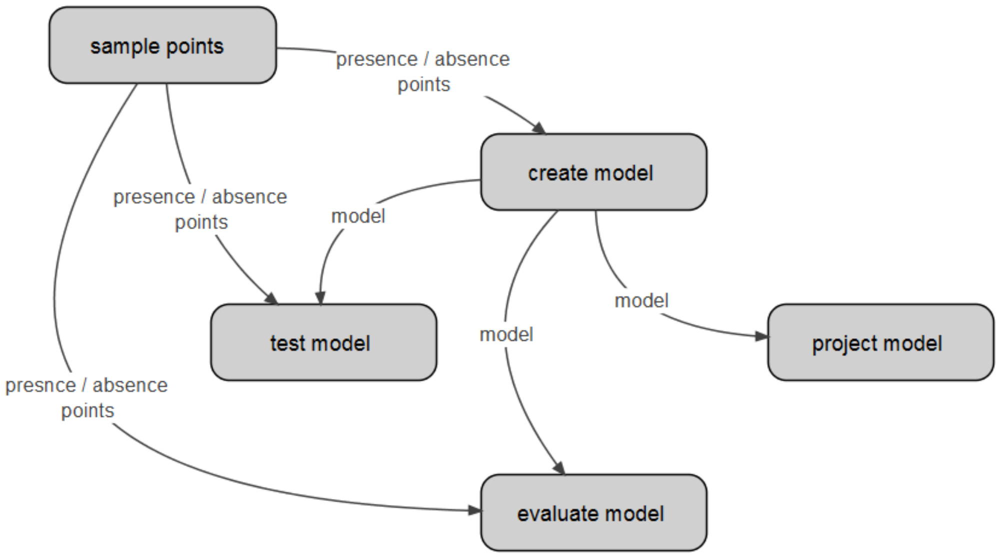

Figure 2: Types of jobs and their possible dependencies in a *runExperiment* call.

ENM Experiment

To illustrate how the service can be used in a real world situation, a typical ENM experiment involving multiple steps was created. The experiment was tested against two services hosted on the same server, each one compatible with one of the protocol versions (hereafter referred to as OMWS1 and OMWS2) for performance comparison. Additionally, different back-end configuretions were used, each time adding more processing power on the server side. Two equivalent client programs were developed in Python, one for OMWS1, where the client has to be responsible for managing the whole workflow sending individual requests for each task, and the other for OMWS2 with all tasks specified in the new *runExperiment* operation where the server is responsible for managing the whole workflow. Client programs were executed at the Internet Data Center from the Brazilian National Research and Educational Network, while the server was located at the Universitat Politècnica de València in Spain. Average connection speed between client and server was measured as 9.7Mbits/s. On the server side, tests started with a single machine with 16GB of RAM and 4 cores running the service initially configured to process a maximum of 3 parallel jobs. Next, the use of HTCondor (Thain *et al.* 2005) was enabled on the server side, with the Master node running on the same machine as the service. Working nodes had the same computing resources (16GB of RAM and 4 cores) and were gradually added to the pool until reaching a maximum of 8 nodes (128 GB of RAM and 32 cores in total). The OMWS2 server implementation for HTCondor used DAGMan (Couvares *et al.* 2007) to handle complex experiment requests.

The ENM experiment consisted of generating individual models for several species using different algorithms. There was no specific concern about comparing the relative performance of each algorithm or even assessing model quality for each species for any particular use, although these would be obvious follow ups in a real use case. Our sole interest was to demonstrate the service with a typical ENM experiment, compare the two protocol versions and show how different back-ends can be used and how they influence the overall processing time and computing resource usage efficiency.

Five arbitrary species of Passifloraceae from the Brazilian Flora having a minimum set of twenty occurrence points were selected. All points were downloaded from BioGeo, where they were previously filtered and cleaned. Also five algorithms were used to generate models: ENFA (Hirzel *et al.* 2002), GARP Best Subsets (Anderson *et al.* 2003), Mahalanobis Distance (Farber & Kadmon 2003), Maxent (Phillips *et al.* 2006) and one-class Support Vector Machines (Schölkopf *et al.* 2001). Since some of these algorithms rely on background or pseudo-absence points and it is known that the area from where such points are sampled can influence model results (Barve *et al.* 2011), individual masks were created for each species. The idea is that each mask approximates the area that has been historically accessible to the species, ensuring that background or pseudo-absence points are only sampled from environments where the species had the opportunity to colonize. Our approximation was done by buffering each set of presence points by 500km and then merging the circles into a single polygon that was finally transformed into a raster. All masks were uploaded to a server where they became accessible to the service. Each mask was used to sample 10k geographically unique background points for algorithms that required them, and also to delimit model projections for each species. For simplicity, and since all species are plants with similar requirements, the same set of high-resolution environmental layers currently used in BioGeo was used for all species (seven bioclimatic variables and altitude). All layers, including masks, had the same resolution of 30 arc-seconds and were locally available on the server (masks were previously cached by running a preparatory experiment just to force mask download). To match the environmental data precision and avoid redundancies, data cleaning filters in BioGeo selected points with a maximum location uncertainty of 500m and removed duplicate points for the same pixel.

The actual experiment executed against the service included running extrinsic tests and generating final models for each pair species-algorithm. Extrinsic tests used 5-fold cross-validation, averaging the partial AUC (Peterson *et al.* 2008) for a maximum omission of 20%. Final models were created using all points and were followed by an internal test using the same measurement of the extrinsic test and by a native projection. Therefore, the whole experiment contained 125 model creations followed by 125 model tests for the extrinsic tests (5 folds * 5 algorithms * 5 species), and 25 model creations (5 algorithms * 5 species) followed by 25 internal model tests and 25 model projections for the final models, totalizing 325 steps. The experiment was repeated 3 times for each protocol and back-end configuration, using the average as the final measurement.

The service code is open source and part of the openModeller toolbox (we used revision #6045 from the openModeller repository[6]). Both clients used on the tests and all input data (masks, points and layer references) are publicly available[7].

Experiment Results

The two protocols performed similarly from the initial single-machine configuration until an HTCondor set up with 3 working nodes (12 cores), from where OMWS2 started to perform increasingly better than OMWS1 (Figure 3).

The initial duration for the experiment was 2h44min for both protocols. OMWS1 reached saturation point with 4 working nodes (duration time of 40min) with a 4.0 speedup compared with the single-machine configuration, which means that adding more computing resources to the back-end did not improve efficiency. Saturation point for OMWS2 could not be detected, as its performance continued to improve until the server infrastructure was saturated, reaching a speedup of 8.6 (19min) with 8 working nodes (32 cores).

Discussion and Conclusions

Regardless the OMWS protocol version, when a service implementation is capable of exploiting more powerful computing resources there can be significant performance improvement in ENM experiments, as clearly demonstrated by the results. OMWS2 performed better than OMWS1 when more computing resources became available. This can probably be explained by the fact that OMWS1 clients need to manage workflows on

[6] http://sourceforge.net/p/openmodeller/svn/HEAD/tree/trunk/openmodeller/.
[7] http://dx.doi.org/10.6084/m9.figshare.1301521.

complex experiments without any clue about or control over server resources, while OMWS2 clients can completely delegate workflow management to the service, where more specialized tools can make use of additional information to optimize resources usage.

Additionally, by being able to specify complex experiments with a single request, OMWS2 requires fewer interactions between client and server, also simplifying client code. The task of developing new server software, however, gets more challenging with OMWS2 to handle workflows, although this also opens the possibility of using existing workflow management tools, such as HTCondor DAGMan in our case. Another example is a new OMWS server implementation under development using COMP Super Scalar with Cloud resources (Lezzi *et al.* 2013), which was used to test a prototype protocol that was later improved and became OMWS2.

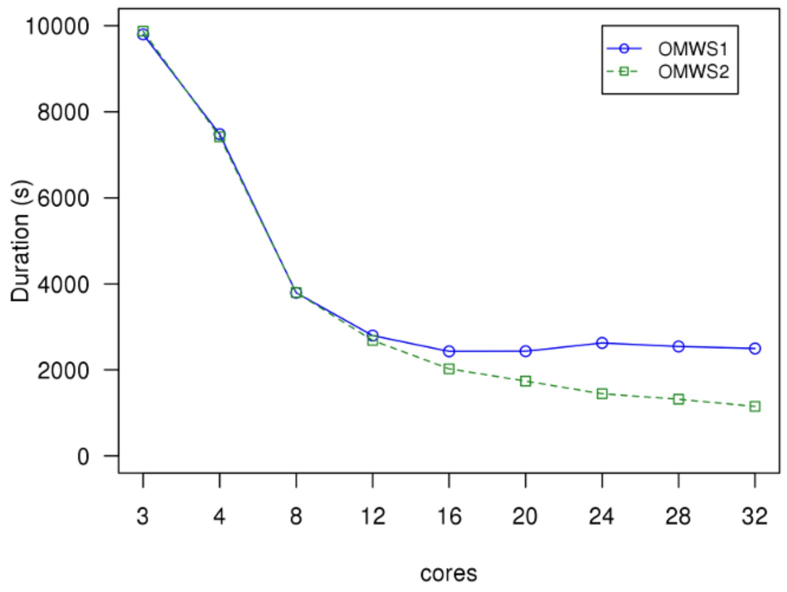

Figure 3: Average performance after three repetitions for the different back-end configurations using both versions of OMWS. Standard deviation was low for the graph scale (55s in average) so it is not being represented here.

The example tested here also shows that a real world ENM experiment likely requires additional tasks not covered by OMWS, such as pre-processing or post-processing data. This does not mean that such tasks can only be performed as unconnected individual steps, since there are many tools that can be used to integrate and orchestrate tasks performed by different services or software, such as Kepler (Altintas *et al.* 2004) and Taverna (Wolstencroft *et al.* 2013).

OMWS was created before some of the existing geospatial standards, in particular those defined by the Open Geospatial Consortium[8]. Since OMWS is mostly based on geospatial data and operations, basic data types such as points could now be expressed according to OGC Geography Markup Language (GML)[9], or even according to the DarwinCore biodiversity data standard (Wieczorek *et al.* 2012). In fact, the whole OMWS protocol could be encapsulated as an OGC Web Processing Service (WPS)[10]. However, in the particular situation of OMWS, the benefits of adhering to such standards are still unclear or premature in terms of improving interoperability with other software. Similar ENM initiatives already started to explore the use of WPS with interesting results (Cavner *et al.* 2011, Skøien *et al.* 2013), although the incipient set of specific software libraries to interact with WPS services and the lack of standard strategies to facilitate WPS service chaining are still issues to be addressed. Nonetheless, future

[8] http://opengeospatial.org.

[9] http://opengeospatial.org/standards/gml.

[10] http://opengeospatial.org/standards/wps.

versions of OMWS could be adjusted or even wrapped to become compatible with other standards. Another possible future improvement for OMWS given the asynchronous nature of most of its operations is to become compatible with the WebSockets protocol[11], which provides bi-directional, full-duplex TCP connections. This could reduce network traffic currently associated with OMWS *getProgress* calls.

The number of ENM applications interested in outsourcing most of the processing tasks to specialized Web Services is increasing over time. Besides other recently emerged protocol initiatives for ENM, the use of OMWS itself also increased over the years. Since its first prototype version used by the Biodiversity World project, OMWS was used for many years by the Global Biodiversity Information Facility[12] data portal, and is still being used by openModeller Desktop users. More recently, BioGeo, through the Brazilian Virtual Herbarium, is using a separate OMWS server to process all ENM requests. The EUBrazilOpenBio project created a Web interface for ENM where users can create complex experiments involving multiple species, algorithms and environmental scenarios to be processed by an OMWS2 service through the new *runExperiment* operation (Amaral *et al.* 2014). All ENM workflows created as part of the BioVeL project[13] also interact with an OMWS2 service. By offering a standard interface for the most important ENM tasks with an open source server implementation that can be deployed on more powerful computational infrastructures, OMWS can be a relevant tool to address some of the challenges of ENM research.

Acknowledgments

The latest version of OMWS contains improvements coming from different sets of requirements originated from two projects that funded their corresponding implementation: EUBrazilOpenBio[14], with grants from the European Commission and the National Council for Scientific and Technological Development of Brazil (CNPq) of the Brazilian Ministry of Science and Technology (MCT), and BioVeL, with grants from the European Commission. Server infrastructure was operated through a provisioning system developed in the frame of the Spanish project CLUVIEM (TIN2013-44390-R) funded by the "Ministerio de Economía y Competitividad".

[11] http://www.websocket.org.
[12] http://gbif.org.
[13] http://biovel.eu.
[14] http://eubrazilopenbio.eu

Literature Cited

Altintas, I., C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. 2004. Kepler: an extensible system for design and execution of scientific workflows, in: Proc 16th International Conference on Scientific and Statistical Database Management, pp.423-424.

Amaral, R., R.M. Badia, I. Blanquer, R. Braga-Neto, L. Candela, D. Castelli, C. Flann, R. Giovanni, W.A. Gray, A. Jones, D. Lezzi, P. Pagano, V.P. Canhos, F. Quevedo, R. Rafanell, V. Rebello, M.S. Sousa-Baena, and E. Torres. 2014. Supporting biodiversity studies with the EUBrazilOpenBio Hybrid Data Infrastructure. Concurr Comp-Pract E doi:10.1002/cpe.3238.

Anderson R.P., D. Lew, and A.T. Peterson. 2003. Evaluating predictive models of species' distributions: criteria for selecting optimal models. Ecol Model 162:211-232.

Barve, N., V. Barve, A. Jimenez-Valverde, A. Lira-Noriega, S.P. Maher, A.T. Peterson, J. Soberón, and F. Villalobos. 2011. The crucial role of the accessible area in ecological niche modeling and species distribution modeling. Ecol Model 222:1810-1819.

Boston, A.N., and D.R.B. Stockwell. 1995. Interactive species distribution reporting, mapping and modelling using the World Wide Web. Comput Networks ISDN 28(1-2):231-238.

Breiman, L. 2001. Random Forests. Mach Learn 45(1):5-32.

Cavner, J.A., A.M. Stewart, C.J. Grady, and J.H. Beach. 2011. An innovative Web Processing Services based GIS architecture for global biogeographic analyses of species distributions. in: FOSS4G 2011 Proceedings, 10:15-25.

Couvares, P., T. Kosar, A. Roy, J. Weber, and K. Wenger. 2007. Workflow in Condor. Workflows for e-science (Eds: I. Taylor, E. Deelman, D. Gannon, M. Shields). Springer Press. ISBN: 1-84628-519-4.

Diniz-Filho, J.A.F., L.M. Bini, T.F. Rangel, R.D. Loyola, C. Hof, D. Nogués-Bravo, and M.B. Araújo. 2009. Partitioning and Mapping Uncertainties in Ensembles of Forecasts of Species Turnover Under Climate Change. Ecography 32:897-906.

Elith, J., C.H. Graham, R.P. Anderson, M. Dudik, S. Ferrier, A. Guisan, R.J. Hijmans, F. Huettmann, J.R. Leathwick, A. Lehmann, J. Li, L.G. Lohmann, B.A. Loiselle, G. Manion, G. Moritz, M. Nakamura, Y. Nakazawa, J.McC. Overton, A.T. Peterson, S.J. Phillips, K. Richardson, R. Scachetti-Pereira, R.E. Schapire, J. Soberón, S. Williams, M.S. Wisz, and N.E. Zimmermann. 2006. Novel methods improve prediction of species' distributions from occurrence data. Ecography 29:129-151.

Farber, O., and R. Kadmon. 2003. Assessment of alternative approaches for bioclimatic modeling with special emphasis on the Mahalanobis distance. Ecol Model 160:115-130.

Feeley, K.J., and M.R. Silman. 2010. Modelling the Responses of Andean and Amazonian Plant Species to Climate Change: The Effects of Georeferencing Errors and the Importance of Data Filtering. J Biogeogr 37:733-740.

Herring, J.R., ed. OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture, version 1.2.1, section 9. Accessed September 2, 2014. http://portal.opengeospatial.org/files/?artifact_id=25355.

Hirzel, A.H., J. Hausser, D. Chessel, and N. Perrin. 2002. Ecological-niche factor analysis: How to compute habitat-suitability maps without absence data? Ecology 83(7):2027-2036.

Kai, H., J. Dongarra, and G.C. Fox. 2011. Distributed and Cloud Computing: From Parallel Processing to the Internet of Things. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Lezzi, D., R. Rafanell, E. Torres, R. Giovanni, I. Blanquer, and R.M. Badia. 2013. Programming Ecological Niche Modeling Workflows in the Cloud, in: 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Barcelona. pp. 1223-1228.

Lorena, A.C., L.F.O. Jacintho, M.F. Siqueira, R. Giovanni, L.G. Lohmann, A.C.P.L.F. Carvalho, and M. Yamamoto. 2011. Comparing Machine Learning Classifiers in Potential Distribution Modelling. Expert Syst Appl 38(5):5268-5275.

Nix, H.A. 1986. A biogeographic analysis of Australian elapid snakes. Atlas of Australian elapid snakes (ed. by R. Longmore), pp. 4-15. Australian Flora and Fauna Series 7, Australian Government Publishing Service, Canberra.

Marmion, M., M. Parviainen, M. Luoto, R.K. Heikkinen, and W. Thuiller. 2009. Evaluation of Consensus Methods in Predictive Species Distribution Modelling. Divers Distrib 15(1):59-69.

Muñoz, M.E.S., R. Giovanni, M.F. Siqueira, T. Sutton, P. Brewer, R.S. Pereira, D.A.L. Canhos, and V.P. Canhos. 2011. openModeller: a generic approach to species' potential distribution modelling. Geoinformatica 15:111-135.

Pahwa, J.S., P. Brewer, T. Sutton, C. Yesson, M. Burgess, X. Xu, A.C. Jones, R.J. White, W.A. Gray, N.J. Fiddian, F.A. Bisby, A. Culham, N. Caithness, M. Scoble, P. Williams, and S. Bhagwat. 2006. Biodiversity World: A Problem-Solving Environment for Analysing Biodiversity Patterns, in: Proc. 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006), Singapore.

Peterson, A.T., M. Papeş, and J. Soberón. 2008. Rethinking receiver operating characteristic analysis applications in ecological niche modeling. Ecol Model 213(1):63-72.

Phillips, S.J., R.P. Anderson, and R.E. Schapire. 2006. Maximum entropy modelling of species geographic distributions. Ecol Model 190:231-259.

Schölkopf, B., J. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. 2001. Estimating the support of a high-dimensional distribution. Neural Comput 13:1443-1471.

Segurado, P. and M.B. Araújo. 2004. An Evaluation of Methods for Modelling Species Distributions. J Biogeogr 31:1555-1568.

Skøien, J.O., M. Schulz, G. Dubois, I. Fisher, M. Balman, I. May, and É.Ó. Tuama. 2013. A Model Web approach to modelling climate change in biomes of Important Bird Areas. Ecol Inform 14:38-43.

Soberón, J. and A.T. Peterson. 2005. Interpretation of models of fundamental ecological niches and species' distributional areas. Biodiversity Informatics 2:1-10.

Stein, L.D. 2008. Towards a Cyberinfrastructure for the Biological Sciences: Progress, Visions and Challenges. Nat Rev Genet 9:678-688.

Tarassenko, L. 1998. A Guide to Neural Computing Applications. Arnold, London, 139 pp.

Taylor, I., M. Shields, I. Wang, and O. Rana. 2003. Triana Applications within Grid Computing and Peer to Peer Environments. J Grid Comput 1(2):199-217.

Thain, D., T. Tannenbaum and M. Livny. 2005. Distributed Computing in Practice: the Condor Experience. Concurr Comp-Pract E 17(2-4):323-356.

Wieczorek, J., D. Bloom, R. Guralnick, S. Blum, M. Döring, R. Giovanni, T. Robertson, and D. Vieglais. 2012. Darwin Core: An Evolving Community-Developed Biodiversity Data Standard. Plos One, 7:e29715.

Wisz, M.S., R.J. Hijmans, J. Li, A.T. Peterson, C.H. Graham, and A. Guisan. 2008. Effects of Sample Size on the Performance of Species Distribution Models. Divers Distrib 14:763-773.

Wolstencroft, K., R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A.N. de la Hidalga, M.P.B. Vargas, S. Sufi, and C. Goble. 2013. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Res 41(1):557-561.

World Wide Web Consortium, 2001. Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. http://www.w3.org/TR/wsdl.

World Wide Web Consortium, 2004a. Web Services Architecture, W3C Working Group Note 11 February 2004. http://www.w3.org/TR/ws-arch/.

World Wide Web Consortium, 2004b. XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-0/.

World Wide Web Consortium, 2006. Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation 16 August 2006. http://www.w3.org/TR/2006/REC-xml11-20060816/.

World Wide Web Consortium, 2007. SOAP Version 1.2 Part 0: Primer (Second Edition). W3C Recommendation 27 April 2007. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.