Framework para a Construção de Redes Filogenéticas em Ambiente de Computação de Alto Desempenho

Rafael Terra[1], Kary Ocaña[1], Carla Osthoff[1], Lucas Cruz[1,2], Philippe Navaux[3], Diego Carvalho[2]

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

[2]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET)
Rio de Janeiro – RJ – Brasil.

[3]Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Rio Grande do Sul – RS – Brasil.

rafaelst@posgrad.lncc.br, {karyann,osthoff,lucruz}@lncc.br,

navaux@inf.ufrgs.br, d.carvalho@ieee.org

Resumo. *No presente artigo é apresentado uma avaliação de desempenho de um Framework de Redes Filogenéticas no ambiente do supercomputador Santos Dumont. O trabalho reforça os benefícios de paralelizar o framework usando abordagens paralelas baseadas em Computação de Alta Vazão (CAV), e Computação de Alto Desempenho (CAD). Os resultados da execução paralela do framework proposto, demonstram que este tipo de experimento da bioinformática é apropriado para ser executado em ambientes de CAD; apesar de que nem todas as tarefas e programas componentes do framework tenham sido criados para usufruir de escalabilidade em ambientes de CAD, ou de técnicas de paralelismo em diferentes níveis. A análise comparativa da execução dos cinco pipelines de forma sequencial (como desenhado e usado originalmente por bioinformatas) apresentou um tempo estimado de* $81,67$ *minutos. Já a execução do mesmo experimento por meio do framework executa os cinco pipelines de forma paralela e usufruindo de um melhor gerenciamento das tarefas, gerando um tempo total de execução de 38,73 minutos. Essa melhora é de aproximadamente* $2,11$ *vezes em tempo de execução sugere que a utilização de um framework otimizado leva à diminuição do tempo computacional, à melhora de alocação de recursos e ao tempo de espera na alocação.*

1. Introdução

Nos últimos anos diferentes áreas científicas se beneficiaram da introdução de ferramentas computacionais especializadas na gerência de tarefas, análise de dados e interpretação de resultados biológicos. A área da bioinformática desenvolve e utiliza metodologias capazes de estudar grandes volumes de dados biológicos, em particular sequências de DNA ou proteínas, obtidos a partir de novas técnicas de sequenciamento de nova geração (NGS) [Ott et al. 2008], viabilizando análises filogenômicas.

Conduzir análises filogenômicas é um desafio devido à complexidade da orquestração e interoperação de diversos programas de bioinformática relacionados às

análises evolutivas. *Workflows* científicos [Yu and Buyya 2005] são uma via efetiva usada para modelar essas análises filogenômicas, que apresentam características exploratórias como o tratamento de dados genéticos usando diferentes métodos de bioinformática para confirmar ou refutar uma hipótese relacionada história evolutiva [Lemey et al. 2009]. Uma ilustração de aplicabilidade é a utilização de ferramentas de filogenômica e genômica comparativa de organelas e genomas completos de Flavivírus.

Para aumentar a abrangência do experimento científico proporcionada pelo grande volume de dados originados pelas NGS e para diminuir a chance de ocorrência de erros, todo o processo de construção de uma rede filogenética pode ser modelado em um *workflow* científico responsável por automatizar todas as tarefas. *Workflows* científicos são usados para gerenciar e representar um conjunto de tarefas computacionais, desde a a modelagem do experimento, a execução de programas, a análise dos resultados e a viabilidade da reprodutividade dos experimentos [Pouchard et al. 2019]. Sistemas de gerência de *workflow* científicos são tecnologias responsáveis pela execução do *workflow*, suas atividades e do *dataflow* resultante da dependência entre as tarefas [Yu and Buyya 2005].

Este processo exploratório de análises genômico-evolutivas com grande volume de dados proporcionados pelas NGS demanda ambientes de Computação de Alto Desempenho (CAD) e técnicas paralelas e de distribuição de tarefas para produzir resultados em um tempo aceitável. Apesar de alguns dos ambientes de computação de alto desempenho fornecerem uma nova dimensão aos experimentos de bioinformática devido a características de paralelismo e distribuição de tarefas, o desafio atual está em acoplar eficientemente os *software* usados na construção de redes filogenéticas.

Neste contexto, foi desenvolvido um *framework* composto por um *workflow* científico que agrega um conjunto de *pipelines* que permitem a construção de árvores e redes filogenéticas, cada um usando diferentes algoritmos filogenéticos e apresentando um comportamento particular [Terra 2022]. O objetivo do *framework* é automatizar e otimizar a execução das redes filogenéticas, permitindo executar desde um até cinco *pipelines* em uma única instanciação. Dessa forma, agilizando o experimento científico. Para testar a usabilidade do *framewofk*, em [Terra 2022] foi realizado um caso de estudo inspirado em um tema real de pesquisa na área de saúde, investigando infeções de Dengue no Brasil. A entrada do *framework* é um conjunto de sequências de genes, com base em genomas completos de Flavivírus obtidos por NGS.

Neste artigo é apresentada uma avaliação de desempenho no ambiente do supercomputador Santos Dumont do *framework* descrito em [Terra 2022]. O trabalho reforça os benefícios de paralelizar o *framework* de inferência de redes filogenéticas, usando abordagens paralelas baseadas em Computação de Alta Vazão (CAV, do inglês *High-Throughput Computing*) e Computação de Alto Desempenho (CAD, do inglês *High-Performance Computing*). Os resultados da execução paralela do *framework* proposto demonstram que este tipo de experimento da bioinformática é apropriado para ser executado em ambientes de CAD; apesar de que nem todas as tarefas e programas componentes do *framework* tenham sido criados para usufruir de escalabilidade em ambientes de CAD, ou de técnicas de paralelismo em diferentes níveis.

O artigo está estruturado de forma a primeiro contextualizar os conceitos biológicos e computacionais e *workflows* científicos na Seção 2, necessários para o enten-

dimento de aplicação em redes filogenéticas. A Seção 3 apresenta trabalhos relacionados ao desenvolvimento desta pesquisa multidisciplinar. A Seção 4 apresenta a metodologia, descreve a arquitetura conceitual do *framework* proposto, a configuração do experimento, o ambiente computacional utilizado e estudo de caso. A Seção 5 apresenta e discute os resultados computacionais e concluindo, a Seção 6 apresenta as considerações finais do presente artigo.

2. Fundamentos Biológicos e Computacionais

2.1. Análises Filogenéticas e Filogenômicas

As relações evolutivas entre os indivíduos podem ser representadas por meio da topologia de uma árvore filogenética. Cada nó representa uma Unidade Taxonômica Operacional (UTO) ou comumente chamada de táxon, os quais estão conectados por arestas que representam a distância evolutiva entre os táxons. Sequências de grupos externos (*outgroups*) são designados a um grupo de organismos mais distantemente relacionados, que serve como referência para determinar as relações evolutivas do grupo interno em estudo. Após a seleção das sequências é necessário realizar o alinhamento múltiplo de sequências (AMS) que determina a similaridade entre as mesmas e servem para construir árvores filogenéticas a partir dos métodos existentes [Lemey et al. 2009].

Para inferir as relações evolutivas de um determinado grupo de espécies, considerando sequências de DNA ou aminoácidos, é necessária a escolha de um método filogenético a ser aplicado em um conjunto de sequências homólogas, isto é, que apresentem uma ancestralidade em comum. A análise evolutiva por meio de técnicas computacionais requer a escolha de algoritmos matemáticos para a reconstrução filogenética. Métodos filogenéticos são classificados de acordo com os tipos de dados ou pela técnica utilizada na reconstrução de filogenias. Para aplicação em inferências baseadas em dados moleculares, destacam-se quatro técnicas principais: agrupamento de vizinhos, máxima parcimônia, máxima verossimilhança e inferência Bayesiana [Lemey et al. 2009].

2.2. Redes Filogenéticas

Em uma análise evolutiva baseada em árvores, é possível encontrar incongruências topológicas provindas de eventos de evolução reticulada (também chamado de evolução horizontal), como transferência horizontal e recombinação, ou até propor outros cenários biogeográficos de dispersão. Uma árvore filogenética pode não ser consistente ao exibir eventos evolutivos decorrentes de processos de transferência horizontal, recombinações genéticas e hibridização. Nesses casos é recomendado a integração com análises baseadas em rede filogenéticas [Huson et al. 2010].

Uma rede filogenética é uma estrutura de grafo usado para visualizar relações evolutivas entre espécies ou organismos e é usada quando se especula que eventos reticulados tais como hibridação, transferência horizontal de genes, recombinação ou duplicação gênica estão envolvidos. As árvores filogenéticas são um subconjunto das redes filogenéticas, sendo redes que não apresentam nenhum evento reticulado. Redes filogenéticas podem ser inferidas e visualizadas usando *software* como Splitstree, Network ou TCS [Huson et al. 2010].

Redes filogenéticas podem ser implícitas e explícitas. Por representarem a história evolutiva de indivíduos, as redes explícitas são as recomendadas e alvo de

estudo no presente artigo; no entanto são consideradas computacionalmente custosas [Solís-Lemus and Ané 2016]. Ressalta-se que a construção de redes filogenéticas segue um processo constituído pela chamada de diversos *software*, algoritmos e *pipelines*. Em termos da bioinformática, construir uma rede filogenética implica na junção de domínios de genômica comparativa para a construção de AMS e análises de ortologia; e de filogenia e filogenômica para a construção de árvores de genes e de espécies.

2.3. *Workflows* científicos

O avanço da tecnologia possibilitou a execução de experimentos científicos de diferentes domínios das ciências, cada vez mais complexos e de dados intensos [Yu and Buyya 2005]. A execução desses experimentos precisam de uma grande quantidade de poder computacional e de *software* especializados, de modo a interagirem de forma eficiente com ambientes computacionais paralelos e distribuídos [Andronico et al. 2011].

O ciclo de vida de um experimento científico pode ser dividido em três etapas: composição, execução e análise. A composição é a etapa na qual todo o experimento científico é estruturado, apresentando a sequência lógica das atividades, além de seus parâmetros e tipos de dados. A fase de execução define o escopo do experimento de forma concreta, i.e., os seus dados de entrada, parâmetros de cada tarefa, o tipo de *software* e características do meio computacional que são necessários para a execução do experimento. Na etapa de análise, os dados, proveniência do experimento e informações dos resultados do domínio avaliado são analisados tal a aceitar ou refutar uma hipótese pre-definida [Mattoso et al. 2010].

Workflows científicos são utilizados como uma abstração para modelar o fluxo de atividades e de dados em um experimento de maneira estruturada. Segundo [Taylor et al. 2007], os *workflows* se tornaram um padrão para a modelagem de experimentos na comunidade científica. Eles podem ser definidos como a especificação formal de um processo científico que representa os passos a serem executados em um determinado experimento científico [Deelman et al. 2018]. Cada atividade corresponde à invocação de um programa, e as dependências representam o fluxo de dados entre as atividades envolvidas na execução. Portanto, o cientista consegue modelar e executar experimentos científicos por meio de tarefas computacionais complexas em larga escala em ambientes de CAD por meio dos *workflows* científicos [Versluis and Iosup 2021].

3. Trabalhos Relacionados

O campo de estudo em redes filogenéticas é recente, com algoritmos conceitualizados teoricamente, sem estarem necessariamente implementados em *software*, pacotes ou bibliotecas. [Mao et al. 2020] é uma referência estado-da-arte sobre experimentos de redes filogenéticas. Eles propuseram um *workflow* genômico que produz resultados usados para alimentar experimentos científicos para a construção de redes filogenéticas; contudo, ainda não foi modelado para usufruir de ambientes de CAV ou CAD.

A grande quantidade de dados intermediários é um problema recorrente na inferência de redes filogenéticas. [Mao et al. 2020] propuseram uma abordagem similar para minimizar a interação entre o usuário e os dados. Eles modelaram um *workflow* escrito em Python que incorpora alinhamento de sequências, reconstrução de árvores de

genes, árvores de espécies e inferência de redes filogenéticas por meio de diversos programas acoplados ao *workflow*. O *workflow* apresenta múltiplos produtos, uma única execução resulta em duas redes filogenéticas, uma de cada programa, uma árvore de espécies inferida a partir de árvores de genes e uma a partir de sequências concatenadas. Tudo isso é gerado a partir de dados contendo sequências codificadoras de proteínas, sequências de nucleotídeos, sequências de polimorfismo de nucleotídeo único, entre outras.

Em [Stenz et al. 2015], os autores apresentam o *pipeline* TICR, capaz de produzir uma tabela de fatores de concordância de quartetos e também uma árvore de população. Esse *pipeline* estima essa árvore procurando por discordâncias nas árvores de genes devido a recombinação. Para isso ele consiste do método MDL [Ané 2011] e os programas MrBayes, para realizar a inferência Bayesiana utilizando uma variante da Cadeia de MarKov Monte Carlo [Huelsenbeck and Ronquist 2001]; BUCKy, para análise de concordância Bayesiana, recebendo como entrada árvores completas geradas pela análise Bayesiana de locus individuais e gerando como saída uma amostra de árvores de genes e suas distribuições de probabilidade conjuntas [Ané et al. 2007, Larget et al. 2010]; e Quartet Maxcut [Snir and Rao 2012], usado para gerar a árvore de população. A tabela de fatores de concordância de quartetos e a árvore de população podem ser usadas em conjunto para a inferência de redes filogenéticas, sendo passadas como entrada do PhyloNetworks.

4. Metodologia

Existem diversas combinações possíveis usando diversos algoritmos para a construção de redes filogenéticas. Para aumentar a variedade de redes geradas, em [Terra 2022] é realizada a modelagem de um *workflow* composto por diferentes ferramentas com o intuito de construir redes filogenéticas. Foram modelados cinco *pipelines*, acoplados em um único *workflow*. Para realizar a construção das redes filogenéticas dois algoritmos foram escolhidos: Máxima Parcimônia (MP), utilizando o programa PhyloNet [Wen et al. 2018] e Máxima Pseudo-verossimilhança (MPV), utilizando o programa SNaQ (*Species Networks applying Quartets*) do pacote PhyloNetworks [Solís-Lemus et al. 2017]. Esses programas utilizam como de entrada os resultados provindos de análises filogenéticas. O Phylonet utiliza como entrada árvores de genes enraizadas, já o PhyloNetworks pode utilizar tanto árvores não-enraizadas quanto uma tabela de fatores de concordância de quartetos, juntamente com uma topologia inicial como, por exemplo, uma árvore de espécies.

Para construir as árvores de genes foram acoplados os programas RAxML (*Randomized Axelerated Maximum Likelihood*) [Stamatakis 2014], IQ-TREE [Minh et al. 2020] e MrBayes [Huelsenbeck and Ronquist 2001]. RAxML e IQ-TREE baseiam os seus cálculos no algoritmo de MV, porém cada programa apresentando características específicas nos seus cálculos probabilísticos. O MrBayes se baseia no algoritmo de inferência Bayesiana (IB) para a construção das árvores. Devido ao PhyloNetworks utilizar como entrada uma árvore como topologia inicial, foi acoplado ao *workflow* o programa ASTRAL [Mirarab et al. 2014] que gera árvores de espécies a partir de um conjunto de árvores de genes não-enraizadas.

Ao todo, o *workflow* apresenta cinco *pipelines* baseados em diferentes metodologias. Esses *pipelines* foram nomeados de acordo com os programas representativos de

filogenia e redes filogenéticas usados, assim tem-se os seguintes *pipelines*: raxml_snaq (Figura 1a), raxml_phylonet (Figura 1b) e mrbayes_snaq (Figura 1c). Os *pipelines* iqtree_snaq e iqtree_phylonet são similares às Figuras (a) e (b) respectivamente, com a devida substiruição do programa de construção filogenética.

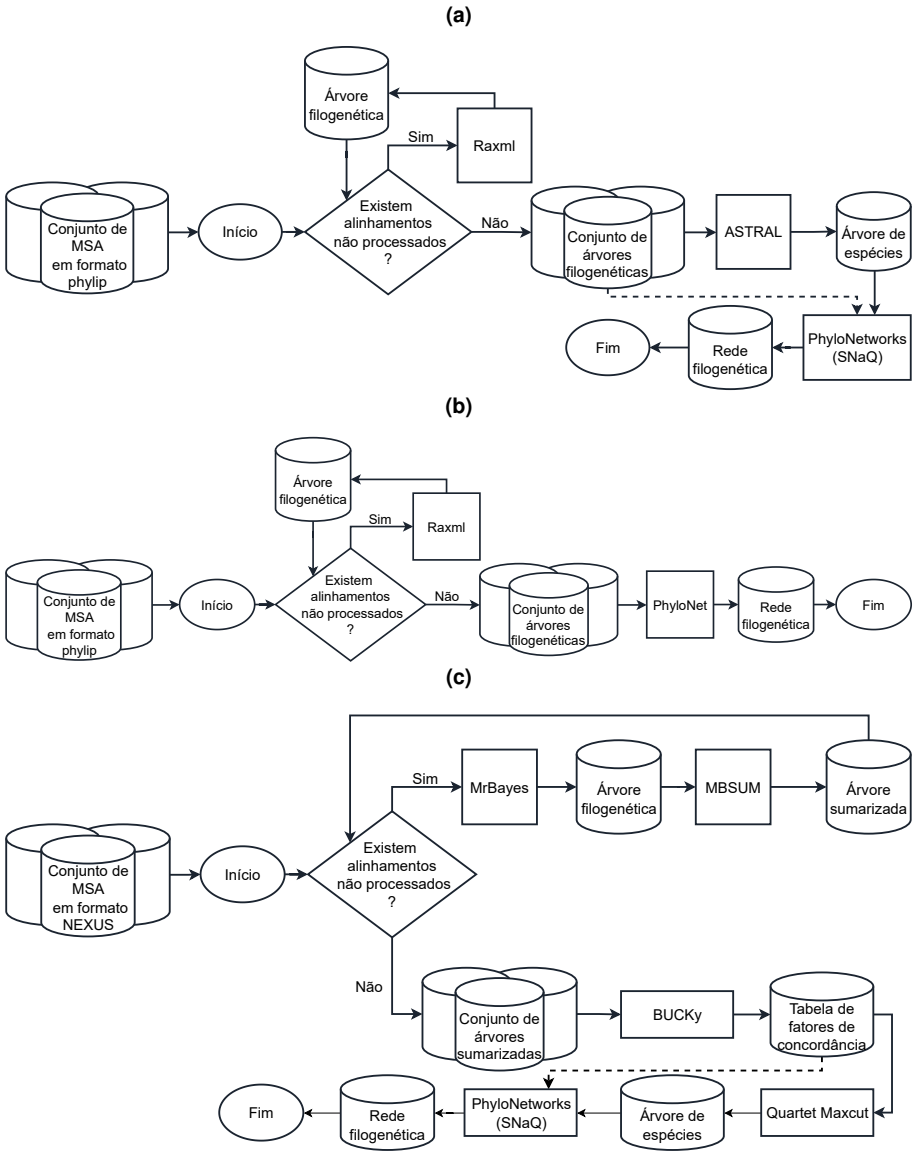

Figura 1. Cinco *pipelines* do *workflow*. (a) raxml_snaq, (b) raxml_phylonet e (c) mrbayes_snaq. iqtree_snaq e iqtree_phylonet possuem composição similar a (a) e (b) respectivamente, substituindo o programa de filogenia.

O *workflow* apresenta um *dataflow*, onde as saídas de cada programa são geralmente utilizadas como entradas pelo seguinte. Entretanto, é necessária a inclusão de atividades intermediárias para realizar diversas manipulações de dados para se compatibilizar a saída de uma tarefa com a entrada da subsequente. Como exemplos, podemos citar a conversão dos dados de entrada dos *pipelines* que utilizam RaXML ou IQ-TREE; a compressão de diversos dados pelos RaXML ou IQ-TREE após suas execuções; e o enraizamento das árvores geradas nos *pipelines* que utilizam o Phylonet.

O grande número de tarefas atreladas à execução do *workflow* abrange desde a execução de *scripts* mais simples usados para manipular dados intermediários até programas e pacotes complexos, com tarefas apresentando diferentes níveis de paralelismo. Por exemplo, as manipulações de dados, geralmente, utilizam apenas um núcleo do processador, enquanto outros programas como RAxML podem utilizar múltiplos núcleos por execução. Esse fator traz a necessidade de um controle da execução dessas tarefas. Para realizar esse controle, foi implemetado um *framework* que traz a implementação e controle da execução do *workflow* [Terra 2022].

A biblioteca Parsl [Babuji et al. 2019] foi utilizada a para gerenciar a execução das tarefas de forma automática e aproveitar ao máximo os recursos disponíveis. Parsl garante ao *framework* um bom sistema de tolerância a falhas; portabilidade ao configurar e executar o *framework* em diversas infraestruturas computacionais; e facilidade de uso, uma vez que o Parsl é desenvolvido para a linguagem de programação Python, que apresenta um alto nível de abstração, o que permite realizar a implementação do *framework* e a gerência do *workflow* científico.

Como falado anteriormente, os programas acoplados ao modelo podem apresentar paralelismo em sua execução. Com isso, em determinados momentos da execução do *workflow* podem haver recursos ociosos. Para contornar essa desvantagem, o *framework* foi desenvolvido de forma a permitir que múltiplas entradas e *pipelines* sejam executados de forma paralela. Dessa forma, todos os cinco *pipelines* podem ser executados usando como entrada um *dataset*, o que gera um conjunto de diferentes redes filogenéticas ao fim da execução do *framework*.

Contudo, apenas executar múltiplos *pipelines* em uma única execução não é tão eficiente, devido à dependência de dados e o paralelismo divergente entre as tarefas. Com o auxílio do Parsl, o *framework* realiza o controle de programas que utilizam múltiplos *threads*, realizando o empacotamento das tarefas (Figura 2) por meio da dependência de dados entre as mesmas. Com isso, as tarefas são escalonadas de forma a serem executadas de imediato assim que houverem recursos disponíveis e nenhuma dependência de dados vigente, ou seja, a tarefa a ser executada não depende da conclusão de outra tarefa em andamento.

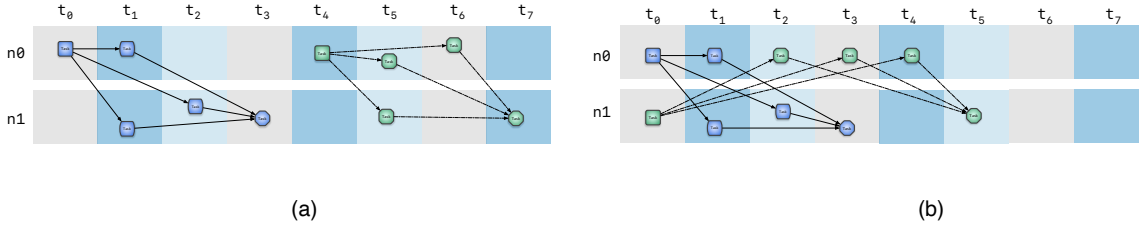

Figura 2. Exemplo de empacotamento. Dois *pipelines* sendo executados em um ambiente com dois núcleos. Em (a) sem empacotamento de tarefas e em (b) com o empacotamento. Fonte: [Terra et al. 2021].

A complexidade dos dados também influencia diretamente o desempenho da construção da rede. Quanto maior é o número de táxons e genes de um experimento, maior é a quantidade de tempo e recursos computacionais necessários. Logo, em estudos que apresentam um grande número de táxons é inviável a execução em máquinas convencionais, sendo requerido um ambiente mais adequado de CAD. O *framework* apre-

sentado neste trabalho foi desenvolvido de forma a ser capaz de ser executado tanto em uma máquina usual quanto em um ambiente CAD.

4.1. Experimento

Para avaliar o desempenho do *framework* foram utilizados três *datasets* contendo alinhamentos de sequências sintéticas, contendo seis táxons e 100, 300 e 1000 genes respectivamente. Esses *datasets* foram usados previamente por [Solís-Lemus and Ané 2016] para avaliar o desempenho do algoritmo SNaQ isoladamente. Essa entrada foi escolhida devido ao seu alto número de genes em cada *dataset*, o que faz com que cada *pipeline* tenha um alto número de tarefas a ser executado.

Todas as execuções foram realizadas utilizando o supercomputador Santos Dumont[1], na fila de execução *cpu_small*, composta por máquinas apresentando processador *Intel(R) Xeon(R) CPU E5-2695 v2* e 66 GB de memória RAM. O desempenho do *framework* foi analisado utilizando a taxa entre o tempo estimado - soma de todos os *pipelines* de uma execução sem o empacotamento de tarefas - e o tempo de execução dos cinco *pipelines* simultaneamente, fazendo uso das melhorias anteriormente apresentadas. Os programas e ferramentas encontravam-se nas seguintes versões durante o experimento: Python v3.8.2, Parsl v1.0, IQ-TREE v2.2.0, ASTRAL v5.7.1, PhyloNet v3.8.2, RAxML v8.2, BUCKy v1.4.4, MrBayes v3.2.7a, Julia v1.5.1 e Java JDK v12.

Dois tipos de experimentos foram executados para cada *dataset* de 100, 300 e 1000 genes usando: (1) **o *framework* de cada *pipeline* por separado** e (2) **o *framework* com todos os cinco *pipelines* simultaneamente**. O tempo total de execução (TTE) foi calculado pelo tempo médio de cada execução repetida cinco vezes. Quanto aos parâmetros dos programas usados, RAxML e IQ-TREE foram executados com seis e dois *threads* respectivamente e com valor de *bootstrap* igual a 1.000; PhyloNet e PhyloNetworks com dez *runs*, três hibridizações máximas e dez *threads*; e o MrBayes foi executado com 1.000.000 gerações e duas *runs*.

5. Resultados e Análises

A Figura 3 apresenta o TTE dos *datasets* de 100 genes (azul claro), 300 genes (azul médio) e 1000 genes (azul escuro) e os cinco primeiros conjuntos de barras representam as execuções dos *pipelines* individualizados (raxml_phylonet, raxml_snaq, mrbayes_snaq, iqtree_phylonet, iqtree_snaq). O grupo seguinte (*5 pipelines*) corresponde ao resultado do nosso *workflow* executando os cinco *pipelines* concomitantemente. Por último, para facilitar a comparação, temos último grupo (Σ teórico) que representa o tempo sintético correspondente ao somatório do tempo dos *pipelines* individualizados (cinco primeiros conjuntos).

Pode ser observado que os *pipelines* que mostraram maior TTE foram os que utilizaram IQ-TREE na construção de árvores (iqtree_phylonet, iqtree_snaq e 5 *pipelines*). O IQ-TREE possui um algoritmo estocástico rápido e eficaz para inferir árvores filogenéticas por MV e ele se compara favoravelmente ao RAxML e PhyML em termos de análises. Contudo, sua execução permite analisar o tamanho das sequências e escolher o número ideal de *threads* a serem usadas para a execução do algoritmo, limitado superiormente por um número máximo.

[1] `https://sdumont.lncc.br/`

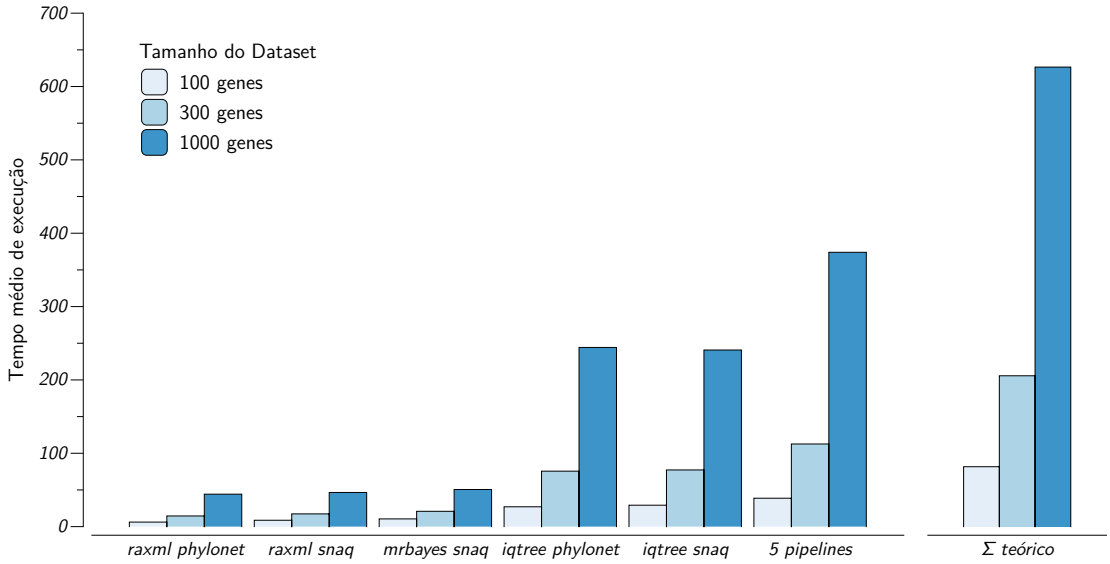

Figura 3. Tempo total de execução de cada *pipeline* por *dataset*.

O penúltimo conjunto de barras representa as execuções do *framework* para **todos os cinco *pipelines* simultaneamente**, que mostra uma diminuição do TTE quando comparado à soma das execuções dos ***frameworks* para os *pipelines* por separado** (último grupo Σ teórico). O mesmo comportamento foi obtido para todos os *datasets* de genes (Figura 3).

Para o *dataset* de 100 genes (azul claro), o TTE da soma dos *frameworks* para os *pipelines* por separado foi de $81,67$ minutos *versus* o TTE dos cinco *pipelines* executados de forma simultânea de $38,73$ minutos, o que resulta em uma melhora de aproximadamente $2,11$ vezes. Já o *dataset* de 300 genes (azul médio) apresentou uma melhora de $1,83$ vezes e o terceiro *dataset* de 1000 genes (azul escuro) apresentou uma melhora, um pouco menor, de $1,68$ vezes.

Os ganhos no TTE são explicados pelo uso de recursos não utilizados pelos *pipelines* individualizados que são explorados quando existem mais tarefas com dependências satisfeitas em decorrência da execução simultânea dos cinco *pipelines* em um único *workflow*. Além disso, como pode ser observado na Figura 1, existem diversos resultados intermediários que passam a ser calculados uma única vez, diminuindo o TTE quando se executam simultaneamente todos os *pipelines*.

6. Considerações finais

A reconstrução da história evolutiva, assim como o estabelecimento de hipóteses que demonstrem as relações filogenéticas de indivíduos, em diferentes níveis de classificação taxonômica desde vírus até organismos complexos como os eucariotos, requerem o uso de várias abordagens e ferramentas, as quais muitas vezes ainda não se encontram disponíveis de maneira integrada. O uso de diferentes abordagens filogenéticas, filogenômicas e evolutivas são necessárias para a inferência da filogenia de espécies e a filogenia de genes.

O presente trabalho apresentou a avaliação de desempenho de um *framework* que encapsula a modelagem de um *workflow* composto por diferentes metodologias para a

construção de redes filogenéticas. O *framework* com a implementação do empacotamento mostrou-se ser até duas vezes mais rápido do que uma execução sem a mesma. Isso também mostra a viabilidade do ambiente de CAD para a construção de redes filogenéticas, apesar dos algoritmos não serem otimizados para tal ambiente, o *framework* é capaz de utilizar os recursos disponíveis nesse tipo de ambiente. Além disso, a execução de múltiplos *pipelines* e *datasets* em uma única instanciação do *framework*, principalmente de dados que possuem um número considerável de genes, permite aumentar consideravelmente a vazão do sistema, utilizando o máximo de recurso possível.

Foi observado que para todas as execuções usando diferentes *datasets* de entrada, utilizar o *framework* para rodar todos os *pipelines* simultaneamente aproveita todas as melhorias implementadas. Pode ser até $2,11$ vezes mais rápido do que executar cada *pipeline* separadamente. Contudo, os resultados mostraram que o desempenho dos *frameworks* decrescem à medida que o número de genes aumenta, provavelmente por falta de ajustes na execução dos múltiplos *pipelines* e o ambiente de CAD.

Como trabalhos futuros, pretende-se avaliar as causas da diminuição da escalabilidade do *framework* à medida em que se aumenta o tamanho do *dataset*. Pretendemos também desenvolver o paralelismo do *framework* para uma ambiente com múltiplos nós computacionais diferentes (não homogênios, ex. com GPUs), de forma a permitir uma vazão ainda maior da execução dos *pipelines* e uso de novas tecnologias.

Referências

Andronico, G., Ardizzone, V., Barbera, R., Becker, B., Bruno, R., Calanducci, A., Carvalho, D., Ciuffo, L., Fargetta, M., Giorgio, E., La Rocca, G., Masoni, A., Paganoni, M., Ruggieri, F., and Scardaci, D. (2011). e-infrastructures for e-science: A global view. *Journal of Grid Computing*, 9(2):155–184.

Ané, C. (2011). Detecting phylogenetic breakpoints and discordance from genome-wide alignments for species tree reconstruction. *Genome Biology and Evolution*, 3:246–258.

Ané, C., Larget, B., Baum, D. A., Smith, S. D., and Rokas, A. (2007). Bayesian estimation of concordance among gene trees. *Molecular biology and evolution*, 24(2):412–426.

Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J. M., Foster, I., Wilde, M., and Chard, K. (2019). Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 25–36. ACM.

Deelman, E., Peterka, T., Altintas, I., Carothers, C. D., van Dam, K. K., Moreland, K., Parashar, M., Ramakrishnan, L., Taufer, M., and Vetter, J. (2018). The future of scientific workflows. *The International Journal of High Performance Computing Applications*, 32(1):159–175.

Huelsenbeck, J. P. and Ronquist, F. (2001). Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755.

Huson, D. H., Rupp, R., and Scornavacca, C. (2010). *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press.

Larget, B. R., Kotha, S. K., Dewey, C. N., and Ané, C. (2010). Bucky: gene tree/species tree reconciliation with bayesian concordance analysis. *Bioinformatics*, 26(22):2910–2911.

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

Mao, Y., Hou, S., Shi, J., and Economo, E. P. (2020). TREEasy: An automated workflow to infer gene trees, species trees, and phylogenetic networks from multilocus data. *Molecular Ecology Resources*, 20(3):832–840.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E., Oliveira, D. D., Cruz, S. M. S. D., Martinho, W., and Murta, L. (2010). Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management*, 5(1):79.

Minh, B. Q., Schmidt, H. A., Chernomor, O., Schrempf, D., Woodhams, M. D., Von Haeseler, A., and Lanfear, R. (2020). Iq-tree 2: new models and efficient methods for phylogenetic inference in the genomic era. *Molecular biology and evolution*, 37(5):1530–1534.

Mirarab, S., Reaz, R., Bayzid, M. S., Zimmermann, T., Swenson, M. S., and Warnow, T. (2014). ASTRAL: genome-scale coalescent-based species tree estimation. *Bioinformatics*, 30(17):i541–i548.

Ott, M., Zola, J., Aluru, S., Johnson, A. D., Janies, D., and Stamatakis, A. (2008). Large-scale phylogenetic analysis on current hpc architectures. *Scientific Programming*, 16(2-3):255–270.

Pouchard, L., Baldwin, S., Elsethagen, T., Jha, S., Raju, B., Stephan, E., Tang, L., and Van Dam, K. K. (2019). Computational reproducibility of scientific workflows at extreme scales. *The International Journal of High Performance Computing Applications*, 33(5):763–776.

Snir, S. and Rao, S. (2012). Quartet maxcut: a fast algorithm for amalgamating quartet trees. *Molecular phylogenetics and evolution*, 62(1):1–8.

Solís-Lemus, C. and Ané, C. (2016). Inferring Phylogenetic Networks with Maximum Pseudolikelihood under Incomplete Lineage Sorting. *PLOS Genetics*, 12(3):e1005896.

Solís-Lemus, C., Bastide, P., and Ané, C. (2017). PhyloNetworks: A Package for Phylogenetic Networks. *Molecular Biology and Evolution*, 34(12):3292–3298.

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

Stenz, N. W., Larget, B., Baum, D. A., and Ané, C. (2015). Exploring tree-like and non-tree-like patterns using genome sequences: an example using the inbreeding plant species arabidopsis thaliana (l.) heynh. *Systematic Biology*, 64(5):809–823.

Taylor, I. J., Deelman, E., Gannon, D. B., Shields, M., et al. (2007). *Workflows for e-Science: scientific workflows for grids*, volume 1. Springer.

Terra, R. (2022). Framework para execução de workflows de redes filogenéticas em ambientes de computação de alto desempenho. Master's thesis, Programa de Pós-Graduação em Modelagem Computacional. Coordenação de pós-graduação - COPGA.

Terra, R., Coelho, M., Cruz, L., Garcia-Zapata, M., Gadelha, L., Osthoff, C., Carvalho, D., and Ocana, K. (2021). Gerência e análises de workflows aplicados a redes filogenéticas de genomas de dengue no brasil. In *Anais do XV Brazilian e-Science Workshop*, pages 49–56. SBC.

Versluis, L. and Iosup, A. (2021). A survey of domains in workflow scheduling in computing infrastructures: Community and keyword analysis, emerging trends, and taxonomies. *Future Generation Computer Systems*, 123:156–177.

Wen, D., Yu, Y., Zhu, J., and Nakhleh, L. (2018). Inferring Phylogenetic Networks Using PhyloNet. *Systematic Biology*, 67(4):735–740.

Yu, J. and Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Record*, 34(3):44.