Desenvolvimento de um *Framework* de Aprendizado de Máquina no Apoio a *Gateways* Científicos Verdes, Inteligentes e Eficientes: BioinfoPortal como Caso de Estudo Brasileiro

Micaella Coelho[1], Guilherme Freire[1], Kary Ocaña[1], Carla Osthoff[1], Marcelo Galheigo[1], André R. Carneiro[1], Francieli Boito[2], Philippe Navaux[3], Douglas O. Cardoso[4]

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

[2]Univ. Bordeaux, CNRS, Bordeaux INP, INRIA, LaBRI. Talence – France.

[3]Informatics Institute, Federal University of Rio Grande do Sul, UFRGS. RS – Brasil.

[4]Instituto Politécnico de Tomar,
Centro de Investigação em Cidades Inteligentes. Tomar – Portugal.

{micaella,gfreire,karyann,osthoff,galheigo,andrerc}@lncc.br

francieli.zanon-boito@u-bordeaux.fr

navaux@inf.ufrgs.br, douglas.cardoso@ipt.pt

Resumo. *Gateways científicos trazem enormes benefícios para usuários finais, simplificando o acesso e ocultando a complexidade da infraestrutura de computação distribuída subjacente. O gateway científico de bioinformática, BioinfoPortal, por meio do seu middleware CSGrid, usufrui dos recursos heterogêneos do Santos Dumont. No entanto, a submissão de tarefas ainda exige um esforço significativo, no que tange à decisão da melhor configuração que leve a uma execução eficiente. O framework de aprendizado de máquina, em desenvolvimento, ao ser integrado ao gateway, viabilizará essa decisão. No presente trabalho apresentamos um estudo de desempenho com caso de estudo da bioinformática visando analisar o comportamento de variáveis de saída do slurm/sacct dado valores das variáveis de entrada obtidas da configuração de tarefas do SDumont, o que pôde ser realizado pela modelagem deste cenário como uma tarefa de classificação binária. Os nossos resultados indicam ser possível extrair regras e avaliar a influência das variáveis de entrada Bootstrap, Nó e Thread, sendo Bootstrap a mais significativa e aquela com mais peso para o sistema de recomendação de alocação de recursos no BioinfoPortal.*

1. Introdução

Gateways científicos são soluções de *software* que visam a integração de dados e processos reutilizáveis a técnicas especializadas por meio de servidores *Web*, enquanto ocultam a complexidade das necessidades de recursos de computação de alto desempenho (CAD) subjacentes. *Gateways* científicos permitem que comunidades de pesquisa acadêmica acessem dados compartilhados, *software*, serviços de computação, tecnologia, materiais educacionais e outros recursos específicos em diversas áreas. Eles são geralmente considerados como portais *Web* ou um conjunto de aplicativos de *desktop* [Gesing et al. 2018].

Instituições internacionais vêm ao apoio de projetos para *gateways* científicos e tecnologias de estrutura de *gateways* que utilizam uma abundante infraestrutura cibernética avançada, visando melhorar a interoperabilidade entre essas tecnologias. Dentre elas, o *Science Gateways Community Institute* (SGCI[1]), apoiado pela *National Science Foundation* (NSF), promove a conferência anual *Gateway*[2] como uma oportunidade para que criadores e entusiastas aprendam, compartilhem e moldem o futuro dos *gateways*, como uma comunidade vibrante com interesses em comum.

Nas ciências biológicas e de bioinformática, devido à ampla gama de aplicações importantes em saúde, clínica, diversidade e ciências da vida, o compartilhamento de informações científicas está atrelado ao entendimento da cultura de CAD para acelerar a transição de simulações computacionais de sistemas biológicos em todas as escalas. BioinfoPortal[3] é um *gateway* científico de bioinformática que segue o modelo de entrega *Software as a Service* e facilita as execuções paralelas e distribuídas de programas de bioinformática e *workflows* científicos.

O BioinfoPortal foi desenvolvido sob a arquitetura do *middleware* CSGrid, está hospedado no Sistema Nacional de Computação de Alto Desempenho[4] (SINAPAD) e utiliza os recursos computacionais do supercomputador Santos Dumont[5] (SDumont) do Laboratório Nacional de Computação Científica (LNCC). O portal suporta execuções de *software*, *workflows* e bibliotecas de bioinformática que foram instaladas e alocadas no ambiente computacional do SDumont. Por esse motivo, o desempenho que o *gateway* oferece a seus usuários está diretamente ligado ao desempenho da execução das tarefas no supercomputador, incluindo a escolha de parâmetros como o número de nós e *threads* a serem utilizados [Ocaña et al. 2020].

Portanto, a motivação deste artigo é de melhorar o desempenho do BioInfoPortal, e promover um melhor uso do SDumont, através da seleção do melhor conjunto de parâmetros a serem usados para submissão do *job* ao gerenciador de recursos da máquina. O maior desafio nessa tarefa vem do fato de que cada aplicação oferecida pelo portal requer um conjunto diferente de parâmetros para obter o seu melhor desempenho. Há diferentes maneiras de lidar com essa situação: um exemplo seria a testagem de todos os possíveis parâmetros para todas as possíveis aplicações. A melhor configuração para cada caso seria guardada e consultada pelo sistema no momento de lançar execuções. Essa alternativa traz um enorme custo em tempo e recursos computacionais. Para reduzir esse custo, apenas parte dos parâmetros para cada aplicação pode ser testada, com os resultados obtidos sendo extrapolados para outras situações. Isso pode ser feito através de modelos analíticos, mas essa tarefa é complexa e requer um estudo detalhado do desempenho de cada aplicação utilizada no *gateway*. Finalmente, outra forma de obter as informações necessárias através de um conjunto não exaustivo de testes é o uso de aprendizado de máquina. Uma mesma metodologia, uma vez decidida, pode ser facilmente aplicada a cada nova aplicação do portal. Esse é o método abordado em nosso trabalho, dado que se encaixa melhor em nosso cenário.

---

[1] `https://sciencegateways.org/`
[2] `https://sciencegateways.org/gateways2022`
[3] `https://bioinfo.lncc.br`
[4] `https://www.lncc.br/sinapad/`
[5] `https://sdumont.lncc.br/`

Neste trabalho, tecnologias de aprendizado de máquina foram exploradas por meio da análise de modelos, que foram alimentados por informações de desempenho provindos das submissões do *gateway* para o SDumont. Os resultados foram utilizados para predizer o comportamento de variáveis baseado no histórico de execuções. Este seria um primeiro passo para o desenvolvimento de um *framework* que visa, através de modelos preditivos, auxiliar na escolha da melhor configuração de cada tarefa do *gateway* para maximizar a vazão e eficiência do SDumont como um todo. Este *framework*, quando concluído, será acoplado à arquitetura do BioinfoPortal de forma totalmente transparente ao usuário.

Para esse primeiro passo, utilizamos como estudo de caso experimentos de filogenia usando o *software* RAxML. O objetivo é encontrar o conjunto de parâmetros que levam à máxima eficiência na execução dessa aplicação. Para isso, nós nos concentramos na especificação dos recursos de dados biológicos (ou seja, tamanho em kB), parâmetros da aplicação de bioinformática (*bootstrap* no caso do RAxML), infraestrutura computacional (ou seja, número de núcleos) e eficiência da execução do desempenho.

O restante do artigo está estruturado como detalhado a seguir. A Seção 2 apresenta os trabalhos relacionados ao desenvolvimento de *gateways* científicos acoplados a tecnologias de aprendizado de máquinas. A Seção 3 mostra a fundamentação da bioinformática. Na Seção 4 são apresentas a metodologia, a configuração do experimento, o ambiente computacional utilizado e estudo de caso. A Seção 5 apresenta e discute os resultados computacionais e a Seção 6 apresenta as considerações finais do presente artigo.

2. Trabalhos relacionados

A modelagem, desenvolvimento e uso de *gateways* científicos ganharam interesse e atenção nos últimos anos. Vários projetos e iniciativas se fortalecem em todo o mundo visando o desenvolvimento de estruturas para *gateways* científicos em diversas comunidades de usuários. Esta seção apresenta alguns dos *gateways* científicos mais recentes para Ciências da Vida, os *frameworks* mais comumente usados, alguns *gateways* científicos desenvolvidos pela comunidade de bioinformática e as abordagens de aprendizado de máquina promissoras para o desenvolvimento de *gateways* científicos eficientes. Existem vários programas e iniciativas globais com foco no desenvolvimento e sustentabilidade dos *gateways* científicos, incluindo o SGCI, programas de financiamento da Comissão Europeia para pesquisa e inovação para comunidades de usuários europeus[6] para criar *gateways* científicos e a Colaboração Nacional de Pesquisa Eletrônica da Austrália[7] (Nectar) para facilitar programas de infraestrutura de software para o desenvolvimento de laboratórios virtuais. Além disso, vários *middleware* proliferaram para suportar *gateways* científicos, incluindo Apache Airavata, CitSci.org, CyVerse, projeto Galaxy, projeto Globus, gUSE (*Grid and Cloud User Support Environment)*), HUBzero, Zooniverse, e MyExperiment. Dentre as implementações de *middleware* bem-sucedidas adaptadas para vários *gateways* científicos de bioinformática se encontra CIPRES[8].

Várias comunidades de Ciências da Vida escolheram a tecnologia de grade para realizar a análise de dados em pesquisas médicas colaborativas, cobrindo suas necessidades computacionais e intensivas em dados. Dentre elas, o portal WSPGRADE que é um

[6] `https://ec.europa.eu/fpi/home_en`
[7] `https://nectar.org.au/nectar-impact/`
[8] `https://www.phylo.org/`

repositório de aplicativos e navegador de arquivos de grade; o portal Grid Enabled web eNvironment for site *Independent User Job Submission* (GENIUS) para *workflows* com Triana; MediGRID implementa aplicações biomédicas usando o *Grid Workflow Description Language (GWorkflowDL)* e o portal para dados de ressonância magnética funcional (fMRI). O *gateway* científico CyberInfrastructure for Phylogenetic RESearch27 (CIPRES) é um recurso público para inferência de grandes árvores filogenéticas. CIPRES foi projetado para fornecer acesso aos recursos computacionais do NSF XSEDE por meio de uma interface de navegador simples.

Em [Alves et al. 2020] é apresentada uma solução baseada no *framework Iterated Local Search* (ILS) visando solucionar o problema de posicionamento de máquina virtual com reconhecimento de interferência (IVMPP) para aplicações de CAD de pequena escala em nuvens. A ideia foi reduzir a interferência que ocorre em aplicações CAD de pequena escala que compartilham máquinas físicas e ao mesmo tempo reduzir o número de máquinas físicas alocadas. Para isso, utilizaram um modelo quantitativo e multivariado para estimar o nível de interferência por aplicações *co-located*, em que considera o acesso SLLC (cache compartilhada de último nível), DRAM (memória de acesso aleatório dinâmico) e rede virtual das aplicações e a semelhança entre os perfis de acesso das aplicações com o objetivo de estimar a interferência. Em suma, a proposta implementada mostrou redução na interferência.

[Gomes et al. 2015] propõem um conjunto de ferramentas para prototipagem de aplicações para *gateways* científicos baseado no *middleware* CSGrid, uma instanciação do *framework* CSBase adotado pelo SINAPAD. *Gateways* científicos de bioinformática como o BioinfoPortal e de modelagem molecular como o Dockthor[9] usufruem do ambiente computacional do SDumont. [Ocaña et al. 2020] Apresenta a arquitetura de quatro camadas (interface, gerência, dados e recursos), funcionalidade e desempenho do BioinfoPortal, usado como motivação no presente artigo. O artigo apresenta diversas análises de desempenho e escalabilidade de *software* e *workflows* de bioinformática alocados no SDumont, indicando a funcionalidade do *gateway*. No presente artigo é proposto um *framework* de aprendizado de máquina como uma quinta camada para viabilizar as alocações de recursos computacionais de maneira dinâmica e eficiente.

[Pierantoni et al. 2022] apresenta uma estrutura de *gateway* científico que suporta a criação, publicação, seleção e implantação de arquiteturas de referência para a nuvem que podem ser instanciadas e executadas automaticamente mesmo por usuários não técnicos. A estrutura incorpora um módulo de aprendizado de repositório de conhecimento que fornece suporte de *e-learning* incorporado. Uma das principais contribuições deste artigo é apresentar uma estrutura genérica de *gateway* científico que suporte a criação, armazenamento, seleção e execução de arquiteturas de referência dentro do PITHIA-NRF.

3. Fundamentação da Bioinformática

3.1. Filogenia

A filogenia visa inferir a história evolutiva por meio das relações filogenéticas entre espécies ou clados em uma árvore. Uma árvore filogenética é composta de vértices ou nós, onde as folhas ou nós externos representam organismos e os nós internos os ancestrais.

[9] https://www.dockthor.lncc.br/v2/

As arestas da árvore indicam as relações evolutivas e o tamanho a distâncias evolutivas entre dois nós. Árvores filogenéticas podem ser construídas em populações, espécies, gêneros ou outros grupos de indivíduos e inclusive para dados morfológicos descritivos e binários. *Bootstrapping* é um procedimento estatístico que *reamostra* um único conjunto de dados para criar muitas amostras simuladas. O valor de *bootstrap* para um clado é a proporção das árvores replicadas que recuperaram aquele clado em particular. Dentre os algoritmos estados-da-arte usados para a construção de árvores filogenéticas estão o de agrupamento de vizinhos (AV), máxima parcimônia (MP), máxima verossimilhança (MV) e inferência bayesiana (IB). Esses algoritmos têm sido desenvolvidos para calcular e reconstruir a árvore onde as relações de homologia entre as sequências sejam melhor refletidas [Lemey et al. 2009].

3.2. *Software* filogenéticos

Desde que a primeira versão do programa PHYLIP (PHYLogeny Inference Package) foi introduzida por Felsenstein, um elevado número de programas filogenéticos vem sendo desenvolvido. Programas como PHYLIP, PAUP, PAML e MrBayes estão entre os mais bem conhecidos e encontram-se disponíveis para milhares de usuários ao redor do mundo. As novas versões paralelizadas e de velocidade otimizada de alguns programas como MrBayes e RAxML tornaram-se a melhor opção para a análise de grandes quantidades de dados com configuração de paralelismo e distribuição de tarefas. Novas alterações de otimização desses programas tornam as análises filogenéticas mais promissoras e eficientes.

3.3. RAxML

Análises filogenéticas apoiam estudos sobre a vida evolutiva dos organismos. *Software* otimizados como o RAxML (*Randomized Axelerated Maximum Likelihood*), baseadas em algoritmos de MV, geram alto custo computacional pelos cálculos de probabilidades filogenéticas e o processamento da elevada quantidade de dados genômicos. O uso eficiente desses *software* em ambiente CAD torna-se um requisito para usufruir da melhor configuração no ambiente do SDumont. Algoritmos de MV implementam modelos probabilísticos complexos e eficazes, que geram alto custo computacional de CPU memória. Dentre os *software* de MV mais usados estão RAxML/ExaML, PhyML e IQ-TREE.

RAxML é um programa popular para análise filogenética para a análise de grandes conjuntos de dados sob MV. RAxML se caracteriza por apresentar um algoritmo rápido e consistente de busca de árvore de MV que retorna árvores com boas pontuações de verossimilhança. RAxML tem sido continuamente mantido e estendido para acomodar conjuntos de dados de entrada cada vez maiores. Alguns dos novos recursos e extensões mais notáveis do RAxML é a extensão substancial de modelos de substituição e tipos de dados suportados; a introdução de vetoriais intrínsecos SSE3, AVX e AVX2; técnicas para reduzir os requisitos de memória do código e uma infinidade de operações para a realização de pós-análises em conjuntos de árvores [Stamatakis 2014].

RAxML oferece uma paralelização de granularidade fina da função de verossimilhança para sistemas *multicore* por meio da versão baseada em PThreads e uma paralelização de granularidade grossa de buscas em árvores independentes via MPI (*Message Passing Interface*). Ele também suporta paralelismo de grão grosso e fino por meio da versão híbrida MPI/PThreads, onde usa o MPI para distribuir entre nós as réplicas de *bootstrap* e o PThreads para em cada nó paralelizar os cálculos de probabilidade de busca de árvore.

Observe que, para análises extremamente grandes em supercomputadores, recomenda-se usar o programa irmão dedicado ExaML (*Exascale Maximum Likelihood*). RAxML implementa uma opção para reduzir o consumo de tempo de execução de memória nos cálculos de probabilidade filogenética e potencialmente os tempos de execução em grandes conjuntos de dados filogenômicos com dados ausentes. A economia de memória é proporcional à quantidade de dados ausentes no alinhamento [Izquierdo-Carrasco et al. 2012].

4. Método

Para a coleta de dados, foi utilizado o supercomputador SDumont que está entre as 500 máquinas mais poderosas do mundo[10]. Ele possui uma capacidade de processamento de 5.1 Petaflop/s, com 36.472 núcleos de CPU, distribuídos em 1.134 nós computacionais interligados por uma rede de interconexão *Infiniband* FDR ou EDR. As execuções foram realizadas no conjunto de nós computacionais denominado "base" e o *software* utilizado foi o RAxML 8.2.12 na versão Híbrida (MPI/PThread) que foi instalado no ambiente do SDumont. Na Figura 1 é apresentada a metodologia do experimento para a geração de dados filogenéticos com o RAxML híbrido realizado no SDumont.

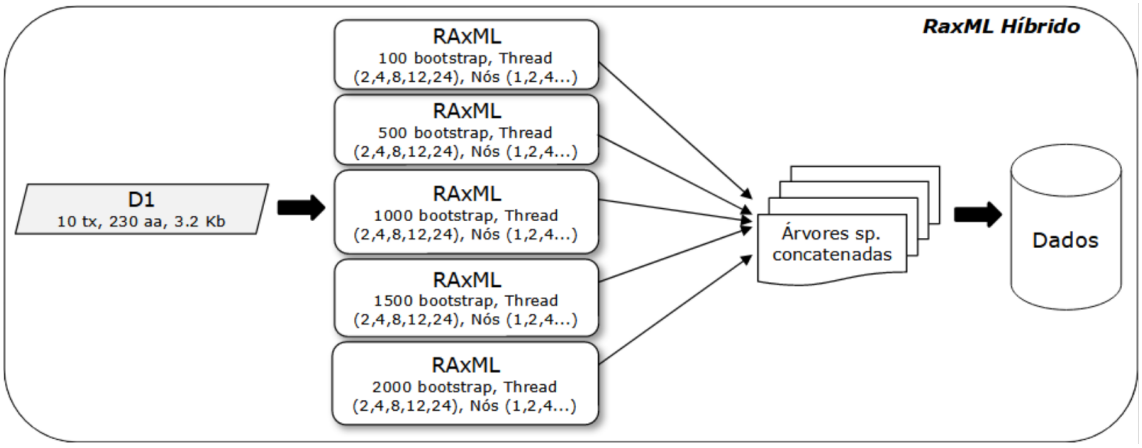

Figura 1. Metodologia para a Geração de Dados Filogenéticos com RAxML

Foi usado um arquivo de entrada que possui formato PHYLIP de superalinhamentos de genomas de protozoários [Ocaña and Dávila 2011] com 10 táxons, 230 caracteres e tamanho 3.2 KB. As variáveis de entrada variaram conforme o parâmetro de entrada da aplicação e as configurações disponíveis no SDumont. Para a aplicação, variou-se o parâmetro *bootstrap*, sendo executado com 100, 500, 1.000, 1.500 e 2.000, também foram variados os números de nós computacionais 1, 2 e 4 e o número de *threads* em 2, 4, 8, 12 e 24. Foram registradas 5 execuções de cada uma das configurações de variáveis, para assim contemplar a ocorrência natural de flutuações no comportamento de execuções de processos em ambientes multitarefas. Cabe ressaltar que este mesmo protocolo experimental pode ser reproduzido em outras plataformas de CAD, considerando outras variações dos parâmetros de configuração das execuções, ou até mesmo superando uma limitação deste estudo no qual foi considerada apenas a aplicação RAxML.

A coleta dos dados computacionais referente a cada execução, e suas respectivas variações de parâmetros mencionadas anteriormente, foi realizada através do comando

[10] `https://www.top500.org/`

sacct presente no slurm, que é o gerenciador e escalonador de tarefas do SDumont. Os dados coletados oferecem uma visão do comportamento da aplicação, no que diz respeito, ao consumo de memória e tempo de execução em CPU. O comando sacct exibe informações sobre o trabalho (*job*) que foi executado, como as etapas do trabalho e o *status*. Por meio do sacct foram coletados dados referentes à contabilidade dos tempos gastos de CPU e do tamanho de memória alocada pelo *software*. No total foram coletadas seis variáveis de saída, as quais são descritas na Tabela 1 e de uma perspectiva estatística na Tabela 2.

Tabela 1. Descrição textual das variáveis de saída consideradas

Nome	Descrição
MaxVMSize	Tamanho máximo de memória virtual alocado.
AveVMSize	Tamanho médio da memória virtual gasta pelo trabalho caso tenha usado a paginação ou *swap*.
MaxRSS	Tamanho máximo de memória RAM alocado.
AveCPU	Tempo médio de CPU, no qual é multiplicado o tempo médio pelo número de núcleos disponíveis.
CPUTime	Tempo total de CPU, no qual é multiplicado o tempo total de um núcleo pelo número total de núcleos disponíveis.
Elapsed	Tempo do trabalho completo.

Tabela 2. Descrição estatística das variáveis de saída na amostra coletada

	MaxVMSize	AveVMSize	MaxRSS	AveCPU	CPUTime	Elapsed
mean	1623330.4	1612360.8	109966.3	2792.0	16655.7	412.8
std	1179758.4	1189902.1	34891.8	3177.8	13910.1	456.8
min	303700.0	39818.0	2296.0	0.0	888.0	12.0
25%	534546.0	529538.5	119410.0	606.5	5376.0	103.0
50%	1076950.0	1071513.5	120868.0	1910.5	13632.0	249.0
75%	2667900.0	2667900.0	122293.0	4045.0	24336.0	557.8
max	4023816.0	4023816.0	126732.0	17252.0	59712.0	2377.0

A coleta deste conjunto de dados teve como alvos avaliar quais variáveis de saída se mostram controláveis de forma mais assertiva pelos parâmetros considerados, e quais destes parâmetros são mais determinantes para que tal controle aconteça. Em outras palavras, foram consideradas duas perguntas que nortearam o presente trabalho, a saber:

1. Quais aspectos das execuções no SDumont são mais previsíveis?
2. Que parâmetros das execuções tem mais impacto no comportamento destas?

A modelagem destas perguntas como problemas de aprendizado de máquina se mostra muito propícia, dada a naturalidade com que pode ser concretizada. Afinal, quanto à primeira pergunta, é possível definir, para cada variável de saída, uma tarefa de aprendizado supervisionado cujas variáveis de entrada são os parâmetros das execuções. Dessa maneira, seria possível avaliar o grau de eficácia em que cada uma dessas tarefas poderia ser solucionada, o que pode ser interpretado como uma medida da previsibilidade das variáveis de saída correspondentes.

Continuando o desenvolvimento do raciocínio recém-proposto, é necessário a definição das tarefas de aprendizado supervisionado a serem consideradas, havendo diversas possibilidades isto, cada um com as suas particularidades. O tipo de tarefa escolhido para compor o método aqui descrito é o de classificação binária, tendo o valor de uma variável de saída num dos registros das execuções convertido num rótulo de classe conforme tal valor fosse inferior ou superior à mediana amostral de tal variável: de forma intuitiva, isto seria semelhante estabelecer o tempo de execução ou uso de memória como "alto" ou "baixo". Desta forma tanto foi empregada a mais simples e clássica tarefa de Aprendizado de Máquina, quanto foi assegurada uma propriedade ideal desta: o uso da mediana como limiar para definição *a priori* das classes visou evitar complicações decorrentes de desbalanceamento no tocante a variável-alvo, o qual é possível tanto em problemas de classificação [Johnson and Khoshgoftaar 2019] quanto regressão [Ribeiro and Moniz 2020]: ainda sobre este último, mesmo que os dados estivessem dispersos numa ampla faixa de valores, o que poderia ser cogitado com base na Tabela 2, eles ainda poderiam estar desbalanceados, por exemplo, por conta de estarem concentrados em regiões desta faixa.

Cabe ainda discutir a definição do modelo de classificação utilizado. É interessante perceber que não se teve como alvo primário no trabalho aqui detalhado a obtenção da eficácia máxima em cada uma das tarefas de classificação que acabam de ser definidas. Tal objetivo, o qual geralmente tem a mais alta prioridade em estudos sobre tarefas deste tipo, acaba por induzir a avaliação de diversos modelos classificadores, em inúmeras configurações, para que a melhor alternativa seja então identificada. Ao invés disto, este trabalho se propôs a comparar diferentes tarefas de classificação, estimando o grau de dificuldade inerente a cada uma destas, pelo que resulta ser mais razoável que instâncias do mesmo modelo classificador fossem utilizadas para cada uma das tarefas em questão.

Por isso, foi feita a opção pelo uso do modelo nomeado *Extra Trees* [Geurts et al. 2006], usando a configuração padrão de sua implementação na biblioteca Scikit-Learn [Pedregosa et al. 2011]. Tal escolha também considerou a reduzida necessidade de um ajuste fino das configurações deste modelo comparado a outros modelos de propósito similar. Uma última característica deste modelo é que este estima de forma integrada ao seu próprio treinamento o grau de influência de cada variável de entrada na predição da variável de saída ao se realizar classificações em geral. Tal estimativa pode ser usada para responder à segunda questão de pesquisa considerada.

5. Resultados

Nesta seção são apresentados e discutidos os resultados dos experimentos realizados tendo em vista uma avaliação inicial da previsibilidade de execuções no supercomputador Santos Dumont. Tal avaliação inicial é materializada nas duas questões de pesquisas previamente estabelecidas na Seção 4. Cada uma delas é abordada a seguir.

Para cada tarefa de classificação, as quais correspondem a cada uma das variáveis de saída consideradas, foram realizadas 100 validações cruzadas de duas partes (*2-fold cross-validation*). Sendo assim, foi gerado um total 200 registros da acurácia do modelo *Extra Trees* ao ser treinado com um subconjunto aleatoriamente definido a partir do conjunto de dados de execuções que foi coletado, e testado com o complemento deste subconjunto. Além da acurácia, também foi incluído em cada um destes 200 registros a importância de cada variável de entrada estimada pelo modelo classificador durante o

treinamento. Os resultados apresentados a seguir são baseados nesta coleção de registros.

5.1. Quais aspectos das execuções no SDumont são mais previsíveis?

Esta primeira questão é relacionada à facilidade de antever atributos do comportamento das execuções do SDumont, sendo este multifacetado, e assim descrito pelas diversas variáveis de saída consideradas. A Figura 2 ilustra por meio de diagramas de caixa (*box-plots*) os 200 valores de acurácia obtidos para cada uma das variáveis de saída: para além dos elementos habituais, os triângulos verdes representam as médias destes valores.

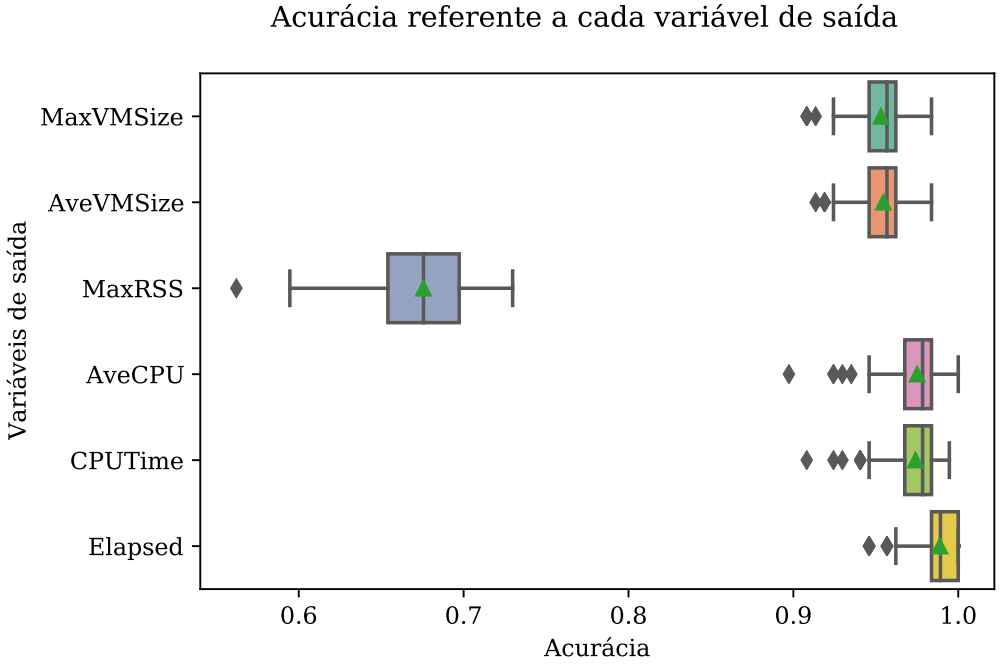

Figura 2. *Boxplot* da Acurácia de variáveis de Saída para Classificação.

É evidente como a variável MaxRSS se destaca das demais, sendo indiscutivelmente a menos previsível. Diametralmente oposta a esta, aquela cuja classificação ocorreu de forma mais eficaz, em geral, foi a variável *Elapsed*, permitindo uma acurácia superior a 95% em quase todas as rodadas do experimento. É também interessante notar como ficaram tecnicamente empatadas as variáveis AveCPU e CPUTime na segunda posição do *ranking* de previsibilidade, e as variáveis MaxVMSize e AveVMSize na terceira posição deste *ranking*: as 3 primeiras variáveis no ranking tem em comum o fato de serem medidas de tempo, o que sinalizaria então de forma ampla que este tipo de aspecto das execuções é o mais controlável; já às duas seguintes são medidas de memória virtual, sendo este aspecto ainda bastante controlável pelas variáveis de entrada consideradas, ainda que um pouco menos que o tempo gasto nas execuções.

5.2. Que parâmetros das execuções tem mais impacto no comportamento destas?

O modelo de classificação utilizado, *Extra Trees*, é definido por um comitê de árvores de decisão, sendo cada uma destas um classificador independente, cujas saídas são agregadas de forma que cada classificação seja feita de forma coletiva e mais eficaz que qualquer uma feita individualmente. De cada uma dessas árvores é possível extrair todas as regras

que determinam cada classificação que é feita, sendo tais regras definidas por critérios que as variáveis de entrada podem ou não atender. Logo, conforme a participação de cada variável de entrada nestas regras, é possível avaliar o grau de influência destas na classificação. A Figura 3 apresenta os valores do grau de influência atribuído a cada uma das 3 variáveis em todas as 1200 rodadas de experimento realizadas (100 repetições da validação cruzada de 2 partes, para cada uma das 6 variáveis de saída).

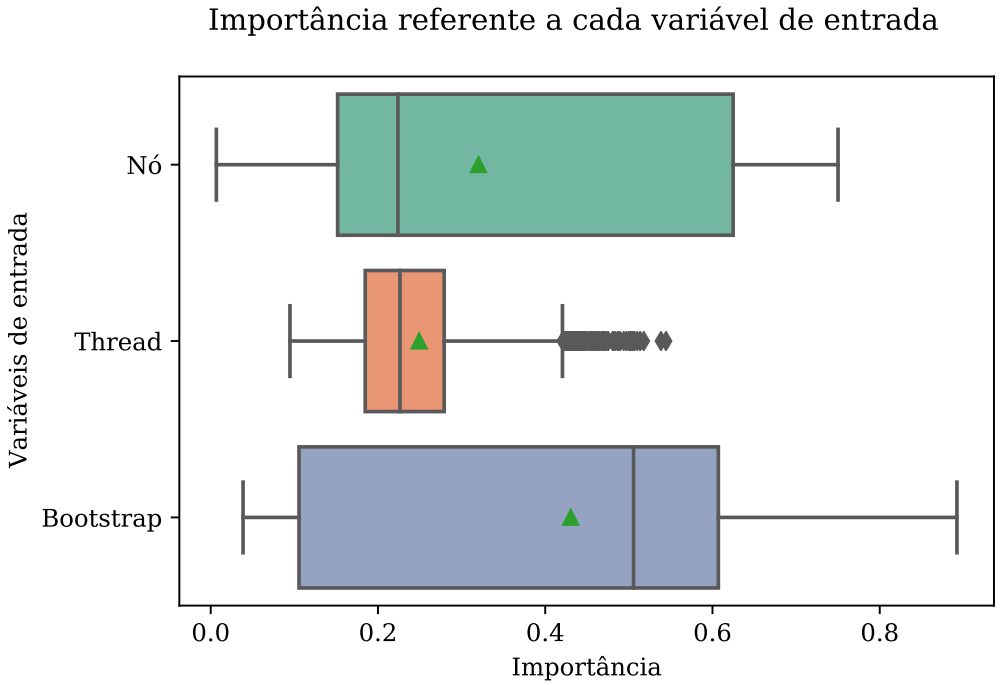

Figura 3. *Boxplot* de Importância de Variáveis de Entrada para Classificação.

Com base na média do grau de importância de cada variável, esta seria a ordem das mesmas, da mais importante para a menos importante: *Bootstrap*, Nó e *Thread*. Para além dessa perspectiva primária, há outras informações interessantes contidas na ilustração em questão: a menor dispersão, representada pela amplitude interquartil, da variável *Thread* comparada às demais aponta no sentindo de uma menor incerteza quanto à expectativa de sua (baixa) importância; as variáveis Nó e *Bootstrap* tem uma amplitude interquartil substancial e parecida, o que permite afirmar que eventualmente cada uma é muito importante em alguns casos e pouco importante em outros, e também questionar se estas alternam entre si como a mais importante de todas de acordo com a variável de saída em questão.

6. Conclusão

O presente estudo visa o melhor entendimento das funções que exercem cada uma das variáveis em um experimento científico, baseado em análises de desempenho e acurácia. Vários sistemas *Web* e de *software* gerenciados pelo SINAPAD podem se beneficiar dessa informação para obter uma resposta rápida no processamento de execuções e na toma de decisão em execuções eficientes do BioinfoPortal no ambiente do SDumont. O *Framework* de aprendizado de máquina proposto é baseado em modelos preditivos que auxiliam na melhor escolha de parâmetros atrelados a uma prospecção de diminuição do

tempo computacional (Figura 4). Ele poderia ser utilizado em vários níveis ou camadas de aplicações distribuídas, tal e como, os meta-escalonadores de grades computacionais, provedores de recursos para *workflows* científicos ou até mesmo sendo incorporado como uma política de alocação no próprio escalonador do *cluster* no SDumont.

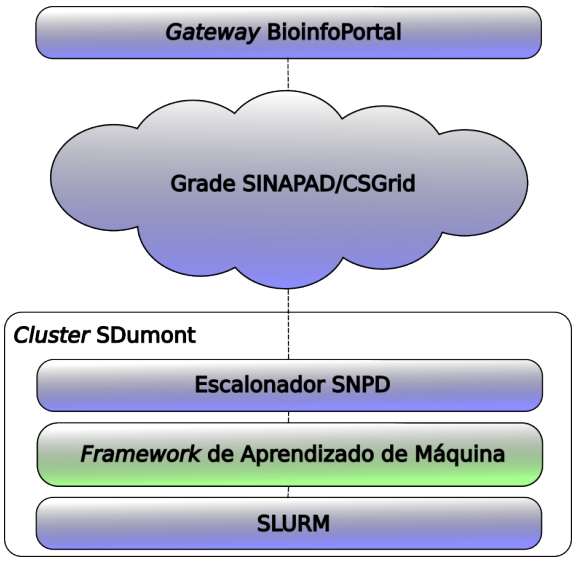

Figura 4. Arquitetura do *Framework* de Aprendizado de Máquina no SDumont

Neste estudo, BioinfoPortal pode se beneficiar das variáveis obtidas dos modelos preditivos e acertados no *framework* de forma a alocar recursos de maneira mais eficiente nos *cluster*. Essa alocação será feita através de um sistema de recomendação de recurso baseada na abordagem de aprendizado de máquina, como demonstrado neste artigo. Este sistema de recomendação pode ser facilmente acoplado ao módulo de escalonamento de tarefas do SINAPAD/CSGrid, o qual permitirá a extensibilidade das formas de alocação de recurso para o BioinfoPortal e outras aplicações hospedadas no SINAPAD.

A abordagem proposta baseada em aprendizado de máquina visa otimizar o desempenho operacional com base no ajuste de parâmetros de rastreamento para otimizar a alocação de recursos. Registros históricos armazenados em bancos de dados podem ser consultados para refinar recursos para alocação que levem à obtenção de melhores valores de eficiência. O caso de estudo de filogenia define a variável de saída MaxRSS como aquela menos previsível com aproximadamente 75% de acurácia obtida pelo modelo *Extra Trees*: a avaliação de outros modelos é pretendida como trabalho futuro e também para realização de novos experimentos com variações de tamanho e configuração de arquivos de entrada para o RAxML e utilização de diferentes parâmetros da aplicação. Tendo em conta que o RAxML apresenta otimizações no consumo dos cálculos de probabilidade filogenética, suportando o baixo consumo de RAM, este resultado abre uma questão a ser resolvida com o uso de casos de estudo com alto consumo de E/S como os experimentos de transcriptômica. Dessa forma, nosso BioinfoPortal poderia se beneficiar de informações obtidas por métodos de aprendizado de máquina. A implementação do *framework* pode tornar BioinfoPortal em um *gateway* de comportamento verde inteligente, pois seu consumo de energia para execuções computacionais seriam inferidos e reduzidos.

Referências

Alves, M., Teylo, L., Frota, Y., and Drummond, L. (2020). *An Interference-Aware Strategy for Co-locating High Performance Computing Applications in Clouds*, pages 3–20.

Gesing, S., Dooley, R., Pierce, M., Krüger, J., Grunzke, R., Herres-Pawlis, S., and Hoffmann, A. (2018). Gathering requirements for advancing simulations in hpc infrastructures via science gateways. *Future Generation Computer Systems*, 82:544–554.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Gomes, A. T. A., Bastos, B. F., Medeiros, V., and Moreira, V. M. (2015). Experiences of the brazilian national high-performance computing network on the rapid prototyping of science gateways. *Concurrency and Computation: Practice and Experience*, 27(2):271–289.

Izquierdo-Carrasco, F., Gagneur, J., and Stamatakis, A. (2012). Trading running time for memory in phylogenetic likelihood computations. In *Bioinformatics*, pages 86–95.

Johnson, J. M. and Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27.

Lemey, P., Salemi, M., and Vandamme, A.-M. (2009). *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press.

Ocaña, K. A. and Dávila, A. M. (2011). Phylogenomics-based reconstruction of protozoan species tree. *Evolutionary Bioinformatics*, 7:EBO–S6861.

Ocaña, K. A., Galheigo, M., Osthoff, C., Gadelha Jr, L. M., Porto, F., Gomes, A. T. A., de Oliveira, D., and Vasconcelos, A. T. (2020). Bioinfoportal: a scientific gateway for integrating bioinformatics applications on the brazilian national high-performance computing network. *Future Generation Computer Systems*, 107:192–214.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Pierantoni, G., Kiss, T., Bolotov, A., Kagialis, D., DesLauriers, J., Ullah, A., Chen, H., Fee, D. C. Y., Dang, H.-V., Kovacs, J., et al. (2022). Toward a reference architecture based science gateway framework with embedded e-learning support. *Concurrency and Computation: Practice and Experience*, page e6872.

Ribeiro, R. P. and Moniz, N. (2020). Imbalanced regression and extreme value prediction. *Machine Learning*, 109(9):1803–1835.

Stamatakis, A. (2014). Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313.