An Autonomic Parallel Strategy for the Projection of Ecological Niche Models in Heterogeneous Computational Environments

Fernanda G.O. Passos(✉) and Vinod E.F. Rebello

Instituto de Computação – Universidade Federal Fluminense (UFF),
Niterói, RJ, Brazil
{fernanda,vinod}@ic.uff.br

Abstract. Ecological Niche Modelling (ENM) is an important process to help ecologists understand and predict the potential geographic distribution of species. In addition to creating correlative models for each species, the projection of the model onto a geographical environment is an essential step in the process to visualize suitable habitats. Given the demand for improved precision and the need to address wider geospatial domains, using larger data sets means that these methods incur increasingly higher processing, memory and I/O demands. This paper proposes a new parallel algorithm for the projection stage of a popular ENM tool. Although the characteristics of ENM already allow this tool to make use of heterogeneous computing environments, a new algorithm has been designed to be autonomic and capable of reconfiguration to respond better to the resource capacities available. The proposal has been compared with the default sequential implementation and a parallel MPI version, both distributed with the ENM tool. An empirical analysis reveals gains from 109 % to 742 % in terms of performance and improved scalability with efficiencies above 81 % in evaluations with up to 128 processors.

1 Introduction

An ecological niche can be defined as the set of environmental conditions for a species to survive and maintain viable populations over time [4]. Ecological Niche Modelling (ENM) is a common procedure most often used in macroecology and biogeography to determine the geographical extension of species distributions. These correlative ecological niche models are generated by relating locations where a given species is known to occur with the environmental conditions at that locale that might influence their distribution [17]. ENM provides a powerful mechanism to predict potential species distribution in distinct geographical and temporal contexts, as well as to study another aspects of evolutionary biology and ecology. ENM has been widely used in various situations such as: searching for rare or endangered species; identifying suitable areas for the (re-)introduction of species; forecasting the impact of climate change on biodiversity; helping in conservation planning and delimitation and evaluation of protected areas; preventing the spread of invasive species; identifying geographical and ecological

P.-F. Dutot and D. Trystram (Eds.): Euro-Par 2016, LNCS 9833, pp. 363–375, 2016.
DOI: 10.1007/978-3-319-43659-3_27

aspects of disease transmission; guiding biodiversity field surveys; among other important applications [12].

ENM applications combine information about the occurrence of species (biotic) with environmental databases (abiotic) in the form of geo-referenced raster layers (such as temperature, rainfall and salinity) to generate potential distribution models. This process includes a combination of 3 dependent steps: model creation, testing and projection. The models are usually generated by statistical techniques, such as maximum entropy, or by machine learning techniques such as artificial neural networks [12]. One of the most widely adopted ENM tools is *openModeller* [7], which offers a choice of 15 modelling algorithms. Although this tool has been adopted by various large scale e-infrastructure projects [1,2,5] to provide ENM services in the cloud, the software tool was designed principally for a single server.

This paper addresses the parallelisation of the costly stage that projects the ecological niche model into the chosen geospatial domain given a set of environmental conditions at the time of interest. *openModeller* is distributed with a default sequential algorithm and an optional parallel version for model projection, but neither implementation takes into sufficient consideration the possibility of having to run on heterogeneous resources or in shared dynamic environments, like clouds. The goal of this paper is to propose an adaptive parallel algorithm for ENM projection that it is able to manage its own execution in any one of these three common types of environments. This algorithm has been integrated with the EasyGrid application management system (EasyGrid AMS) [10] to harness the available environment more efficiently by making the projection autonomic.

The paper is structured as follows. Section 2 presents the openModeller tool and describes the projection of an ecological niche model, as well as the default sequential implementation adopted in openModeller. The existing parallel MPI version is explained in Sect. 3. In Sect. 4, the presentation of the EasyGrid AMS and its programming model is followed by that of the proposed new autonomic algorithm for ENM projection. Section 5 supports the proposal through experimental evaluations, with some conclusions being drawn in Sect. 6.

2 OpenModeller and ENM Projection

OpenModeller (OM), an ENM tool developed by a Brazilian Reference Centre for Environmental Information (*Centro de Referência em Informação Ambiental* – CRIA) together with national and international partners [7], is widely used in the biogeographical and ecological research communities [3,13,18]. OM produces species potential distribution models and includes mechanisms: to read environmental data and species occurrence points; to select environmental layers on which the model may be based; to create a fundamental niche model; project models in an environmental scenario and produce detailed graphical images in several formats.

Several algorithms are available as plug-ins for model creation [7] including: BIOCLIM (Bioclimatic envelopes), GARP (Genetic Algorithm for Ruleset Production), GARP_BS (GARP with Best Subsets), DG_GARP (Desktop GARP), DG_GARP_BS (Desktop GARP_BS), ENVSCORE (Envelope Score), ENVDIST (Environmental Distance), RF (Random Forests), MAXENT (Maximum Entropy), NICHE_MOSAIC (Niche Mosaic), and SVM (Support Vector Machines).

Effort has been invested in distributing ENM workflows consisting of hundreds or more instances of dependent tasks to create, test and project each model [1,2,6]. However, since ENM projection often involves large amounts of data, our work seeks additional gains by exploring parallelism in individual projection tasks. The OM tool presents an optional MPI implementation for the ENM projection [11] using the traditional parallel programming model described in Sect. 3.

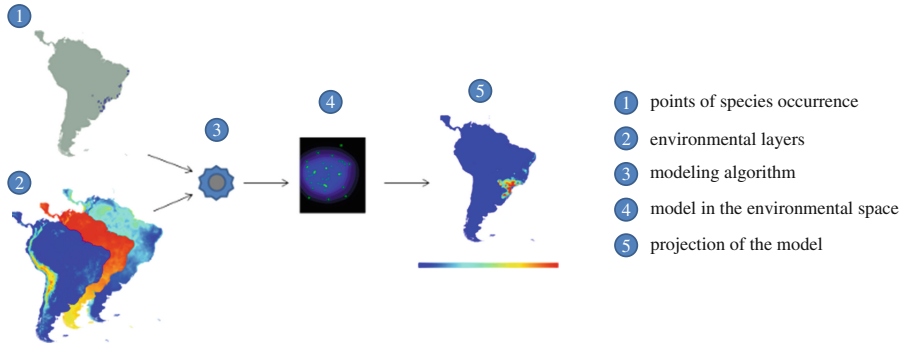

Fig. 1. Example of modelling and projection of an ecological niche [11]. (Color figure online)

OM project is the OpenModeller mechanism to project the distribution models and generate a high resolution geographical 2D rectangular image of a region bounded by coordinates (a, b) and (c, d) where $a < c$ and $b < d$. A geospatial mask can be selected by the user to project a model in to arbitrary areas within the defined region. The value of each coordinate point represents the probability of the environmental conditions at that locale being hospitable for a given species. Figure 1 shows the modelling and projection of an ecological niche, where item 5 indicates the result of the projection of a model. The colour scale that varies from blue to red represents the suitability of a region (red means high and blue means low). The scientist is responsible for selecting the environmental data layers for which the prediction should be based.

Algorithm 1 briefly presents the sequential implementation of a model projection of a ecological niche. The algorithm inputs are the model, generated previously in a modelling stage by a chosen OM algorithm described in Sect. 2, and the environmental data for the region of interest. The output image contains the map with the

predicted probabilities. The following steps are executed by the algorithm for each point (x, y) where $a \leq x \leq c$ and $b \leq y \leq d$: Line 3 obtains the necessary environmental layer data for (x, y); Line 4 applies the model to the coordinate (x, y) considering the environmental data and calculates a probability p; finally, Line 5 converts p to a image value (RGB) and writes it in the file at position (x, y).

Algorithm 1. The sequential algorithm for ENM projection.

Input: *model* - model previously generated by an OM algorithm.
env_data - environmental data.
Output: *map* - file which contain the projection.

```
1 for x ← a to c − 1 do
2     for y ← b to d − 1 do
3         env ← env_data at position (x, y)
4         apply model with env and put the value in p
5         write p in map at position (x, y)
```

3 Original MPI Version for OM Projection

The OM tool, currently, has a MPI implementation for the ENM projection. This MPI version uses a parallel programming model similar to master-worker. In addition to the worker processes that compute the projection, there are two master processes instead of one to improve performance. We refer to *masterD* as the process that distributes on-demand ranges of coordinates for each worker to evaluate and *masterR* the process that receives the partial solutions from each worker and writes them to the output image file.

The *masterD* algorithm partitions the total number of points to be projected into fixed-size blocks (the default is 30,000 points). The worker processes request blocks from the *masterD* and compute the species distribution potential for each coordinate within the block. The result is a partial projection block that must be sent to *masterR* which in turn must aggregate the projected blocks and write them to the output file.

4 Autonomic ENM Projection with EasyGrid AMS

The EasyGrid Application Management System (AMS) [10] is an example of an application-centric middleware that is responsible for managing an application's execution on the computational resources available. The EasyGrid philosophy promotes the idea that an AMS should employ autonomic management strategies tuned for each application instead of single resource-centric management approach if one wishes to improve efficiency and performance.

The EasyGrid AMS offers some autonomic features, with different implementations strategies for different classes of MPI applications: a three-level hierarchical dynamic scheduler [8], providing *self-optimization*, and; a fault-tolerant mechanism capable of detecting and recovering application failures [16], providing *self-healing*. *Self-configuration* allows an application to change its configuration dynamically during execution. The implementation of this feature is heavily dependent on characteristics of the application that is to become autonomic. Previous studies with other MPI applications have shown more than satisfactory results using this middleware [9,14–16].

Key to this success is the adoption of an alternative programming model, 1Ptask [9], to the traditional MPI one (referred to here as 1Pproc). The 1Pproc model considers a single process per processor and thus each process is relatively coarse-grained. The 1Ptask model considers each process to be a finer grained task and consequently, the total number of processes tends to be relatively larger than the number of processors. While more processes might mean higher overheads, it also provides greater flexibility to improve load balancing, for example.

4.1 A Self-Configuring Projection Implementation

The original MPI version suffers from two drawbacks: (1) the algorithm is centralized and the messages sent and received by the two masters create a bottleneck, impairing performance; (2) the programming model 1Pproc may not be the best alternative for large-scale distributed environments. Previous work has shown the 1Ptask model to be more appropriate for multi-core servers environments that are typically dynamic, heterogeneous and/or are shared, such as those commonly used in grid and cloud computing. The proposal of an autonomic algorithm with the EasyGrid AMS necessitates: the use of the 1Ptask model; the definition of application tasks that can dynamically self-configure, and; the use of the EasyGrid AMS for the autonomic management of the tasks.

Another issue related to the programming model is the division of the ENM projection domain in fixed-size blocks as presented in the original MPI version. Figure 2 shows the variation in execution times for each block (one map line) of the ENM projection using the BIOCLIM algorithm. One can see a reasonable variation in the execution times, in this case, with an average of 0.28 s and standard deviation of 0.15. The longest block is about to 32 times slower than the fastest. Although the blocks have a fixed size, their processing costs can be different. Furthermore, for example in the case of the first and last lines of the map domain, the processing times can be so low that the communication times may exceed the processing time, causing further losses in performance.

Given that one cannot assume that the processing capacity is the same for all available resources and, from Fig. 2, the task granularities for fixed-sized blocks are not going to be similar amongst them, the proposed EasyGrid AMS version adopts variable block sizes that changes dynamically over time. At first, the workload is divided equally to tasks across all processors cores independently of their processing capacities. Under the 1Ptask programming model, each task is implemented as an MPI process. In this application, the block size is set initially

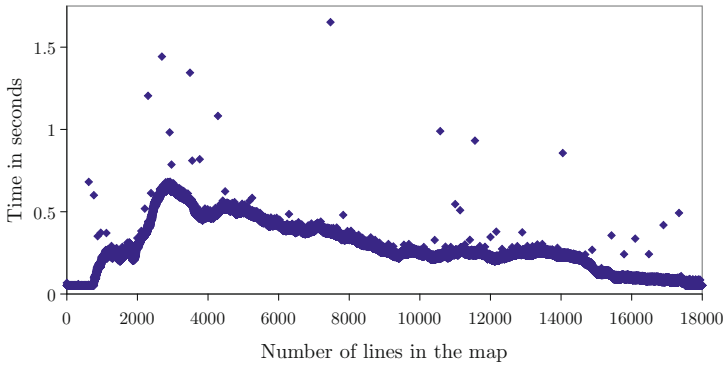

Fig. 2. Execution times of each line of the ENM projection.

to the total number of map lines divided by the number of processor cores. During execution, the tasks calculate the projection of the block in the same way as the sequential algorithm, but with one subtle difference, the task execution time is limited by a timer *timeout*. Should the task's current processing time exceed the *timeout* value, a number of new tasks are created to continue the projection of the remaining portion of the block and the original task then terminates. For each level of task creation (assume the initial tasks to be in level 0, their children belong to level 1 and so on recursively), the task's *timeout* duration is reduced. Effectively, the block size allocated to a task changes dynamically, depending on the resource location and the application's progress. One goal is for later tasks to become finer in granularity so that the EasyGrid AMS can distribute them among the available processors more efficiently [8].

The Algorithm 2 highlights the execution steps of each task of the new ENM projection, called *partition*. As input, this algorithm receives the model, the environmental data, the Y dimension of the original map, the block and the *timeout* value. Each task has its output identified by map_{l_i}, where l_i represents the initial line of the block. The projection of the allocated block is processed line by line (Lines 1) with Lines 2 to 5 evaluating the coordinates in a given block line in the same way as Algorithm 1. Now, in the Line 6, after completing the projection of a map line, the timer *timeout* is verified. If the time limit has been exceeded, the number of new tasks *ntasks* to be created is estimated, in Line 6, based on the processing rate during the last time period. In a total of Dim_x map lines, if x is the current line map that was calculated in *timeout* seconds, to calculate the remaining $Dim_x - x$ we need $\frac{(Dim_x - x)}{x}$ tasks. Line 8 determines a shorter *timeout* value for the new tasks while in Lines 9 to 11, the remaining unprocessed portion of the block is split and sent to the *ntasks* tasks. The new *timeout* value is the quotient of old *timeout* (starting from 10 s) divided by the level plus 1. If the value is lower than 1, the new *timeout* is 1 s. Unfortunately by the end of this process, the parallelisation of ENM projection means that the image is made up of several sub-maps each stored in a separate output file (one

Algorithm 2. Algorithm for the *partition* tasks of the ENM projection with EasyGrid AMS.

Input: *model* - model previously generated by an OM algorithm.
env_data - environmental data.
Dim_y - dimension Y of the complete map.
(l_i, l_j) - block to be calculated.
timeout - initial timer.
Output: map_{l_i} - partial maps.

```
1  for x ← l_i to l_j do
2      forall the y ← 0 to Dim_y do
3          env ← env_data at position (x, y)
4          Apply model in env and put the value in p
5          Write p in map_{l_i} at position (x − l_i, y)
6      if timeout is over then
7          Calculate ntasks
8          Calculate the new timeout
9          Split remaining block (x + 1, l_j) into ntasks and put each sub-block in S
10         forall the k ∈ S do
11             Create task with arguments model, env_data, Dim_y, k and timeout
```

per task). Thus, a *merge* task is required to combine the files and generate a final image containing the projection map. At present, this is implemented using a sequential algorithm.

5 Experimental Analysis

Three sets of experiments were carried out. The first aims to compare the two parallel implementations (the original MPI version that distributed fixed block sizes on demand with the proposed autonomic version with EasyGrid AMS) using a variety of OM model algorithms at small scales (with up to 24 CPUs). The second aims to evaluate the scalability of the proposal at a larger scale, obviously. The final experiment evaluates the efficiency of the proposal in a dynamic heterogeneous environment where external loads were introduced periodically. The last two experiments used a cluster of 16 8-core processors each, totalling 128 CPUs.

5.1 Experiment 1: Small Scale Performance

In this experiment, p initial worker processes are considered for both projection implementations running on p CPUs. For the original MPI version, there are actually $p+2$ processes in total (plus 2 masters). The projection modelling algorithms BIOCLIM, ENVSCORE, GARP, DG_GARP, DG_GARP_BS, MAXENT, GARP_BS, SVM, RF and NICHE_MOSAIC are selected for this experiment.

Table 1. Speed-up obtained by the original MPI version and the autonomic version.

Algorithm	Version	Speed-up				
		4	8	12	16	24
BIOCLIM	Original MPI	2.58	4.29	5.54	5.45	5.70
	EasyGrid	3.67	7.15	9.96	13.03	19.21
ENVSCORE	Original MPI	2.65	4.20	5.49	4.86	6.18
	EasyGrid	3.32	6.48	9.35	12.67	18.28
GARP	Original MPI	3.12	4.90	6.85	8.25	8.03
	EasyGrid	3.85	7.57	11.21	14.59	21.58
DG_GARP	Original MPI	2.90	4.74	6.17	7.59	7.49
	EasyGrid	3.67	7.11	10.01	13.10	19.73
DG_GARP_BS	Original MPI	3.11	5.80	7.26	9.68	10.49
	EasyGrid	3.78	7.65	10.87	14.32	21.48
MAXENT	Original MPI	3.46	6.07	7.84	10.38	11.45
	EasyGrid	3.88	7.62	10.99	14.61	21.70
GARP_BS	Original MPI	2.98	5.42	7.50	9.46	10.33
	EasyGrid	3.85	7.60	10.96	14.52	21.77
SVM	Original MPI	3.32	5.63	8.01	10.54	12.59
	EasyGrid	3.90	7.71	11.17	14.85	22.24
RF	Original MPI	3.28	4.99	7.97	10.15	11.30
	EasyGrid	3.59	7.14	10.35	13.78	20.58
NICHE_MOSAIC	Original MPI	3.28	5.68	7.70	7.49	4.63
	EasyGrid	3.94	7.81	10.10	11.48	12.74

Table 2. Execution Time in seconds for *partition* and *merge* operations, when using the ENVDIST and SVM modelling algorithms.

	ENVDIST			SVM		
p	*partition*	*merge*	speed-up	*partition*	*merge*	speed-up
24	70,758.88	75.93	22.85	1,186.28	41.18	24.05
48	35,316.90	85.26	45.72	614.53	59.50	43.80
96	17,758.27	88.44	90.69	313.29	47.33	81.86
128	13,390.62	98.59	119.99	238.36	46.59	103.60

Table 1 presents a comparison of the speed-ups, in relation to sequential algorithm, obtained by original MPI approach and autonomic approach with EasyGrid AMS for the ENM projection. Using different OM modelling algorithms and varying the number of processors (from 4 to 24), the autonomic version presented significantly better speed-ups and, for the most of modelling algorithms, the value was close to the number of processors used.

Figure 3 provides a better visualisation of the improvements in speed-up. Each bar group with labels 4, 8, 12, 16 and 24 processors represents the gain (in percent) of the autonomic EasyGrid AMS version over the original MPI one. As seen in Table 1, the autonomic version always achieved higher speed-ups. For most algorithms, as the number of processors increases, so does the gain in relation to the original MPI version. For BIOCLIM, with 24 processors, the parallel projection with EasyGrid AMS is more than 3 times better than the existing solution.

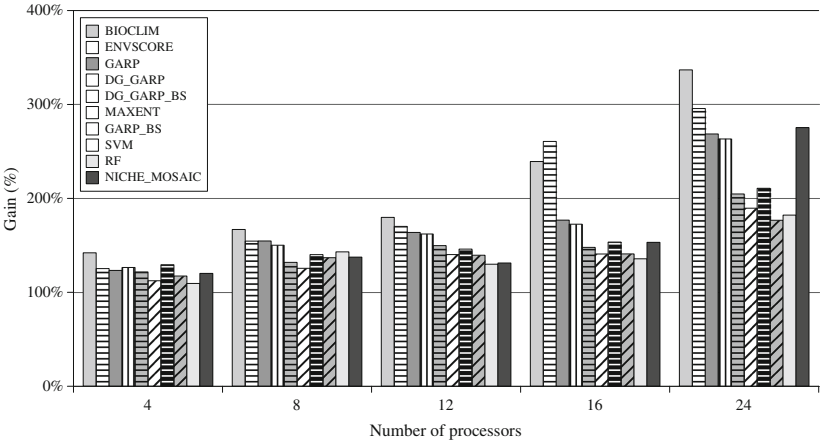

Fig. 3. Performance gain of the autonomic approach over the original MPI versions.

Scalability is a weakness of the original MPI version. For some algorithms, the speed-up values continue to increase as the number of processors grows, while for others, already with 24 CPUs the speed-ups taper off or fall (as in the case of DG_GARP and NICHE_MOSAIC). To obtain good scalability, algorithm designers should aim for near linear speed-up, especially at relatively lower processor counts. In the case of the EasyGrid AMS version, while this property is better it is not perfect, for example, in the case of NICHE_MOSAIC, but in part due to the overhead of the unparallelised merge operation.

5.2 Experiment 2: Larger Scale Homogeneous Performance

From the first experiment, the better performance of the autonomic parallel ENM version with the EasyGrid AMS over the original MPI one was not only clear but also indicated that the former approach was more scalable. This second experiment aims analyse further the scalability of the EasyGrid AMS version. The two modelling algorithms with the longest execution times were selected: ENVDIST and SVM. The sequential implementation of ENM Projection with ENVDIST, for example, has an execution time of almost 20 days. The number of processors varied from 24, 48, 96 to 128 CPUs.

The number of tasks created by the autonomic version during execution can be quite high, relatively speaking. The total amount of tasks for the instance SVM is approximately 5,400 for all CPUs numbers used, with low variance. For the instance ENVDIST, the number is approximately 16,000 for all numbers of CPUs used, again with low variance. For both, the number of tasks keeps more or less static as the processors increase. For this reason, the merge time has a low variability as approximately the same number of sub-maps are merged.

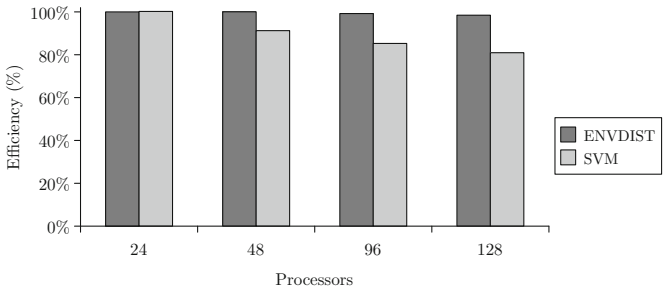

Fig. 4. Efficiency of the autonomic parallel ENM projection with EasyGrid AMS for the algorithms ENVDIST and SVM using 24, 48, 96 and 128 CPUs.

Table 2 presents the execution times, in seconds, for the *partition* tasks and *merge* tasks of the autonomic ENM projection with EasyGrid AMS, separately, for both the ENVDIST and SVM model algorithms. There is also a column that represents the speed-up of the combined *partition* plus *merge* execution. We consider the total execution time to be the sum of the two but are interested in the *partition* execution since it dwarfs the *merge* time. The sequential *merge* algorithm is only used to concatenate the images generated by the *partition* tasks but as in the case of Amdahl's Law will eventually limit the speedup.

The efficiency of the proposal for the 8 ENM projections (i.e. the partition and merge tasks) is indicated in Fig. 4.

The self-configuring and self-optimisation abilities of the autonomic version combined allow significantly better performance and good scalability for the ENM projections with costly OM model algorithms. The self-configuring mechanism ensures an appropriate degree of parallelism by dynamically creating progressively finer grained tasks taking into consideration the application's behaviour. This over-provisioning of tasks is then managed by the self-optimisation through the dynamic scheduler of the EasyGrid AMS. Together, these abilities explore the characteristics of the ENM projection and the execution environment, achieving acceptable efficiency levels for large scale computing.

5.3 Experiment 3: Larger Scale Heterogeneous Performance

The aim of this experiment is to briefly highlight the difference in performance of the original MPI and EasyGrid AMS approaches in a heterogeneous or dynamic

Table 3. Results with a simulated shared environment.

	Original MPI version			EasyGrid AMS version		
Algorithm	Dedicated	With load	Slowdown (%)	Dedicated	With load	Slowdown (%)
SVM	1,156.40	2,446.02	111.5	238.36	329.66	38.30
ENVDIST	24,611.82	46,490.34	88.89	13,390.62	17,840.64	33.23

shared environment. Using the previous dedicated homogeneous 128-core cluster, CPU-intensive loads were introduced to compete with the application. These independent externals loads are not constant – they are effectively active for intercalated periods of processing (20 s) and idleness (20 s) on each CPU-core. Thus, the full capacity of the cluster is available to the OM projection applications for half the time, the rest of the time they will compete, obtaining only half the capacity.

Table 3 presents execution times in seconds for the SVM and ENVDIST modelling algorithms again. For each approach, the execution times in a dedicated environment and in this shared environment were measured. The column *Slowdown* indicates the relative loss in performance due to resource sharing (or dynamic heterogeneity), *i.e.*, how much longer the applications take to execute. These results indicate significantly better performance is obtained by the autonomic version in a shared configuration. Furthermore, while this version is 4.85 and 1.84 times faster in a static homogeneous environment, this shoots up to 7.42 and 2.61 times in this dynamic heterogeneous one, for OM projection with the SVM and ENVDIST modelling algorithms, respectively.

6 Conclusion

This paper proposed a new autonomic approach for the projection of the ecological niches in geospatial domains. This kind of application appears to be easily parallelisable since data processing can be decomposed into independent tasks. However traditional approaches may not be as efficient as expected in current execution environments due to workload granularities of unknown size.

The current MPI approach distributed with the biodiversity tool *openModeller* is based on the master-worker model and uses fixed-block partitioning allowing on-demand load balancing between tasks executing under a 1Pproc model. The new autonomic approach proposes the use of the 1Ptask model with a dynamic partition that generates tasks with variable block sizes. This partitioning is achieved by a simple self-configuring strategy that takes into consideration the execution behaviour. Tasks are created to process blocks on the fly, each being managed by the EasyGrid AMS, which provides self-optimisation.

Results showed that the proposed algorithm presented better performance and scalability in all experiments when compared to the original traditional algorithm. With 24 processors, speed-ups with the EasyGrid AMS version are between 109 % to 337 % higher than those of the original version. The proposed

approach scales up to 128 processors, obtaining efficiency values of at least 81 % even though sub-maps are still merged sequentially. Future work includes substituting this with a parallel version. Finally, in heterogeneous or shared environments, the improvements in performance are even greater.

References

1. Amaral, R., Badia, R.M., Blanquer, I., Braga-Neto, R., Candela, L., Castelli, D., Flann, C., De Giovanni, R., Gray, W.A., Jones, A., et al.: Supporting biodiversity studies with the EUBrazilOpenBio hybrid data infrastructure. Concurrency Comput. Pract. Experience **27**(2), 376–394 (2015)
2. EUBrazil Cloud Connect Project: EUBrazil Cloud Connect (2016). http://www.eubrazilcloudconnect.eu. Accessed Feb 2016
3. Geller, G.N., Melton, F.: Looking forward: applying an ecological model web to assess impacts of climate change. Biodiversity **9**(3–4), 79–83 (2008)
4. Grinnell, J.: Field tests of theories concerning distributional control. Am. Nat. **51**(602), 115–128 (1917)
5. Leidenberger, S., De Giovanni, R., Kulawik, R., Williams, A.R., Bourlat, S.J.: Mapping present and future potential distribution patterns for a meso-grazer guild in the baltic sea. J. Biogeogr. **42**(2), 241–254 (2015)
6. Lezzi, D., Rafanell, R., Torres, E., Giovanni, R., Blanquer, I., Badia, R.: Programming ecological niche modeling workflows in the cloud. In: 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 1223–1228, March 2013
7. Muñoz, M.E.S., De Giovanni, R., Siqueira, M.F., Sutton, T., Brewer, P., Pereira, R.S., Canhos, D.A.L., Canhos, V.P.: Openmodeller: a generic approach to species' potential distribution modelling. GeoInformatica **15**(1), 111–135 (2011)
8. Nascimento, A., Sena, A., Boeres, C., Rebello, V.E.F.: Distributed and dynamic self-scheduling of parallel MPI grid applications. Concurrency Comput. Pract. Experience **19**(14), 1955–1974 (2007)
9. Nascimento, A., Sena, A., da Silva, J., Vianna, D.Q.C., Boeres, C., Rebello, V.E.F.: On the advantages of an alternative MPI execution model for grids. In: CCGRID 2007, pp. 575–582. IEEE Computer Society, Rio de Janeiro (2007)
10. Nascimento, A., Sena, A., da Silva, J., Vianna, D.Q.C., Boeres, C., Rebello, V.E.F.: Autonomic application management for large scale MPI programs. Int. J. High Perform. Comput. Networking **5**(4), 227–240 (2008)
11. Team, O.: Openmodeller webpage (2016). http://openmodeller.sourceforge.net/. Accessed Feb 2016
12. Peterson, A.T., Soberón, J., Pearson, R.G., Anderson, R.P., Martínez-Meyer, E., Nakamura, M., Araújo, M.B.: Ecological niches and geographic distributions (MPB-49). Princeton University Press (2011)
13. Ramachandra, T., Kumar, U., Aithal, B.H., Diwakar, P., Joshi, N.: Landslide susceptible locations in western ghats: prediction through openmodeller. In: Proceedings of the 26th Annual In-House Symposium on Space Science and Technology, pp. 65–74. Indian Institute of Science, Bangalore, Indian, January 2010
14. Ribeiro, F., Nascimento, A., Boeres, C., Rebello, V., Sena, A.: Autonomic malleability in iterative MPI applications. In: 25th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 192–199, October 2013

15. Sena, A., Nascimento, A., Boeres, C., Rebello, V.: Easygrid enabling of iterative tightly-coupled parallel MPI applications. In: International Symposium on Parallel and Distributed Processing with Applications (ISPA 2008), pp. 199–206, December 2008
16. da Silva, J.A., Rebello, V.E.F.: Low cost self-healing in MPI applications. In: Cappello, F., Herault, T., Dongarra, J. (eds.) PVM/MPI 2007. LNCS, vol. 4757, pp. 144–152. Springer, Heidelberg (2007)
17. Soberón, J., Peterson, A.T.: Interpretation of models of fundamental ecological niches and species distributional areas. Biodivers. Inform. **2**, 1–10 (2005)
18. Zarco-González, M.M., Monroy-Vilchis, O., Alaníz, J.: Spatial model of livestock predation by jaguar and puma in mexico: conservation planning. Bio. Conserv. **159**, 80–87 (2013)