Paralelização do *Framework* Model-R de Modelagem de Nichos Ecológicos com a Plataforma Apache Spark

Matheus Machado da Rosa Albuquerque[1,2]
Luiz M. R. Gadelha Jr.[2]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ)
Petrópolis – RJ – Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ – Brasil

{matheusm,lgadelha}@lncc.br

Resumo. *Este trabalho tem como objetivo avaliar o desempenho computacional do framework Model-R de modelagem de nichos ecológiocs no modelo de programação Spark para processamento de dados massivos (*Big Data*) em uma plataforma de supercomputação. Com o crescimento exponencial dos dados ecológicos e ambientais, torna-se necessário que ferramentas de modelagem de nichos ecológicos, como o Model-R, estejam preparadas para processar tais dados de forma escalável, sendo capazes de analisá-los em um tempo hábil. Nesta pesquisa, o Model-R, originalmente implementado em R com a biblioteca Snowfall, foi parcialmente portado para a plataforma Spark e sua avaliação no supercomputador Santos Dumont está em andamento.*

1. Introdução

O nicho ecológico [Drake et al. 2006] é um conceito biológico fundamental. A modelagem de nicho pode ser entendida como uma forma de uso e combinação de dados de ocorrência de espécies e variáveis preditivas para inferir projeções derivadas de modelos estatísticos e teóricos. Entretanto, com o avanço dos métodos empenhados na coleta e análise de dados ecológicos, o volume de dados tal como a precisão exigida nos estudos tem crescido gradativamente de forma a não escalar bem com o funcionamento das ferramentas atuais de modelagem de nicho ecológico, exigindo cada vez mais eficiência nas execuções.

O Model-R [Sánchez-Tapia et al. 2018] é um *framework* desenvolvido em linguagem R que nasceu com o objetivo de agrupar ferramentas de modelagem de nichos ecológicos em um único ambiente integrado e amigável onde fosse possível automatizar as principais etapas da análise, como o processo de modelagem e a recuperação dos dados de ocorrência de espécies. Na versão mais atualizada da ferramenta foram empregadas técnicas de paralelismo tradicionais. Como a aplicação de uma função sobre os elementos de uma lista de forma paralela, com o intuito de melhorar o desempenho. Para isto, foi utilizado o pacote *Snowfall* [Knaus 2010] e foram feitas alterações para que fosse empregado o paralelismo por espécie, onde a análise é particionada de acordo com a quantidade de espécies, assim como por algoritmo de modelagem.

2. Metodologia

Para o desenvolvimento do presente estudo tem-se como estratégia utilizar técnicas de paralelismo computacional através da plataforma Apache Spark [Zaharia et al. 2016] usufruindo de sua API em linguagem R. O Spark nasceu como um mecanismo unificado para processamento de dados distribuídos. A ferramenta possui um modelo de programação semelhante às funções de *Map* e *Reduce*, porém expande esse modelo com uma abstração para compartilhamento de dados chamada *Resilient Distributed Datasets* (RDDs). Para o desenvolvimento do código foi proposto o uso dos pacotes *SparkR* [Venkataraman et al. 2016] e *Sparklyr*[1], que fornecem suporte necessário para o uso e integração do Spark a linguagem R. Mesmo com o desempenho satisfatório, é necessário que a implementação original do Model-R esteja preparada para o futuro devido a crescente demanda por maiores velocidades de processamento e ao constante aumento na quantidade dos dados ecológicos.

Com a finalidade de implementação da estrutura para modelagem de nichos presente no Model-R de forma paralelizada foram levados em consideração algumas questões como: a natureza dos dados e também o plano de paralelismo a ser empregado. A principal limitação encontrada durante o estudo foi a questão da própria natureza diversificada dos dados necessários à modelagem, que variam de arquivos de variáveis preditivas em formato *Raster Stack*[2] até arquivos de ocorrência em formato CSV. Para isto, foram criados diversos *Data Frames* de acordo com a necessidade de cada etapa de processamento, convertendo e encapsulando os dados necessários a análise no próprio *Data Frame*[3]. Em virtude desse formato de dado ser nativo ao Spark, torna-se possível realizar toda manipulação e análise necessária via Spark, ganhando em performance. Foi possível desenvolver versões otimizadas dos algoritmos BIOCLIM e SVM aplicando técnicas de paralelismo computacional nas etapas de processamento de cada algoritmo. Outra questão que está sendo avaliada no trabalho é a viabilidade da execução do modelo de programação do Spark em plataformas para computação de alto desempenho (HPC), umas vez que essas possuem vastas quantidades de recursos computacionais na ordem de *petaflops* (10^{15} operações de ponto flutuante por segundo).

2.1. Adaptação do Spark ao Supercomputador Santos Dumont

Arquiteturas *Big Data*, conhecidas por serem do tipo *shared nothing*, dispõem de nós com suas próprias unidades de armazenamento. Para processamento nesse tipo de plataforma, os dados são distribuídos, processados e por fim agrupados para uma análise final. A vantagem desse tipo de arquitetura é o custo dos equipamentos, uma vez que *clusters* HPC necessitam de sistemas e *hardware* mais sofisticado, como interconexões de baixa latência (p.ex. *Infiniband*), enquanto *clusters Big Data* podem ser vistos como um conjunto máquinas comuns processando paralelamente dados particionados. A fim de viabilizar o Spark em um ambiente onde fosse possível proferir análises em larga escala, que exigem processamento massivo, bem como, usufruir do poder computacional disponível no supercomputador Santos Dumont, foi desenvolvido um *script*[4] de submissão de *jobs* para utilizar o Spark no SDumont. Esta implementação se mostrou promissora

[1]https://cran.r-project.org/web/packages/sparklyr/index.html
[2]https://www.rdocumentation.org/packages/raster/versions/2.6-7/topics/stack
[3]https://www.rdocumentation.org/packages/base/versions/3.5.3/topics/data.frame
[4]https://github.com/matheusalb31/Spark-on-Sdumont

frente a quantidade de memória RAM disponível nos nós do SDumont (64GB por nó do tipo *thin* e 6TB no nó do tipo *fat*). Assim, é possível que o Spark utilize o sistema de arquivos compartilhado paralelo Lustre como armazenamento e a memória como unidades de armazenamento não compartilhado, emulando o ambiente de um *Cluster Big Data*. O processamento paralelo utiliza os diversos nós computacionais, quem possuem de 24 (nós *thin*) a 240 (nó *fat*) núcleos de processamento cada um.

No supercomputador, o *script* inicia elegendo um nó como *Master* e em seguida, o mesmo elege os demais nós alocados como *Workers*. Durante essa configuração, são criados novos diretórios temporários no diretório *home* do usuário a fim de possibilitar o compartilhamento de arquivos necessários à execução do Spark entre todos os nós previamente alocados. Após esta etapa de configuração, os arquivos que serão consumidos pelo *job* são distribuídos entre os *Workers* e a execução se inicia. Ao finalizar o processamento, o resultado dos *Workers* é agrupado para gerar a solução final. O *job* termina copiando o conteúdo dos diretórios temporários para um outro diretório, podendo ser utilizado posteriormente para eventuais consultas, como arquivos de *Log*, e também parando os nós *Workers* e o *Master* respectivamente.

2.2. Avaliação de Algoritmos de Modelagem de Nichos Ecológicos

O BIOCLIM é um algoritmo clássico de modelo de envelope climático. Seu funcionamento se apoia no cálculo da similaridade de uma localização, comparando os valores das variáveis ambientais em qualquer localização com uma distribuição percentual dos valores em locais conhecidos de ocorrência. Em especial, para o desenvolvimento dessa versão do algoritmo foram utilizadas apenas consultas SQL via SparkR. Foi necessário criar dois *Data Frames*, um referente ao arquivo de variáveis preditivas e outro, referente ao arquivo de ocorrências. Como o arquivo original de variáveis preditivas se encontra em formato *Raster Stack*, ou seja, um arquivo com várias matrizes (*layers*), foi necessário converte-lo para um formato tabular para realização de consultas. Com isso, foi realizada a concatenação vertical de todas as matrizes, resultando em uma única matriz. Uma das etapas do BIOCLIM é mapear esses pontos geográficos (latitude, longitude) presentes no arquivo de ocorrências nos valores das variáveis ambientais (linha, coluna) das matrizes. A primeira consulta tem esse objetivo e retorna um *Data Frame* contendo uma coluna com o nome das espécies e outras duas com os valores de linha e coluna, referente aos índices das matrizes. A segunda consulta consome esse *Data Frame* gerado e juntamente com os dados de ocorrência gera um novo *Data Frame*, contendo o nome das espécies e o valor das variáveis preditivas em todos os *layers*. Por fim, é gerado o envelope contendo os valores de máximo e mínimo das variáveis ambientais para cada espécie.

O *Suporting Vector Machine* (SVM) é um método de aprendizagem de máquina capaz de analisar e reconhecer padrões em dados e, nesse caso, foi usado para a classificação. Para desenvolver essa versão do algoritmo foi utilizada a versão paralelizada do SVM disponível no Spark, derivada da biblioteca MLlib [Meng et al. 2016]. Inicialmente é criado um *Data Frame* contendo todos os dados de ocorrência das espécies, em seguida, é acrescentada uma coluna referente a presença da espécie no determinado ponto. Essa coluna foi completamente preenchida com *"Yes"*. Depois é gerado um segundo *Data Frame*, de tamanho ajustável, contendo pontos de ocorrência gerados aleatoriamente e com a coluna de presença completamente preenchida com *"No"*. Esse segundo *Data Frame* é essencial à classificação devido ao SVM linear necessitar de da-

dos de ausência também. Por fim, esse *Data Frame* é concatenado verticalmente ao primeiro gerando uma matriz única. Para executar o SVM, uma parte dos dados é separada para treino e o restante para teste. O modelo é criado usando os dados de treino e a classificação é feita utilizando a coluna de presença em relação a cada espécie. A fim de testar a precisão da classificação, é realizada a predição do modelo com os dados de teste. Finalmente é criada a matriz de confusão com base no resultado da predição em relação à coluna presença dos dados de teste, para visualizar o número total de acertos, falsos positivos e falsos negativos. Visando uma análise mais detalhada, foram desenvolvidas 3 consultas SQL para visualizar individualmente o número de acertos, falsos positivos e falsos negativos de cada espécie. Em testes preliminares, utilizando um arquivo com 619 pontos de ocorrências, referentes a 4 espécies distintas, o SVM foi capaz de classificar corretamente 88.97% dos registros de ocorrência do conjunto de teste, gerando 11.02% de falsos positivos e 0% de falsos negativos. Foram utilizados 10% de dados de ausência, 80% para treino e 20% para teste.

3. Conclusão

Com isso, avalia-se a possibilidade de implementação de técnicas de paralelismo às etapas de modelagem do Model-R. Assim estima-se tornar a ferramenta computacionalmente superior em desempenho de forma a escalar melhor com conjunto de dados maiores e também proporcionar a portabilidade para arquiteturas do tipo *Big Data*, não necessitando de sistemas de processamento paralelo com arquiteturas mais sofisticadas. Atualmente, testes de *Benchmark* estão sendo realizados afim de mensurar os possíveis ganhos de desempenho do Spark em um ambiente HPC (SDumont). Como trabalho futuro visa-se a continuidade do estudo de forma a criar versões otimizadas para todos os algoritmos presentes no Model-R.

Referências

Drake, J. M., Randin, C., and Guisan, A. (2006). Modelling ecological niches with support vector machines. *Journal of applied ecology*, 43(3):424–432.

Knaus, J. (2010). Developing parallel programs using snowfall. *Retrieved from CRAN*.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

Sánchez-Tapia, A., de Siqueira, M. F., Lima, R. O., Barros, F. S. M., Gall, G. M., Gadelha, L. M. R., da Silva, L. A. E., and Osthoff, C. (2018). Model-r: A framework for scalable and reproducible ecological niche modeling. In Mocskos, E. and Nesmachnow, S., editors, *High Performance Computing*, pages 218–232, Cham. Springer International Publishing.

Venkataraman, S., Yang, Z., Liu, D., Liang, E., Falaki, H., Meng, X., Xin, R., Ghodsi, A., Franklin, M., Stoica, I., and Zaharia, M. (2016). Sparkr: Scaling r programs with spark. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 1099–1104, New York, NY, USA. ACM.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65.