CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

ARANDUFLOW: RUMO AO USO DE METADADOS DE DESEMPENHO E DOMÍNIO PARA PREVER TEMPO DE EXECUÇÃO DE *WORKFLOWS* EM AMBIENTES DE HPC

André Igor Pereira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do grau de mestre.

Orientadores:
Rafaelli de Carvalho Coutinho
Daniel Cardoso Moraes de Oliveira

Rio de Janeiro,

25 de Setembro de 2026

`AranduFlow`: Rumo ao uso de metadados de desempenho e domínio para prever tempo de execução de *workflows* em ambientes de HPC

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

André Igor Pereira

Aprovada por:

Presidente, Prof. Rafaelli de Carvalho Coutinho, D.Sc. (orientadora)

Prof. Daniel Cardoso Moraes de Oliveira, D.Sc. (coorientador)

Prof. Eduardo Soares Ogasawara, D.Sc.

Prof. Victor Stroele de Andrade Menezes, D.Sc.

Rio de Janeiro,

25 de Setembro de 2026

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

P436 Pereira, André Igor
Aranduflow: rumo ao uso de metadados de desempenho e domínio para prever tempo de execução de workflows em ambientes de HPC / André Igor Pereira. — 2026.
73f. + apêndice : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2026.
Bibliografia : f. 69-73
Orientador: Rafaelli de Carvalho Coutinho
Coorientador: Daniel Cardoso Moraes de Oliveira

1. Computação de alto desempenho. 2. Fluxo de trabalho. 3. Aprendizado do computador. 4. Tempo – Medição. I. Coutinho, Rafaelli de Carvalho. (Orient.). II. Oliveira, Daniel Cardoso Moraes de. (Coorient.). III. Título.

CDD 004.35

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

DEDICATÓRIA

Aos meus sobrinhos, com todo o meu amor.
Dedico este trabalho a vocês como demonstração
de que o conhecimento transforma caminhos e
realiza sonhos. Que vocês vejam nesta conquista
a certeza de que a educação é a ferramenta mais
poderosa para construir o futuro.

AGRADECIMENTOS

Agradeço à minha família, aos meus amigos e à minha namorada pelo apoio e incentivo ao longo desta trajetória. À minha orientadora, professora Rafaelli de Carvalho Coutinho, e ao meu coorientador, professor Daniel Cardoso Moraes de Oliveira, agradeço pela orientação e pelas contribuições fundamentais para esta pesquisa. Agradeço também aos professores e colegas do curso, ao CEFET/RJ e ao PPCIC pela formação e pelas oportunidades proporcionadas.

RESUMO

`AranduFlow`: Rumo ao uso de metadados de desempenho e domínio para prever tempo de execução de *workflows* em ambientes de HPC

André Igor Pereira

Orientadores:
Rafaelli de Carvalho Coutinho
Daniel Cardoso Moraes de Oliveira

Resumo da Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do grau de mestre.

A crescente complexidade dos experimentos científicos tem ampliado o uso de *workflows* para organizar aplicações executadas em ambientes de Computação de Alto Desempenho (HPC). Nesses ambientes, estimar antecipadamente o tempo de execução é uma tarefa desafiadora, devido à diversidade dos dados de entrada, dos parâmetros das aplicações e das configurações da infraestrutura computacional. Além disso, a concorrência por recursos em ambientes compartilhados pode fazer com que execuções semelhantes apresentem durações distintas. Para auxiliar usuários na submissão de *workflows* e no planejamento de seus experimentos, este trabalho apresenta o `AranduFlow`, um *gateway* científico adaptável que utiliza dados históricos e modelos de aprendizado de máquina para estimar faixas de tempo de execução. A proposta integra mecanismos de submissão, coleta de metadados e predição, combinando características dos dados de entrada, parâmetros específicos do domínio e informações da infraestrutura computacional. Embora a arquitetura possa ser adaptada a diferentes aplicações, os modelos preditivos são treinados especificamente para cada *workflow*. A abordagem foi avaliada com dois *workflows* científicos reais, pertencentes aos domínios de bioinformática e astronomia e com perfis distintos de utilização de recursos, executados em infraestruturas heterogêneas de HPC. Os experimentos mostraram que os modelos baseados em árvores superaram o método de referência e alcançaram acurácia superior a 80% nos cenários com faixas temporais mais abrangentes. Os resultados também indicaram a relevância dos metadados de domínio e das características da carga de trabalho para as predições. Em uma nova infraestrutura computacional, a utilização de um pequeno conjunto de execuções locais para calibração produziu uma melhora expressiva no desempenho preditivo. Esses resultados fornecem evidências de que o `AranduFlow` pode apoiar a estimativa do tempo de execução de *workflows* heterogêneos e ser adaptado a diferentes ambientes computacionais.

Palavras-chave:
Predição de tempo de execução, *Workflows* científicos, Computação de Alto Desempenho, Aprendizado de máquina.

Rio de Janeiro,

25 de Setembro de 2026

ABSTRACT

`AranduFlow`: Towards using performance and domain metadata to predict workflow execution time in HPC Environments

André Igor Pereira

Advisors:
Rafaelli de Carvalho Coutinho
Daniel Cardoso Moraes de Oliveira

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

The increasing complexity of scientific experiments has expanded the use of workflows to organize applications executed in High-Performance Computing (HPC) environments. In these environments, predicting execution time in advance is challenging because of variations in input data, application parameters, and computing infrastructure configurations. Furthermore, resource contention in shared environments can cause similar executions to have different durations. To assist users with workflow submission and experiment planning, this dissertation presents `AranduFlow`, an adaptable science gateway that uses historical data and machine learning models to estimate execution-time ranges. The proposed approach integrates submission, metadata collection, and prediction mechanisms by combining input-data characteristics, domain-specific parameters, and computing infrastructure information. Although the architecture can be adapted to different applications, the predictive models are trained specifically for each workflow. The approach was evaluated using two real-world scientific workflows from the bioinformatics and astronomy domains, with distinct resource-utilization profiles, executed on heterogeneous HPC infrastructures. The experiments showed that tree-based models outperformed the reference method and achieved accuracies above 80% in scenarios using broader execution-time ranges. The results also indicated the relevance of domain metadata and workload characteristics to the predictions. When applied to a new computing infrastructure, calibration using a small set of local executions produced a substantial improvement in predictive performance. These results provide evidence that `AranduFlow` can support execution-time estimation for heterogeneous workflows and can be adapted to different computing environments.

Key-words:
Execution time prediction, scientific workflows, high-performance computing, machine learning.

Rio de Janeiro,

25 de Setembro de 2026

Sumário

I Introdução **1**

II Referencial Teórico **5**

II.1 *Workflows* Científicos 5

II.2 Ambientes de Computação de Alto Desempenho 8

II.3 Aprendizado de Máquina 9

II.3.1 Formulação do Problema e Discretização por Quantis 9

II.3.2 Modelos Lineares de Regressão e Classificação 10

II.3.3 Árvores de Decisão 11

II.3.4 Florestas Aleatórias 12

II.3.5 Métricas de Avaliação de Desempenho 13

II.4 Aprendizado por Transferência e Calibração de Modelos 15

II.5 Dados Históricos e Reprodutibilidade 16

III Trabalhos relacionados **17**

III.1 Análise dos Trabalhos Relacionados 18

III.2 Discussão Comparativa e Lacunas da Literatura 22

IV Arquitetura e Funcionamento do *Gateway* `AranduFlow` **24**

IV.1 Escopo e Objetivos Arquiteturais 24

IV.2 Representação de uma Execução e Entrada do Modelo Preditivo 26

IV.3 Arquitetura Geral do `AranduFlow` 27

IV.4 Interface *Web* 29

IV.4.1 Cadastro do *Workflow* e Definição de seus Parâmetros 29

IV.4.2 Configuração de uma Execução 30

IV.5 Gerenciador de Predição e Persistência 33

IV.6 Seletor e Conector de Infraestrutura 35

IV.7 Executores de *Workflows* 36

IV.8 Registro do Tempo de Execução Observado 37

IV.9 Considerações Finais 38

V Avaliação Experimental do `AranduFlow` 39

V.1 Delineamento da Avaliação Experimental 39

V.1.1 Configuração do Ambiente 39

V.1.2 Configuração dos Experimentos 41

V.1.3 Discretização do Tempo de Execução 42

V.1.4 Modelos de Aprendizado de Máquina no `AranduFlow` 42

V.1.5 Estratégias de Calibração Dinâmica Local 45

V.1.6 Disponibilidade dos Artefatos e Reprodutibilidade 46

V.2 Análise Exploratória dos Dados 47

V.2.1 *Workflow* NMFSt.P 47

V.2.2 *Workflow* Montage 48

V.3 Resultados e Discussão 49

V.3.1 Resultados para o *Workflow* NMFSt.P 50

V.3.2 Resultados para o *Workflow* Montage 52

V.3.3 Generalização Trans-Ambiente 55

V.4 Síntese da Avaliação 62

V.5 Limitações e Ameaças à Validade 63

V.6 Considerações Finais 64

VI Conclusão 65

VI.1 Limitações 67

VI.2 Trabalhos Futuros 67

Referências 69

A Dicionários de Dados 74

A.1 Dicionário de Dados do NMFSt.P 74

A.2 Dicionário de Dados do Montage 75

A.3 Vetor de Características da Infraestrutura 75

B Intervalos de Discretização por Quantis 77

B.1 Intervalos de Discretização para o Cenário de 3 Classes 77

B.2 Intervalos de Discretização para o Cenário de 7 Classes 78

B.3 Intervalos de Discretização para o Cenário de 10 Classes 79

B.4 Intervalos de Discretização para o Cenário de 15 Classes 80

Lista de Figuras

II.1 *Workflow* executado pelo *notebook* NMFSt.P. Fonte: Ferrari et al. [2023] 6

II.2 *Workflow* executado pelo Montage 7

III.1 Fluxograma do processo de busca, triagem e seleção dos estudos. 19

IV.1 Arquitetura lógica do *gateway* `AranduFlow`. 28

V.1 Distribuição do tempo de execução das múltiplas execuções do *workflow* NMFSt.P. 47

V.2 Tempo de execução do NMFSt.P (em segundos) em função do tamanho do arquivo de entrada (em *bytes*). 48

V.3 Tempo de execução do NMFSt.P (em segundos) em função do número de *threads* alocadas. 48

V.4 Distribuição do tempo de execução das múltiplas execuções do *workflow* Montage. 49

V.5 Tempo de execução do Montage (em segundos) em função do tamanho total de entrada (em MB). 49

V.6 Atributos mais bem ranqueados de acordo com a importância de Gini no modelo Árvore de Decisão (cenário com cinco classes) no NMFSt.P. 51

V.7 Matrizes de confusão para o classificador Árvore de Decisão no *workflow* NMFSt.P. 52

V.8 Atributos mais bem ranqueados de acordo com a importância de Gini no modelo Árvore de Decisão (cenário com cinco classes) no Montage. 53

V.9 Matrizes de confusão para o classificador Árvore de Decisão no *workflow* Montage. 54

V.10 Matrizes de confusão da Árvore de Decisão calibrada no *Host* C para o cenário com três classes. 59

V.11 Sensibilidade da acurácia média e desvio padrão em função do tamanho do conjunto de sondagem local (N_{cal}) para 3, 5 e 10 classes no *Host* C. 61

Lista de Tabelas

III.1 Comparação entre abordagens de predição e suporte à execução de *workflows* científicos 23

IV.1 Campos de cadastro de *workflows*, atividades e parâmetros científicos no `AranduFlow`. 29

IV.2 Campos de configuração de uma execução do NMFSt.P no `AranduFlow`. 31

V.1 Características de *hardware* e sistema operacional das classes de infraestrutura *High Performance Computing* (HPC) avaliadas 40

V.2 Distribuição de frequências e intervalos de tempo das classes de tempo de execução (cenário com cinco classes) no NMFSt.P. 42

V.3 Distribuição de frequências e intervalos de tempo das classes de tempo de execução (cenário com cinco classes) no Montage. 43

V.4 Acurácia e F1-Macro para os modelos preditivos no *workflow* NMFSt.P 50

V.5 Acurácia e F1-Macro para os modelos preditivos no *workflow* Montage 52

V.6 Desempenho do melhor modelo base no teste agregado e nos subconjuntos de teste por infraestrutura. 55

V.7 Comparação de Acurácia e F1-Macro no *Host* C usando as representações por Pontuações de *benchmark* vs. vetor de características 57

V.8 Comparação de Acurácia obtida no *Host* C sob as diferentes estratégias de calibração local. 59

V.9 Comparação de F1-Macro obtido no *Host* C sob as diferentes estratégias de calibração local. 59

A.1 Dicionário de dados do NMFSt.P. 74

A.2 Dicionário de dados do Montage. 75

A.3 Vetor de características da infraestrutura comum ao NMFSt.P e ao Montage. 76

B.1 Distribuição de frequências e intervalos de tempo (cenário com três classes) no NMFSt.P. 77

B.2 Distribuição de frequências e intervalos de tempo (cenário com três classes) no Montage. 77

B.3 Distribuição de frequências e intervalos de tempo (cenário com sete classes) no NMFSt.P. 78

B.4 Distribuição de frequências e intervalos de tempo (cenário com sete classes) no Montage. 78
B.5 Distribuição de frequências e intervalos de tempo (cenário com dez classes) no NMFSt.P. 79
B.6 Distribuição de frequências e intervalos de tempo (cenário com dez classes) no Montage. 79
B.7 Distribuição de frequências e intervalos de tempo (cenário com quinze classes) no NMFSt.P. 80
B.8 Distribuição de frequências e intervalos de tempo (cenário com quinze classes) no Montage. 81

Lista de Abreviações

AM	Aprendizado De Máquina	2, 9, 18
CPU	*Central Processing Unit*	1, 3, 8, 20, 21, 39, 40, 48, 50, 57, 65
CRCPS	*Configurable Resource Cost Prediction Service*	18, 20, 21, 23
CWL	*Common Workflow Language*	1, 5
DAG	*Directed Acyclic Graph*	5, 65
DNA	Ácido Desoxirribonucleico	6
DTC	*Decision Tree Classifier*	11, 43, 44, 50, 51, 52, 58, 65
E/S	Entrada/Saída	2, 3, 7, 20, 21, 39, 40, 62, 64, 65
FITS	*Flexible Image Transport System*	8
GPU	*Graphics Processing Unit*	1, 8
GRASP	*Greedy Randomized Adaptive Search Procedure*	20
HPC	*High Performance Computing*	1, 2, 3, 4, 5, 6, 8, 9, 15, 17, 18, 20, 21, 22, 23, 24, 25, 38, 39, 40, 54, 56, 58, 65, 67
ICEIS	*International Conference on Enterprise Information Systems*	66
IPAC	*Infrared Processing and Analysis Center*	7
JSON	*JavaScript Object Notation*	25, 27, 32, 33, 35, 36, 37, 62
KNN	*K-Nearest Neighbors*	21
LIR	*Linear Regression*	10, 43, 50, 52, 58, 65
LOR	*Logistic Regression*	11, 43, 50, 57, 65, 66
MDI	*Mean Decrease in Impurity*	13
MSA	*Multiple Sequence Alignment*	6
NJ	*Neighbor-Joining*	6
NMFST.P	*Notebook for Parallel Identification of Frequent Subtrees*	3, 6, 7, 39, 41, 47, 48, 54, 65, 66
OLS	*Ordinary Least Squares*	10
OOB	*Out-of-Bag*	13
RAM	*Random Access Memory*	40, 53, 56
RFC	*Random Forest Classifier*	12, 20, 43, 44, 50, 52, 54, 57, 60, 65, 66
RL	*Reinforcement Learning*	21
RNA	Ácido Ribonucleico	6

SFTP	*SSH File Transfer Protocol*	35
SGE	*Sun Grid Engine*	35, 36, 37, 38, 40
SQL	*Structured Query Language*	22
SSH	*Secure Shell*	35, 37
WFMS	*Workflow Management System*	1

Capítulo I Introdução

Nos últimos anos, a computação passou por uma transformação profunda, caracterizada pelo aumento exponencial no volume de dados gerados e pela complexidade dos experimentos científicos. Áreas como astronomia, bioinformática, meteorologia e física nuclear dependem fortemente de simulações em larga escala e *pipelines* de análise de dados. Nesse cenário, os *workflows* científicos emergiram como uma abstração conceitual e prática amplamente adotada para representar essas aplicações complexas [de Oliveira et al., 2019]. Um *workflow* representa uma sequência estruturada de tarefas computacionais interconectadas por dependências de dados, permitindo capturar a organização lógica, a concorrência e o fluxo de informações ao longo da execução da aplicação.

Existem diversas abordagens para especificar, orquestrar e gerenciar a execução de *workflows*. Usuários individuais muitas vezes desenvolvem soluções *ad-hoc*, utilizando linguagens de programação de propósito geral como `Python` ou *scripts* de *shell*. Embora essa prática ofereça flexibilidade e controle refinado sobre a execução de tarefas locais, torna-se ineficiente para gerenciar execuções distribuídas em larga escala. Para contornar essa limitação, foram desenvolvidos Sistemas de Gerenciamento de *workflows* (do inglês, *Workflow Management System* (WfMS)), que fornecem abstrações de alto nível, tratamento automático de falhas, portabilidade e facilidade de escalonamento. Exemplos na literatura e na indústria incluem o `Apache Airflow`[1], `Pegasus` [Deelman et al., 2015], `Nextflow` [Di Tommaso et al., 2017], `Snakemake` [Köster and Rahmann, 2012] e *Common Workflow Language* (CWL) [Amstutz et al., 2016].

Muitos desses *workflows* manipulam conjuntos de dados massivos e envolvem processamento contínuo intenso, aproveitando-se de infraestruturas robustas de Computação de Alto Desempenho (do inglês, *High Performance Computing* (HPC)) [Afgan et al., 2018]. O uso de ambientes HPC, como supercomputadores, *clusters* de computação e plataformas em nuvem otimizadas para alto desempenho [Friese et al., 2025], é essencial para viabilizar execuções em tempo hábil. Esses ambientes caracterizam-se por recursos computacionais massivos e heterogêneos, compreendendo unidades centrais de processamento (do inglês, *Central Processing Unit* (CPU)), unidades de processamento gráfico (do inglês, *Graphics Processing Unit* (GPU)s), sistemas de arquivos paralelos distribuídos e redes de interconexão de altíssima velocidade e baixa latência.

Contudo, a utilização eficiente dessas plataformas apresenta desafios consideráveis. Infraestru-

[1] https://airflow.apache.org/

turas de HPC são tipicamente compartilhadas entre múltiplos usuários concorrentes. As tarefas dos *workflows* disputam recursos críticos, como a largura de banda de Entrada/Saída (E/S), a *cache* dos sistemas de arquivos e a memória principal. Essa concorrência gera volatilidade temporal, fazendo com que execuções idênticas do mesmo *workflow* apresentem durações flutuantes dependendo da carga do ambiente computacional [Boito et al., 2025; Friese et al., 2025]. Nesse contexto, estimar o tempo de execução dos *workflows* torna-se fundamental. Essa estimativa apoia os usuários finais no planejamento de suas atividades de pesquisa, auxilia administradores no provisionamento adequado de recursos e fornece suporte analítico essencial a escalonadores e agendadores de tarefas para otimizar o uso do ambiente computacional de produção.

Apesar da importância estratégica de estimar o tempo de execução de *workflows* em ambientes de HPC, formular e solucionar esse problema envolve desafios conceituais e metodológicos significativos. Tradicionalmente, as tentativas de estimar o tempo de execução apoiam-se em modelos de regressão clássicos para prever o tempo de execução exato de uma tarefa ou de um *pipeline* completo [Alvarenga et al., 2025; Yildirim et al., 2026]. No entanto, a predição contínua apresenta alta sensibilidade a ruídos operacionais e à concorrência de recursos, o que compromete sua utilidade prática.

Diante disso, reformular a predição contínua como um problema de classificação multiclasse, dividindo o tempo de execução em intervalos ou faixas preditivas discretas, surge como uma alternativa robusta. No entanto, o projeto dessa abordagem enfrenta o desafio de definir faixas informativas. Divisões manuais ou baseadas em intervalos uniformes frequentemente resultam em classes severamente desbalanceadas devido à assimetria na distribuição dos tempos de execução. Isso pode introduzir viés no treinamento de algoritmos de Aprendizado de Máquina (AM), concentrando as previsões na classe majoritária.

Além disso, as soluções propostas na literatura apresentam limitações de generalização e de escopo de atributos. A primeira dessas limitações consiste no direcionamento a domínios específicos, uma vez que muitos trabalhos preditivos são projetados de forma estritamente voltada a um único *workflow* científico ou a um tipo restrito de *workflow* [Ribeiro et al., 2024; Ocaña et al., 2020], o que dificulta sua adaptação quando novas aplicações científicas são introduzidas no ecossistema. Outro desafio reside no acoplamento a plataformas de execução: várias abordagens dependem de métricas intrínsecas de baixo nível de um *hardware* particular [Porto et al., 2026; de C. Coutinho et al., 2016], podendo perder precisão diante da heterogeneidade de plataformas que caracteriza o HPC moderno. Por fim, observa-se frequentemente um foco exclusivo nas métricas de infraestrutura. Modelos típicos predizem a duração com base apenas em recursos computacionais alocados, como a quantidade de *threads* e a memória solicitada, e no tamanho dos arquivos de entrada em disco [Boito et al., 2025], negligenciando metadados de domínio, como algoritmos selecionados na configuração

do *workflow* ou parâmetros de execução, que podem exercer influência relevante sobre o esforço computacional e a variabilidade do tempo de execução.

Diante desse contexto, formula-se a seguinte questão de pesquisa: em que medida uma abordagem baseada em aprendizado de máquina supervisionado, que combina metadados de domínio, características dos dados de entrada e atributos da infraestrutura, permite estimar faixas de tempo de execução de *workflows* científicos e adaptar essas estimativas a diferentes ambientes computacionais?

Com o intuito de abordar esse problema, esta dissertação estabelece um conjunto de metas voltadas à concepção, implementação e validação experimental de uma arquitetura preditiva baseada em aprendizado de máquina, acoplada a um sistema de submissão. O objetivo geral desta dissertação é implementar e avaliar o `AranduFlow` [2], uma arquitetura de *gateway* científico adaptável que utiliza modelos de aprendizado de máquina supervisionados para estimar faixas de tempo de execução de *workflows* científicos em ambientes HPC compartilhados, integrando metadados de infraestrutura e de domínio de aplicação. Adicionalmente, investiga-se a adaptação das estimativas a ambientes computacionais não observados durante o treinamento.

Cabe destacar a distinção entre a generalidade da arquitetura proposta e o escopo dos elementos específicos de cada *workflow*: embora a estrutura arquitetural e os mecanismos compartilhados de configuração, predição, submissão e persistência do `AranduFlow` sejam reutilizáveis, cada *workflow* integrado requer a definição de seus próprios parâmetros, regras de transformação, atributos preditivos, executor e modelo treinado. Cada modelo é construído com base no respectivo histórico de execução e nos metadados específicos do domínio da aplicação. Assim, a contribuição de generalidade deste trabalho reside na arquitetura e no processo de integração de novos *workflows*, e não na proposição de um único modelo universal para todos os domínios.

Para atingir o objetivo geral proposto, definem-se os seguintes objetivos específicos:

1. Projetar a arquitetura do *gateway* científico `AranduFlow`, garantindo modularidade na ingestão de dados e extensibilidade a novos tipos de *workflows*.

2. Formular a modelagem preditiva temporal como uma classificação multiclasse, fundamentada na discretização por quantis de igual frequência, para contornar o desbalanceamento dos dados históricos.

3. Implementar coletores e *pipelines* de engenharia de atributos capazes de extrair variáveis de infraestrutura computacional e metadados de domínio específicos para dois estudos de caso de perfis distintos: o *workflow Notebook for Parallel Identification of Frequent Subtrees* (NMFSt.P) (intensivo em CPU) e o *workflow* Montage (intensivo em E/S).

[2] O nome “Arandu” provém da língua Guarani e significa “sabedoria” ou “conhecimento” [Proyecto Montoya], refletindo a capacidade do sistema de aprender a partir de execuções anteriores e realizar previsões informativas.

4. Avaliar a influência de múltiplas resoluções de quantis de tempo (3, 5, 7, 10 e 15 classes de tempo) no desempenho dos modelos preditivos.

5. Treinar, validar e comparar diversos algoritmos de classificação supervisionada contra um *baseline* operacional, sob um protocolo experimental com *hold-out* aleatório e validação cruzada estratificada no conjunto de treinamento.

6. Realizar análises de importância de atributos para estimar empiricamente o impacto relativo dos metadados de domínio e dos atributos de infraestrutura na predição das faixas de tempo de execução.

7. Avaliar a generalização dos modelos preditivos em uma infraestrutura não observada durante o treinamento e mensurar os ganhos obtidos por estratégias de calibração utilizando execuções locais.

As principais contribuições desta dissertação organizam-se em cinco frentes. A primeira consiste no projeto e no desenvolvimento do `AranduFlow`, uma arquitetura extensível de *gateway* científico para configuração, predição, submissão e registro de execuções de *workflows* em ambientes de HPC, integrada a uma camada preditiva de suporte ao usuário. A extensibilidade da proposta reside nos mecanismos de integração, coleta de metadados e execução, enquanto cada *workflow* requer a definição de seus próprios atributos e o treinamento de um modelo preditivo específico. A segunda contribuição corresponde à formulação da estimativa do tempo de execução como um problema de classificação multiclasse, com faixas temporais definidas por quantis e modelos que combinam metadados de domínio, características dos dados de entrada e atributos da infraestrutura. A terceira consiste na avaliação experimental da abordagem por meio de dois estudos de caso com perfis computacionais distintos, executados em diferentes infraestruturas e avaliados em múltiplas granularidades de faixas temporais, utilizando acurácia, F1-Macro e comparação com um *baseline* operacional. A quarta contribuição compreende a análise da importância relativa dos atributos e a investigação da adaptação dos modelos a uma infraestrutura não observada durante o treinamento, incluindo a avaliação de estratégias de calibração com execuções locais. Por fim, a quinta contribuição é a disponibilização pública da implementação e da base de dados de proveniência produzida nos experimentos, favorecendo a reprodutibilidade e a realização de novas avaliações.

O restante deste trabalho está organizado da seguinte forma. O Capítulo II apresenta os fundamentos teóricos necessários à compreensão do trabalho. O Capítulo III discute os trabalhos relacionados. O Capítulo IV descreve a arquitetura e o funcionamento do `AranduFlow`, detalhando seus componentes e a interação do usuário com o *gateway*. O Capítulo V apresenta o delineamento da avaliação experimental, analisa os dados e discute os resultados obtidos. Por fim, o Capítulo VI apresenta considerações finais e indicações para trabalhos futuros.

Capítulo II Referencial Teórico

Este capítulo apresenta os fundamentos conceituais necessários para a compreensão da execução e da análise de *workflows* científicos em ambientes de HPC, bem como os princípios associados à predição do tempo de execução por meio de técnicas de aprendizado de máquina. Inicialmente, discute-se a natureza e a modelagem formal de *workflows* científicos. Em seguida, são apresentados aspectos dos ambientes HPC e a necessidade de mecanismos de gerenciamento e de suporte à decisão. Posteriormente, aborda-se o problema de predição do tempo de execução sob diferentes formulações, incluindo sua transformação em um problema de classificação e a introdução de conceitos de aprendizado por transferência e de calibração de modelos para portabilidade trans-ambiente. Por fim, examinam-se os fundamentos da avaliação de modelos e o papel dos dados históricos e dos metadados de execução como base para abordagens orientadas a dados.

II.1 *Workflows* Científicos

Os *Workflows* são abstrações utilizadas para automatizar aplicações complexas, científicas ou não, compostas por múltiplas tarefas interdependentes. Cada tarefa representa uma etapa computacional, como processamento de dados, limpeza de dados ou redução de dados, enquanto as dependências definem a ordem de execução e, em alguns casos, o fluxo de dados entre as tarefas [Deelman et al., 2015]. De forma geral, um *workflow* pode ser representado como um grafo acíclico direcionado (do inglês, *Directed Acyclic Graph* (DAG)), denotado por $\mathcal{W} = (V, A)$, em que V representa o conjunto de vértices associados às tarefas ou entidades de dados, e A representa o conjunto de arestas que expressam suas dependências.

Workflows são amplamente adotados em diversos domínios científicos e de engenharia intensivos em dados e computação, incluindo bioinformática, astronomia, modelagem climática e física. Esses *workflows* fornecem uma forma estruturada de representar análises complexas como tarefas interconectadas, com dependências explícitas de dados, o que permite uma execução escalável e reprodutível. Para apoiar a implantação e a execução desses *workflows* em infraestruturas de HPC e supercomputadores, diversos sistemas de gerenciamento de *workflows*, *e.g.*, `Pegasus` [Deelman et al., 2015], `Nextflow` [Di Tommaso et al., 2017] e `Snakemake` [Köster and Rahmann, 2012], foram desenvolvidos. Além disso, *frameworks* de especificação, como a CWL [Amstutz et al., 2016], for-

necem descrições padronizadas e portáveis de *workflows*. Em conjunto, esses sistemas e *frameworks* permitem a execução coordenada de aplicações heterogêneas ao se integrarem a escalonadores de lote e gerenciadores de recursos, abstraindo detalhes de execução de baixo nível e utilizando de forma eficiente recursos computacionais em larga escala.

Entre esses domínios, a bioinformática destaca-se como uma das principais áreas que utilizam *workflows* científicos. *Workflows* em bioinformática envolvem múltiplas tarefas computacionalmente intensivas e utilizam ferramentas heterogêneas, beneficiando-se de execuções reprodutíveis, escaláveis e paralelas em ambientes HPC. Um exemplo representativo nesse domínio é o *workflow* NMFSt.P, ilustrado na Figura II.1. O NMFSt.P é um *workflow* voltado à descoberta de padrões estruturais recorrentes em grandes coleções de árvores filogenéticas, com ênfase na identificação de subárvores frequentes nas topologias das árvores.

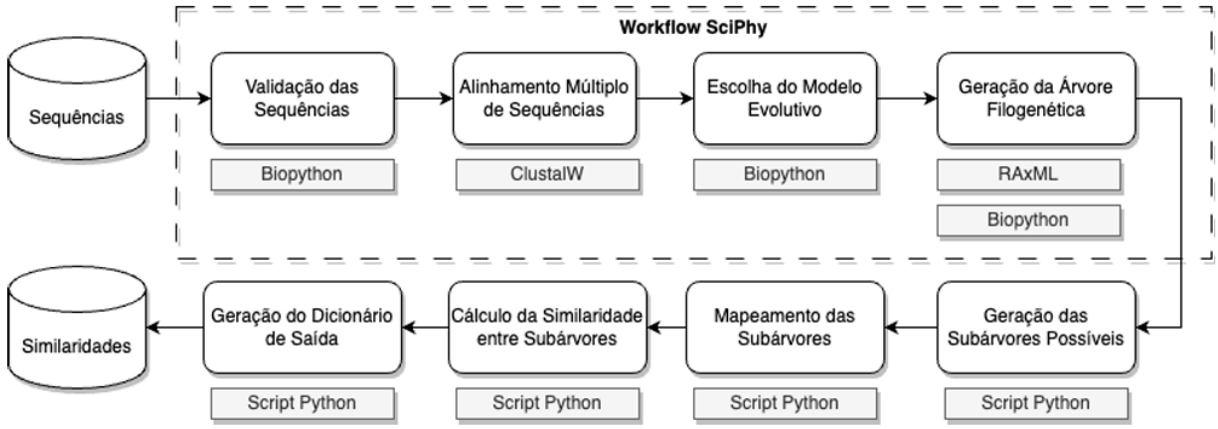


Figura II.1: *Workflow* executado pelo *notebook* NMFSt.P. Fonte: Ferrari et al. [2023]

O *workflow* inicia-se com uma tarefa de validação de sequências, que verifica a integridade e o formato dos dados de entrada. Essa validação é realizada por meio da biblioteca `Biopython`, garantindo que todos os arquivos de sequência fornecidos estejam no formato FASTA. Após a validação, o *workflow* prossegue para a etapa de alinhamento múltiplo de sequências (em inglês, *Multiple Sequence Alignment* (MSA)). Na implementação atual do NMFSt.P, o `ClustalW` é adotado como ferramenta padrão de alinhamento. O alinhamento resultante é armazenado em um arquivo `.ALN`.

Uma vez que as sequências de Ácido Desoxirribonucleico (DNA), Ácido Ribonucleico (RNA) e aminoácidos estejam alinhadas, aplica-se a etapa de escolha do modelo evolutivo, na qual se definem as premissas evolutivas que fundamentam a reconstrução da árvore filogenética. Essa tarefa também é suportada pela biblioteca `Biopython`. Por padrão, o NMFSt.P utiliza o método *Neighbor-Joining* (NJ), uma abordagem amplamente utilizada, baseada em distância, para reconstrução de árvores filogenéticas a partir de dados de sequências moleculares [Saitou and Nei, 1987]. Com o modelo evolutivo estabelecido, o *workflow* avança para a etapa de construção da árvore filogenética. Nesse

estágio, os usuários podem utilizar ferramentas consolidadas de inferência filogenética, como `RAxML` ou `MrBayes`, ou o próprio `Biopython`. O resultado dessa atividade é uma coleção de árvores filogenéticas que servirá de base para a análise de padrões estruturais.

Quando a geração das árvores é concluída, o NMFSt.P inicia a descoberta de subárvores frequentes. A primeira tarefa é a geração de subárvores, que enumera todas as possíveis subárvores contidas em cada árvore filogenética. Cada subárvore identificada é armazenada em um arquivo individual. O procedimento de geração de subárvores é executado de forma independente para cada árvore filogenética. Após a geração de todas as subárvores, o *workflow* realiza a etapa de mapeamento de subárvores, na qual as subárvores são classificadas de acordo com seu tamanho. Em paralelo, o NMFSt.P constrói uma matriz de ocorrência de subárvores que contém o índice de similaridade para cada par de subárvores.

Outro domínio de grande relevância científica que se beneficia amplamente da automação por meio de *workflows* científicos é a astronomia. Processamentos de imagens astronômicas frequentemente lidam com volumes massivos de dados coletados por telescópios, exigindo etapas sistemáticas de projeção espacial, correção de radiação de fundo e coadição física para gerar imagens sinóticas do céu (mosaicos). Um exemplo clássico e amplamente utilizado na literatura de e-Ciência é o *workflow* Montage [Bharathi et al., 2008; Deelman et al., 2015], desenvolvido pelo *Infrared Processing and Analysis Center* (IPAC) do Caltech.

Nesta dissertação, para avaliar o `AranduFlow` em um cenário intensivo em E/S, adota-se um fluxo simplificado baseado na primeira parte do tutorial oficial do Montage, focado na geração de um mosaico astronômico sem a aplicação de correções de radiação de fundo. A execução desse fluxo compreende as tarefas mostradas na Figura II.2.

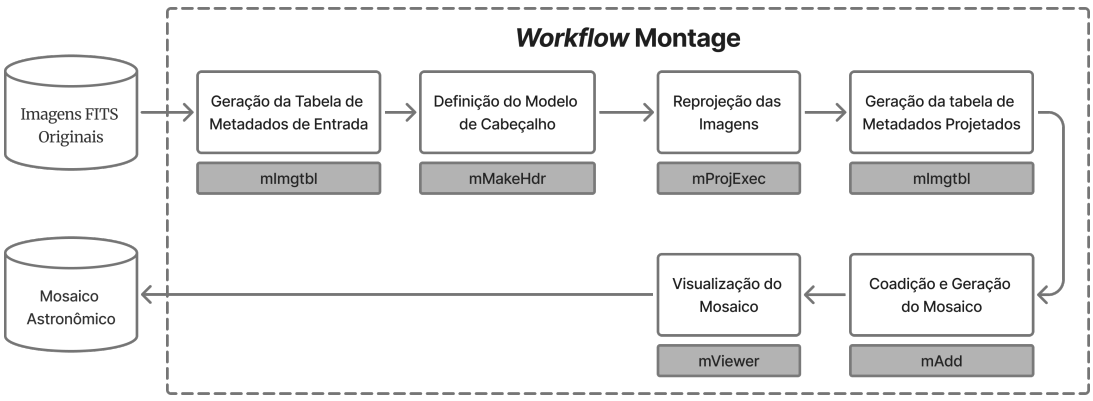


Figura II.2: *Workflow* executado pelo Montage

O fluxo inicia-se com a geração da tabela de metadados de entrada, realizada pelo módulo

`mImgtbl`. Nessa etapa, todo o diretório contendo as imagens originais no formato *Flexible Image Transport System* (FITS) é lido, e os cabeçalhos dos arquivos são inspecionados para extrair coordenadas celestes e parâmetros de projeção. Como resultado, é gerado um arquivo de tabela consolidada com os metadados das imagens de entrada. Em seguida, define-se o modelo de cabeçalho. A partir da tabela de metadados das imagens de entrada, gera-se um arquivo de cabeçalho padrão FITS que define a projeção espacial, a escala de *pixels*, a orientação e a área total a ser coberta pelo mosaico final.

Com o modelo de cabeçalho definido, o fluxo prossegue para a reprojeção das imagens, realizada pelo módulo `mProjExec`. Essa etapa utiliza o arquivo de cabeçalho padrão e a tabela de metadados para reprojetar as imagens originais para um sistema de coordenadas celestes e uma escala geográfica unificados. As imagens resultantes são armazenadas como arquivos FITS intermediários reprojetados. Após a reprojeção, é gerada uma nova tabela de metadados, novamente por meio do módulo `mImgtbl`, agora considerando a pasta que contém as imagens projetadas. Essa tabela atualizada descreve os arquivos intermediários e fornece as informações necessárias para coordenar a consolidação física final.

Por fim, o fluxo realiza a coadição e geração do mosaico com o módulo `mAdd`. Utilizando a tabela de metadados projetados e o modelo de cabeçalho, os pixels das imagens sobrepostas são intercalados e normalizados, produzindo a imagem FITS final unificada do mosaico astronômico, sem a aplicação de correções de brilho de fundo.

II.2 Ambientes de Computação de Alto Desempenho

Ambientes HPC são frequentemente organizados como *clusters* compostos por múltiplos nós interconectados, nos quais as aplicações podem explorar diferentes formas de paralelismo para reduzir o tempo total de processamento. Esses ambientes podem ainda incluir arquiteturas heterogêneas, combinando CPUs, GPUs e outros aceleradores especializados. Essa heterogeneidade de *hardware* entre os nós e entre diferentes infraestruturas pode impactar significativamente o desempenho das aplicações.

Execuções em ambientes HPC são tipicamente gerenciadas por sistemas de escalonamento de tarefas (*e.g.*, `Slurm`, `PBS` ou `Torque`), nos quais as aplicações são submetidas como *jobs* que aguardam em filas até a alocação dos recursos necessários.

A execução nesses ambientes ocorre sob condições realistas de compartilhamento e uso multiusuário nas quais múltiplas aplicações competem simultaneamente por recursos computacionais. Esse contexto introduz variabilidade no desempenho observado, tanto devido a efeitos de contenção de recursos quanto a diferenças de configuração entre infraestruturas e ambientes de execução [Friese et al., 2025; Boito et al., 2025].

Nesse cenário, o suporte à decisão para a submissão e para o planejamento de execuções torna-se particularmente relevante. A capacidade de estimar antecipadamente o tempo total de execução de um *workflow* científico permite apoiar o planejamento eficiente de tarefas e otimizar a ocupação dos nós de processamento.

II.3 Aprendizado de Máquina

O AM é uma subárea da Inteligência Artificial voltada ao desenvolvimento de algoritmos estatisticamente fundamentados, capazes de extrair padrões e aprender correlações a partir de conjuntos de dados históricos [Russell and Norvig, 2020]. Em vez de seguir regras explicitamente programadas, os modelos de AM ajustam seus parâmetros internos por meio de processos de otimização, permitindo realizar previsões precisas sobre novas instâncias inéditas.

No âmbito do gerenciamento de *workflows* científicos e de infraestruturas de HPC, os modelos de AM têm sido aplicados para estimar demandas de recursos computacionais e prever o tempo de execução de tarefas ou de *pipelines* completos, fornecendo uma base analítica sólida para o suporte à decisão [Sobrinho and Paumgartten, 2021; Bader et al., 2024].

Os paradigmas de aprendizado são tipicamente classificados em supervisionados, não supervisionados e por reforço [Bishop, 2006; Mitchell, 1997]. Esta dissertação centra-se no aprendizado supervisionado, no qual o modelo é treinado a partir de um conjunto de treinamento composto por pares de entrada-saída rotulados, denotado por $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, em que $\mathbf{x}_i \in \mathbb{R}^d$ representa um vetor contendo as d características (*features*) da execução e y_i denota a variável alvo associada [James et al., 2023; Géron, 2022]. Dependendo da natureza de y_i, o problema enquadra-se em regressão (quando a variável alvo é contínua, $y_i \in \mathbb{R}$) ou classificação (quando assume valores em um conjunto discreto de C classes, $y_i \in \{1, \ldots, C\}$).

II.3.1 Formulação do Problema e Discretização por Quantis

Embora o tempo de execução de um *workflow* científico seja por natureza uma variável contínua $t \in \mathbb{R}^+$, prever um valor contínuo exato em ambientes HPC compartilhados é extremamente desafiador. Pequenas variações de *hardware* produzem erros significativos em modelos de regressão pontuais.

Para contornar essa limitação e conferir maior robustez prática ao preditor, o `AranduFlow` formula a estimativa de tempo como uma tarefa de classificação multiclasse, por meio da discretização da variável de tempo contínuo t em um conjunto de C intervalos correspondentes às classes discretas $y \in \mathcal{Y} = \{Q_1, Q_2, \ldots, Q_C\}$ [Hastie et al., 2009; James et al., 2023].

Para garantir que a divisão do tempo preserve a equidade de distribuição de amostras históricas — evitando o desbalanceamento de classes em distribuições assimétricas de cauda longa —, adota-

se a discretização baseada em quantis. O espaço contínuo é particionado utilizando $C-1$ limites de divisão, definidos pelos percentis de ordem k/C para $k \in \{1, \ldots, C-1\}$.

Desta forma, a probabilidade empírica *a priori* de que uma amostra pertença a qualquer uma das classes de quantis Q_k é aproximadamente uniforme, conforme Equação II.1.

$$P(y = Q_k) \approx \frac{1}{C}, \quad \forall k \in \{1, \ldots, C\} \tag{II.1}$$

A modelagem de classificação baseia-se na indução de uma função de decisão $f : \mathbb{R}^d \to \mathcal{Y}$, que busca minimizar o erro de classificação nas amostras de teste inéditas. Essa formulação orientada a faixas de quantis preserva a expressividade necessária para auxiliar o usuário final e oferece maior robustez contra flutuações e ruídos espúrios da infraestrutura [Hastie et al., 2009].

II.3.2 Modelos Lineares de Regressão e Classificação

No `AranduFlow` são avaliadas duas abordagens lineares clássicas da literatura: a Regressão Linear e a Regressão Logística.

Regressão Linear (LiR)

A Regressão Linear, do inglês *Linear Regression* (LiR), modela a variável alvo contínua (tempo de execução t) como uma combinação linear ponderada das características de entrada $\mathbf{x} \in \mathbb{R}^d$ [James et al., 2023; Hastie et al., 2009], conforme Equação II.2, onde $\mathbf{w} = [w_1, \ldots, w_d]^T \in \mathbb{R}^d$ denota o vetor de coeficientes angulares (pesos) e $b \in \mathbb{R}$ representa o viés (*bias*).

$$\hat{t}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{j=1}^{d} w_j x_j + b \tag{II.2}$$

O ajuste dos parâmetros no conjunto de treinamento contendo N amostras é realizado via Mínimos Quadrados Ordinários (do inglês, *Ordinary Least Squares* (OLS)) que minimiza a soma dos erros quadráticos médios, conforme Equação II.3.

$$J(\mathbf{w}, b) = \frac{1}{2N}\sum_{i=1}^{N}\left(t_i - \hat{t}(\mathbf{x}_i)\right)^2 = \frac{1}{2N}\sum_{i=1}^{N}\left(t_i - (\mathbf{w}^T\mathbf{x}_i + b)\right)^2 \tag{II.3}$$

Para adequar a Regressão Linear ao problema de classificação por quantis avaliado, as previsões contínuas resultantes $\hat{t}(\mathbf{x})$ são discretizadas *a posteriori* nos mesmos limiares de quantis calculados sobre o conjunto de treino.

Regressão Logística

Diferente da regressão linear, a Regressão Logística (do inglês, *Logistic Regression* (LoR)) é uma técnica linear adaptada nativamente à tarefa de classificação. No cenário multiclasse com C classes de quantis, emprega-se a extensão multinomial da regressão logística (também conhecida como regressão Softmax) [Bishop, 2006]. A probabilidade de uma instância $\mathbf{x}$ pertencer a uma classe particular $c \in \{1, \ldots, C\}$ é estimada pela função Softmax aplicada sobre os escores lineares de cada classe, conforme Equação II.4, onde $\mathbf{w}_k$ e b_k representam os pesos e o viés específicos da classe k.

$$P(y = c \mid \mathbf{x}) = \frac{e^{\mathbf{w}_c^T \mathbf{x} + b_c}}{\sum_{k=1}^{C} e^{\mathbf{w}_k^T \mathbf{x} + b_k}} \tag{II.4}$$

O modelo é treinado minimizando a função de entropia cruzada multinomial (do inglês, *categorical cross-entropy loss*) via métodos baseados em gradiente, conforme apresentado na Equação II.5, onde y_{ic} é uma variável indicadora binária que assume valor 1 se a classe real da amostra i for igual a c, e 0 caso contrário.

$$L(\mathbf{W}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \ln P(y = c \mid \mathbf{x}_i) \tag{II.5}$$

II.3.3 Árvores de Decisão

O Classificador por Árvore de Decisão (do inglês, *Decision Tree Classifier* (DTC)) é um modelo preditivo não paramétrico estruturado na forma de um grafo acíclico de decisão. A árvore realiza o particionamento hierárquico e recursivo do espaço de atributos de entrada em regiões disjuntas, associando a cada nó folha uma previsão da classe majoritária [Hastie et al., 2009; Géron, 2022].

A indução da árvore baseia-se na seleção do melhor atributo e do melhor limiar de divisão (*split*) em cada nó interno, visando maximizar a homogeneidade (pureza) dos subnós gerados. Para avaliar a qualidade de uma divisão, podem ser utilizados diferentes critérios de impureza estatística, entre os quais se destacam a Impureza de Gini e a Entropia. Nos experimentos desta dissertação, utilizou-se a Impureza de Gini, correspondente à configuração padrão dos modelos avaliados.

A Impureza de Gini ($I_G(v)$) mede a probabilidade de uma instância escolhida aleatoriamente ser rotulada incorretamente caso fosse classificada de acordo com a distribuição das classes no nó v. Para C classes de quantis, essa impureza é definida pela Equação II.6, em que $p(i \mid v)$ representa a frequência relativa de instâncias pertencentes à classe i no nó v.

$$I_G(v) = 1 - \sum_{i=1}^{C} p(i \mid v)^2 \tag{II.6}$$

O valor máximo de $I_G(v)$ é $1 - 1/C$, indicando heterogeneidade máxima, enquanto o valor mínimo

é 0, correspondente à pureza absoluta do nó, quando ele contém amostras de uma única classe.

Outro critério consolidado é a entropia de Shannon ($H(v)$), baseada na teoria da informação, que quantifica a desordem ou incerteza contida no nó v. Sua formulação é apresentada na Equação II.7. Por convenção matemática, considera-se que $0 \log_2 0 = 0$.

$$H(v) = -\sum_{i=1}^{C} p(i \mid v) \log_2 p(i \mid v) \tag{II.7}$$

Em cada etapa da indução recursiva, avaliam-se todas as possíveis divisões compostas por um atributo X_j e um ponto de corte s, gerando subnós filhos à esquerda (v_L) e à direita (v_R). Seleciona-se o par (X_j, s) que resulta na maior redução de impureza, conhecida como Ganho de Informação (ΔI), definido pela Equação II.8.

$$\Delta I(v, X_j, s) = I(v) - \left(\frac{N_{v_L}}{N_v} I(v_L) + \frac{N_{v_R}}{N_v} I(v_R) \right) \tag{II.8}$$

onde N_v, N_{v_L} e N_{v_R} representam o número total de amostras no nó pai, no subnó esquerdo e no subnó direito, respectivamente, e $I(\cdot)$ representa a métrica de impureza escolhida (Gini ou Entropia). A recursão é encerrada quando critérios de parada (como profundidade máxima da árvore ou número mínimo de amostras por divisão) são atingidos. As árvores de decisão destacam-se pela alta interpretabilidade das regras induzidas e pela capacidade inerente de capturar relações não lineares complexas sem a exigência de pré-processamento de escala de variáveis numéricas.

II.3.4 Florestas Aleatórias

Apesar de sua interpretabilidade, árvores de decisão isoladas sofrem de alta variância, sendo altamente suscetíveis a pequenas perturbações nos dados de treinamento, o que pode levar ao sobreajuste (*overfitting*). Para mitigar essa limitação, os métodos de comitê (*ensemble learning*) combinam as previsões de múltiplos modelos individuais. O principal modelo baseado em comitê avaliado neste trabalho é a Floresta Aleatória, do inglês *Random Forest Classifier* (RFC) [Hastie et al., 2009; James et al., 2023].

A Floresta Aleatória baseia-se na combinação de duas técnicas fundamentais. O primeiro mecanismo é o *Bootstrap Aggregating*. Para treinar um comitê composto por B árvores independentes, geram-se B subconjuntos de treinamento distintos por meio de amostragem com reposição a partir do conjunto de treino original. Cada árvore $h_b(\mathbf{x})$ é ajustada sobre o seu respectivo subconjunto de *bootstrap* D_b. A previsão final do comitê de classificação é obtida por meio de votação majoritária entre todas as árvores individuais, conforme definido na Equação II.9.

$$\hat{y}(\mathbf{x}) = \text{moda}\left\{h_b(\mathbf{x})\right\}_{b=1}^{B} \tag{II.9}$$

O processo de *bagging* tende a reduzir a variância global do modelo, uma vez que a média ou votação sobre múltiplos classificadores atenua as flutuações e os erros individuais, frequentemente preservando um nível de viés semelhante ao dos modelos-base [James et al., 2023; Hastie et al., 2009]. O segundo mecanismo é o subespaço aleatório de atributos (*Feature Subspace Sampling*). Para descorrelacionar as árvores geradas, garantindo que não utilizem sempre os mesmos atributos dominantes, a Floresta Aleatória introduz uma perturbação adicional na fase de indução. Em cada nó de cada árvore, a seleção do melhor ponto de divisão limita-se a avaliar um subconjunto aleatório composto por r atributos selecionados a partir das p variáveis originais. Tipicamente, adota-se $r \approx \sqrt{p}$ para problemas de classificação.

Além desses mecanismos, o uso de amostragem *bootstrap* permite estimar o erro *Out-of-Bag* (OOB), que aproxima o erro de generalização do modelo utilizando amostras não selecionadas nas etapas de *bootstrap* de cada árvore. Essa estimativa fornece uma aproximação do erro de generalização do modelo dispensando validações cruzadas custosas durante o ajuste dos hiperparâmetros, embora não substitua protocolos de avaliação em experimentos mais rigorosos [James et al., 2023; Hastie et al., 2009].

Adicionalmente, a Floresta Aleatória permite extrair uma estimativa da relevância global das variáveis preditivas por meio da importância por decréscimo de impureza, como por exemplo *Mean Decrease in Impurity* (MDI) e Importância por Gini. Para um atributo X_j, a importância é calculada como a soma ponderada de todos os decréscimos de impureza gerados por divisões sobre X_j ao longo de todas as B árvores da floresta, conforme definido na Equação II.10.

$$\text{Importância MDI}(X_j) = \frac{1}{B}\sum_{b=1}^{B}\sum_{v \in T_b:\text{split}(v)=X_j} \frac{N_v}{N_{total}}\Delta I(v) \tag{II.10}$$

onde a soma interna estende-se sobre todos os nós internos v da árvore b que utilizam X_j para realizar o *split*, e N_v/N_{total} representa a proporção de amostras do nó v em relação ao total do conjunto de dados [James et al., 2023; Hastie et al., 2009].

Embora útil para interpretação exploratória, essa métrica deve ser analisada com cautela, pois pode favorecer atributos contínuos ou variáveis categóricas com maior número de pontos de divisão possíveis. Nesta dissertação, a importância por Gini é utilizada como evidência complementar para discutir a relevância relativa dos metadados de domínio e de infraestrutura, e não como prova causal isolada do impacto de cada atributo.

II.3.5 Métricas de Avaliação de Desempenho

Para avaliar quantitativamente a qualidade preditiva dos modelos supervisionados em diferentes granularidades de classes de quantis, calculam-se métricas de desempenho derivadas da matriz de

confusão. A matriz de confusão cruza as informações reais de classe de teste contra as classes previstas pelo modelo, mapeando contagens de Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN).

A acurácia quantifica a proporção de previsões corretas (tanto positivas quanto negativas) em relação ao total de instâncias avaliadas [Powers, 2020; James et al., 2023], conforme Equação II.11.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{II.11}$$

Embora seja uma métrica intuitiva e direta, a acurácia global pode ser enganosa sob desbalanceamento de classes, pois mascara erros em classes minoritárias ao ser dominada pela classe com maior suporte. Para contornar essa limitação e fornecer uma avaliação detalhada, utilizam-se a Precisão e a Revocação (*Recall*). A Precisão (Equação II.12) mede a proporção de instâncias que realmente pertencem a uma classe em relação ao total de instâncias previstas como tal. Representa a confiabilidade da previsão positiva atribuída pelo modelo.

$$Precisão = \frac{VP}{VP + FP} \tag{II.12}$$

A Revocação (Equação II.13) mede a proporção de instâncias corretamente identificadas pelo modelo em relação ao total de instâncias reais daquela classe. Representa a sensibilidade ou capacidade de recuperação do modelo.

$$Revocacão = \frac{VP}{VP + FN} \tag{II.13}$$

O equilíbrio entre Precisão e Revocação para uma classe individual é dado pela média harmônica entre ambas, denominada escore F1 (ou F_1-escore) [Powers, 2020; James et al., 2023], conforme Equação II.14.

$$F_1 = 2 \times \frac{Precisão \times Revocacão}{Precisão + Revocacão} \tag{II.14}$$

Em cenários multiclasse como o avaliado nesta dissertação, é necessário agregar os escores F1 individuais das C classes de quantis para obter uma métrica unificada de desempenho global. Existem duas abordagens principais de agregação: a ponderada (*F1-Weighted*) e a macro (F1-Macro) [Powers, 2020].

Esta dissertação adota sistematicamente o escore F1-Macro como a métrica primária de agregação. O F1-Macro calcula a média aritmética simples dos escores F1 individuais das classes, atribuindo igual importância preditiva a todas as faixas temporais, independentemente de seu suporte, conforme a Equação II.15, onde $F_{1,c}$ representa o escore F1 calculado especificamente para a classe

$c \in \{1, \ldots, C\}$.

$$F_{1,\text{Macro}} = \frac{1}{C} \sum_{c=1}^{C} F_{1,c} \tag{II.15}$$

A escolha do escore F1-Macro justifica-se pela importância de avaliar o desempenho do preditor em todas as classes temporais. Em ambientes de HPC de produção, identificar adequadamente as execuções pertencentes às faixas de maior duração é relevante para apoiar o planejamento do usuário e reduzir o risco de subestimação do tempo necessário. O uso do F1-Macro reduz o risco de que o desempenho nessas classes seja mascarado pelo desempenho nas classes mais frequentes, fornecendo uma métrica alinhada ao objetivo de avaliar todas as faixas temporais com igual importância.

II.4 Aprendizado por Transferência e Calibração de Modelos

Modelos preditivos baseados em aprendizado de máquina frequentemente assumem que os dados de treinamento e de teste seguem a mesma distribuição de probabilidade. Contudo, em ambientes heterogêneos de computação de alto desempenho, migrar um preditor para uma nova infraestrutura pode violar essa premissa. Esse fenômeno é conhecido como mudança de domínio ou desvio de covariáveis (*covariate shift*) [Shimodaira, 2000].

Para contornar o desvio de distribuição sem a necessidade de reexecutar todo o processo de modelagem a partir do zero em um novo ambiente computacional, utilizam-se técnicas de Aprendizado por Transferência (*Transfer Learning*) [Pan and Yang, 2009]. O objetivo principal é aproveitar o conhecimento adquirido em um domínio de origem para otimizar o desempenho do preditor em um domínio de destino relacionado com dados limitados.

Como fundamento para a extensão experimental avaliada no Capítulo V, apresentam-se duas estratégias de adaptação do preditor a novos ambientes computacionais utilizando pequenas amostras de execuções locais. A primeira consiste na calibração de probabilidades (*Platt Scaling*). Nessa abordagem, classificadores de aprendizado de máquina geram escores de confiança que nem sempre correspondem à probabilidade real da previsão correta. O método de Calibração de Platt [Platt, 1999] ajusta um modelo logístico sobre as previsões do modelo base para converter os escores originais em probabilidades calibradas. Em problemas multiclasse, a calibração sigmoide ajusta o mapeamento entre os escores produzidos pelo classificador e as probabilidades estimadas para cada classe, sem modificar diretamente a estrutura interna do modelo base.

Já o segundo método, chamado de ajuste fino ponderado (do inglês, *Weighted Fine-Tuning*), baseia-se na técnica de ponderação de amostras (*instance re-weighting*), sob desvio de covariáveis [Shimodaira, 2000]. Em vez de treinar um modelo novo puramente no reduzido conjunto de calibração local, o que causaria sobreajuste, treina-se o preditor combinando as instâncias de origem e destino, aplicando pesos de amostragem diferenciados. O peso de treino global é mantido baixo

(λ_{origem}), enquanto as instâncias de sondagem local do ambiente computacional de destino recebem um peso significativamente maior ($\lambda_{destino}$). Essa ponderação força o otimizador do modelo a priorizar a minimização dos erros sobre as características locais do novo ambiente computacional, preservando as correlações globais aprendidas.

II.5 Dados Históricos e Reprodutibilidade

A construção de modelos preditivos depende da disponibilidade de dados históricos, incluindo registros de execução capazes de descrever as condições e os resultados de execuções anteriores. No contexto de *workflows* científicos, a captura sistemática desses registros favorece análises comparativas e reprodutibilidade, além de viabilizar estratégias adaptativas baseadas em experiências passadas [de Oliveira et al., 2012; Freire and Silva, 2012].

Esses registros podem incluir informações sobre parâmetros de entrada, configurações de execução, recursos computacionais utilizados, métricas de desempenho e *logs* das tarefas que compõem o *workflow*. Tais dados são frequentemente coletados por mecanismos de proveniência, que registram tanto a estrutura dessas aplicações quanto as informações geradas durante sua execução.

A disponibilidade desses dados históricos possibilita a construção de modelos de aprendizado de máquina capazes de estimar métricas relevantes, como tempo de execução ou consumo de recursos, a partir de padrões observados em execuções anteriores. Além disso, a preservação sistemática dessas informações contribui para a reprodutibilidade científica, permitindo que experimentos sejam replicados, analisados e comparados em diferentes contextos computacionais.

Capítulo III Trabalhos relacionados

Para mapear de forma sistemática o estado da arte e identificar as abordagens mais relevantes voltadas à predição de desempenho e ao suporte à execução de *workflows* científicos, esta dissertação adotou um protocolo estruturado de revisão bibliográfica. A busca primária foi realizada na base de dados Scopus e considerou os campos de título, resumo e palavras-chave das publicações.

A escolha da Scopus como base para a busca primária foi motivada por sua cobertura multidisciplinar e pela indexação de periódicos e anais de conferências revisados por pares, característica especialmente relevante para a área de Computação, na qual parte significativa dos resultados científicos é publicada em eventos.

A *string* de busca foi construída integrando termos de quatro dimensões fundamentais do problema de pesquisa: (i) o objeto de estudo (*workflows* ou *dataflows* científicos); (ii) a métrica de desempenho alvo (tempo de execução); (iii) a tarefa preditiva (predição ou estimativa); e (iv) o ambiente computacional de execução (*High Performance Computing* ou *clusters*). A formulação lógica da consulta está apresentada a seguir:

```
("scientific workflow*"OR "scientific dataflow*") AND ("execution time"OR
runtime) AND (predict* OR estimat*) AND ("high performance computing"OR HPC
OR cluster*)
```

A busca foi executada em 09 de março de 2026, resultando em um conjunto inicial de 41 artigos. Para orientar a seleção das publicações relevantes dentre os resultados, foram definidos critérios de inclusão e exclusão. Como critérios de inclusão, consideraram-se: CI1, artigos científicos completos publicados em periódicos ou anais de conferências revisados por pares; CI2, trabalhos que propõem, avaliam ou revisam modelos de estimativa de tempo de execução, consumo de CPU, memória ou E/S para tarefas ou *workflows* científicos completos; e CI3, abordagens direcionadas a ambientes distribuídos de alto desempenho, como *clusters* de computação, infraestruturas HPC ou infraestruturas de nuvem computacional destinadas a processamento paralelo.

Como critérios de exclusão, foram considerados: CE1, trabalhos focados na predição de tarefas de sistemas operacionais de uso geral ou consultas isoladas de bancos de dados relacionais que não estejam modeladas como grafos de fluxo de trabalho científicos, salvo trabalhos de proveniência metodologicamente próximos, como o APOENA, mantidos para fins de contextualização arquitetural;

CE2, textos curtos, resumos estendidos, relatórios técnicos, tutoriais e patentes; e CE3, estudos que utilizam estimativas de desempenho meramente como caso de uso ilustrativo secundário, sem descrever detalhadamente os métodos preditivos ou sem realizar validação experimental sistemática do erro de predição.

O processo de busca, triagem e seleção dos estudos é apresentado na Figura III.1. Na triagem inicial, realizou-se a leitura dos títulos e resumos dos 41 artigos retornados. A aplicação dos critérios de inclusão e exclusão resultou na eliminação de 25 artigos, restando 16 candidatos para leitura na íntegra. Em seguida, os 16 artigos foram integralmente examinados para confirmar seu alinhamento com o escopo da pesquisa. Durante essa etapa, aplicou-se a técnica de *snowballing*, que permitiu identificar cinco publicações adicionais relevantes não recuperadas pela busca inicial, totalizando 21 estudos alinhados ao tema. Para a análise comparativa aprofundada, foram selecionados os dez trabalhos com maior afinidade temática e metodológica com a proposta desta dissertação. Essa afinidade foi determinada pela proximidade simultânea em relação às principais dimensões investigadas: predição de tempo de execução ou de recursos computacionais, utilização de *workflows* científicos, execução em ambientes distribuídos ou de HPC, emprego de dados históricos e metadados, aplicação de técnicas de aprendizado de máquina e integração com mecanismos de submissão ou suporte à decisão. Os demais estudos contribuíram para a contextualização do campo, mas apresentaram aderência parcial a essas dimensões ou abordagens semelhantes às já contempladas no conjunto selecionado. Os dez trabalhos são analisados detalhadamente nas subseções seguintes e sintetizados no mapeamento comparativo.

III.1 Análise dos Trabalhos Relacionados

Esta seção apresenta uma análise detalhada e comparativa das iniciativas identificadas na literatura, organizando-as de acordo com a sua abordagem metodológica e escopo tecnológico. Inicialmente, são analisados os trabalhos centrados no uso de técnicas de aprendizado de máquina supervisionado para a predição de recursos e tempo. Em seguida, examinam-se as estratégias que combinam estimativas a algoritmos de escalonamento ativos baseados em heurísticas ou aprendizado por reforço. Por fim, são discutidos os portais científicos e sistemas de gerenciamento desenvolvidos para suportar a execução de *workflows* em infraestruturas distribuídas.

As abordagens baseadas em Aprendizado de Máquina (AM) têm ganhado destaque na literatura recente pela sua capacidade de capturar a complexidade não linear associada ao tempo de execução de tarefas e *workflows* em sistemas distribuídos, substituindo ou complementando modelos analíticos rígidos.

O trabalho de Rosa et al. [2021] propôs o *Configurable Resource Cost Prediction Service* (CRCPs), um serviço configurável voltado a prever o tempo de execução e o custo monetário de *workflows*

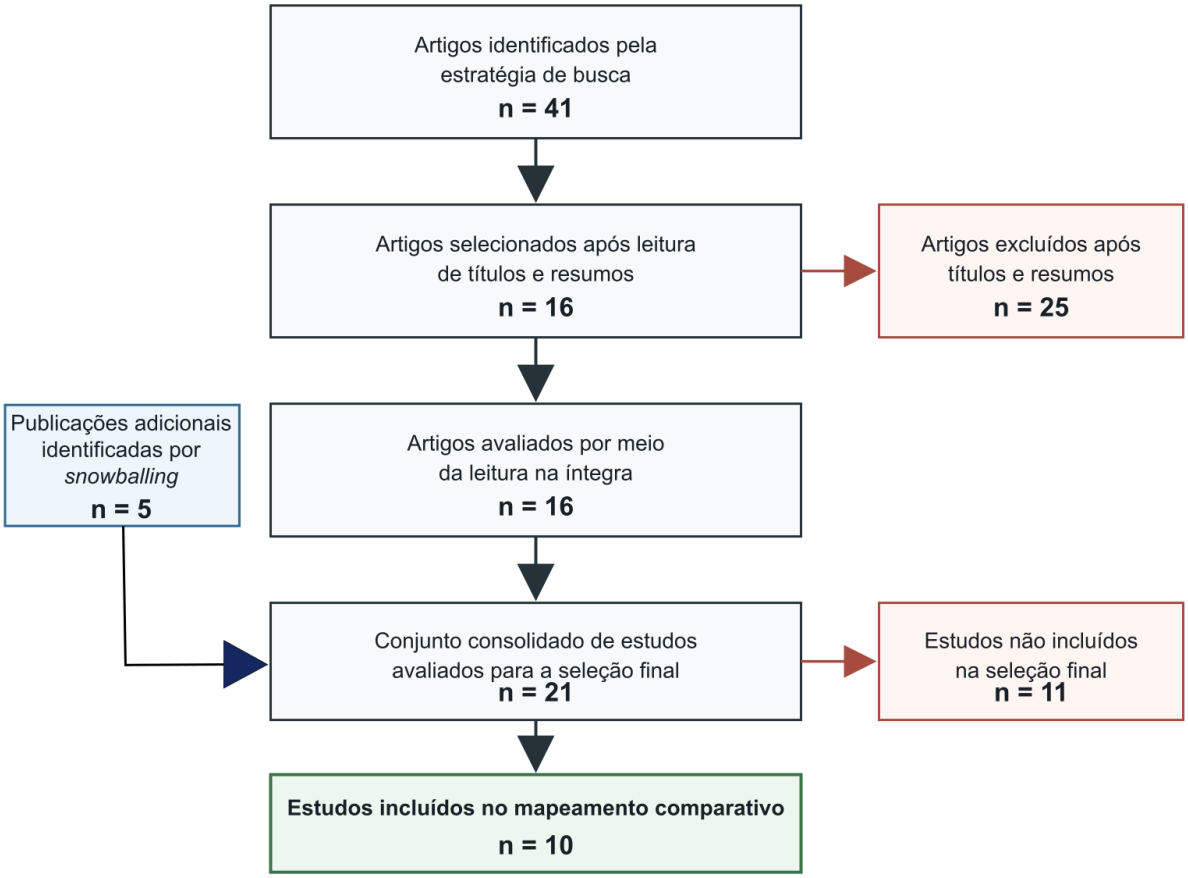


Figura III.1: Fluxograma do processo de busca, triagem e seleção dos estudos.

científicos em nuvens federadas. Para isso, o CRCPs utiliza modelos de Regressão Linear Múltipla. O principal diferencial da proposta é a integração das predições com a meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para explorar o espaço de busca e sugerir alocações de recursos balanceadas sob a perspectiva de tempo de processamento e custo financeiro. Embora robusto para cenários de nuvem federada pública, o uso de regressão contínua em ambientes altamente dinâmicos e compartilhados de HPC pode introduzir erros substanciais nas previsões pontuais devido a flutuações rápidas na concorrência de recursos físicos.

Com o objetivo de estimar o tempo de execução de tarefas de *workflows* em infraestruturas heterogêneas locais, Bader et al. [2024] introduziram o método `Lotaru`. Esta abordagem baseia-se na execução sistemática de *microbenchmarks* de CPU e de E/S na infraestrutura de destino e no emprego de modelos de regressão Bayesiana. A principal inovação do `Lotaru` consiste em fornecer estimativas acompanhadas de uma medida de incerteza explícita, auxiliando o escalonador a lidar com a volatilidade. No entanto, o método limita-se à predição de tarefas individuais e exige a execução periódica de *microbenchmarks* intrusivos que consomem capacidade útil dos nós. Adicionalmente, o sistema não é acoplado a uma interface de submissão transparente capaz de capturar e catalogar dados históricos estruturados de proveniência de forma contínua e automatizada.

Voltado a infraestruturas de grande escala e integração de ambientes, Yildirim et al. [2026] desenvolveram modelos baseados em Redes Neurais Artificiais para predizer o tempo de execução e o perfil de utilização de recursos de *batch jobs* executados em plataformas híbridas de HPC e nuvem (AWS). A predição é utilizada para otimizar o escalonamento e suportar decisões de migração de carga entre a infraestrutura local e pública. As redes neurais alcançaram boa acurácia (medida por MAE e R^2), porém, a proposta baseia-se em regressão contínua sob atributos genéricos fornecidos pelo escalonador (como a memória solicitada e o tempo limite definido pelo usuário), desconsiderando metadados do domínio intrínseco da aplicação.

No contexto de suporte específico a supercomputadores compartilhados de produção, Porto et al. [2026] propuseram um protocolo para predição confiável de recursos (CPU, memória e tempo de execução) baseado em aprendizado supervisionado. Os autores compararam múltiplos algoritmos e propuseram uma lógica conservadora de decisão estatística para dimensionamento seguro, mitigando o risco de estouro de limites (*walltime*) impostos pelos escalonadores `Slurm` e `PBS`. No entanto, as predições apoiam-se predominantemente em variáveis estáticas de infraestrutura e no histórico global do usuário, sem a modelagem de características de dados de entrada ou parâmetros específicos dos *workflows* científicos.

Por fim, no cenário brasileiro, Sobrinho and Paumgartten [2021] descreveram um serviço de predição de recursos voltado a *workflows* de bioinformática em nuvens federadas. O trabalho empregou algoritmos de classificação como RFC e K-Vizinhos Mais Próximos (do inglês, *K-Nearest*

Neighbors (KNN)) para categorizar o consumo de recursos computacionais e estimar tempos de execução baseando-se em parâmetros de entrada do domínio genômico. Embora represente uma iniciativa focada na integração de metadados específicos de aplicação, a solução foi especificamente projetada para o ambiente de nuvem federada e avaliada sob *workflows* puramente intensivos em CPU, carecendo de uma validação sob infraestruturas de HPC de produção compartilhadas e sob fluxos com alta intensidade de E/S.

Paralelamente aos preditores baseados puramente em aprendizado supervisionado, diversos trabalhos focam no desenvolvimento de estratégias de escalonamento ativo, nas quais estimativas de recursos e tempo de execução guiam algoritmos heurísticos ou agentes inteligentes no particionamento e na alocação de tarefas.

Alvarenga et al. [2025] propuseram uma abordagem baseada em uma heurística híbrida denominada *layered-bucket* para otimizar a estimativa e o dimensionamento de recursos de *workflows* científicos em ambientes de HPC. O método visa reduzir a fragmentação e o tempo de fila ao agrupar requisições em faixas discretas de capacidade física. Embora demonstre eficácia para mitigar os gargalos de agendamento em *clusters* de produção sob alta volatilidade de concorrência, o trabalho foca no algoritmo de escalonamento e depende de previsões simplificadas baseadas no histórico geral do *cluster*, sem induzir classificadores refinados que combinam metadados complexos do fluxo científico.

Abordando o problema sob o paradigma de controle inteligente, Nascimento et al. [2019] desenvolveram uma estratégia de escalonamento para *workflows* paralelos em nuvem baseada em Aprendizado por Reforço (do inglês, *Reinforcement Learning* (RL)). O agente de RL aprende de forma ativa a priorização de tarefas e a seleção de instâncias virtuais com base nas recompensas de tempo total de execução (*makespan*) e custo monetário observadas a cada passo. Embora a abordagem evite a necessidade de um modelo preditivo explicitamente treinado para estimar o tempo de execução, as técnicas de RL enfrentam desafios de convergência em espaços de estados complexos e longas etapas de treinamento na inicialização do sistema, o que limita sua adoção imediata em *gateways* científicos de produção em ambientes de HPC.

No contexto de dimensionamento proativo de recursos, de C. Coutinho et al. [2016] propuseram uma estratégia para *workflows* genômicos paralelos executados em nuvem. A abordagem baseia-se na criação de perfis históricos de execução combinados a modelos analíticos de desempenho para estimar o número ideal de máquinas virtuais necessárias para satisfazer prazos de execução (*deadlines*) definidos pelo usuário. A proposta visa equilibrar o paralelismo e os custos computacionais associados. Contudo, assim como no caso do CRCPs, a formulação está voltada estritamente ao provisionamento sob demanda de nuvem elástica, o que difere das restrições operacionais de infraestruturas HPC locais baseadas em filas estáticas de *batch*.

A fim de abstrair a complexidade operacional de ambientes distribuídos heterogêneos, diversos sistemas de gerenciamento e portais científicos (*science gateways*) foram desenvolvidos na literatura para atuar como intermediários entre os pesquisadores e a infraestrutura física de computação.

Sistemas clássicos como `Pegasus` [Deelman et al., 2015], `Nextflow` [Di Tommaso et al., 2017] e `Snakemake` [Köster and Rahmann, 2012] atuam na orquestração de tarefas e no controle do fluxo de dados em *clusters* HPC e na nuvem. Essas plataformas oferecem robustez contra falhas de rede e facilitam o agendamento de tarefas. No entanto, embora forneçam estatísticas pós-execução, elas não dispõem de componentes nativos para predição prévia de tempo de execução com base no aprendizado dinâmico de dados de proveniência anteriores, exigindo que soluções externas realizem essa atividade.

No contexto brasileiro, destaca-se o `BioinfoPortal` [Ocaña et al., 2020], que integra aplicações de bioinformática a infraestruturas nacionais de HPC, oferecendo suporte à submissão, monitoramento e análise de desempenho de aplicações científicas. A proposta explora o uso de dados de proveniência para avaliar eficiência, escalabilidade e estratégias de alocação de recursos em aplicações paralelas. Embora compartilhe similaridades arquiteturais com a abordagem proposta nesta dissertação, como a existência de uma interface *web* e o armazenamento de informações de execução, o uso dos dados coletados no `BioinfoPortal` concentra-se predominantemente em análises *a posteriori* e na otimização do uso de recursos, não enfatizando a predição prévia do tempo de execução como instrumento direto de suporte à decisão antes da submissão do *workflow*.

Com foco na modelagem baseada em proveniência, o *middleware* `APOENA` [Ribeiro et al., 2024] foi concebido para coletar e estruturar de forma sistemática metadados de execução a fim de treinar modelos supervisionados para predição de recursos. Embora utilize uma arquitetura orientada a proveniência similar à do `AranduFlow`, o `APOENA` foi avaliado apenas sob consultas *Structured Query Language* (SQL) analíticas em nuvem e não sob grafos complexos de dependências de arquivos característicos de *workflows* científicos.

A Tabela III.1 apresenta uma síntese comparativa das principais abordagens discutidas neste capítulo. A comparação destaca diferenças quanto ao ambiente computacional considerado, às técnicas utilizadas para predição de desempenho e ao nível de integração com mecanismos de submissão de *workflows* científicos.

III.2 Discussão Comparativa e Lacunas da Literatura

Como pode ser observado na Tabela III.1, embora a literatura apresente contribuições relevantes para a estimativa de tempo e alocação de recursos em sistemas distribuídos, existem lacunas que limitam a usabilidade prática dessas abordagens sob a perspectiva do usuário de *gateways* científicos em HPC.

Tabela III.1: Comparação entre abordagens de predição e suporte à execução de *workflows* científicos

Trabalho	Ambiente alvo	Técnica utilizada	Tipo de predição	Int. submissão
CRCPs [Rosa et al., 2021]	Nuvem federada	Regressão linear + GRASP	Tempo e custo de execução	Não
`Lotaru` [Bader et al., 2024]	*Clusters* heterogêneos	*Microbenchmarks* + reg. Bayesiana	Tempo de execução	Não
Yildirim *et al.* [Yildirim et al., 2026]	Nuvem e HPC	Redes Neurais Artificiais	Tempo e uso de recursos	Não
Porto *et al.* [Porto et al., 2026]	HPC compartilhado	Aprendizado supervisionado	CPU, memória e tempo	Não
Sobrinho & Paumgartten [Sobrinho and Paumgartten, 2021]	Nuvem federada	*Random Forest* e KNN	Tempo e recursos	Não
Alvarenga *et al.* [Alvarenga et al., 2025]	HPC compartilhado	Heurística *layered-bucket*	Estimativa de recursos	Não
Nascimento *et al.* [Nascimento et al., 2019]	Nuvem computacional	Aprendizado por Reforço	*Makespan* e custo (decisão)	Não
Coutinho *et al.* [de C. Coutinho et al., 2016]	Nuvem computacional	Perfis hist. + modelos analíticos	VM ideal para *deadlines*	Não
APOENA [Ribeiro et al., 2024]	Computação em nuvem	Aprendizado de máquina + proveniência	Tempo de consultas SQL	Parcial
`BioinfoPortal` [Ocaña et al., 2020]	HPC	Proveniência e análise pós-execução	Análise *a posteriori*	Sim
`AranduFlow`	HPC heterogêneo	Aprendizado de máquina + proveniência	Faixas de tempo (quantis)	Sim

A primeira dessas lacunas reside na robustez contra ruído e variabilidade em HPC, uma vez que a maioria dos trabalhos focados na predição de tempo (*e.g.*, CRCPs, `Lotaru`, Yildirim *et al.* e Porto *et al.*) formula o problema como regressão contínua para estimar o tempo pontual exato em segundos. No entanto, em *clusters* HPC de produção, flutuações na carga física introduzem ruído imprevisível. O `AranduFlow` busca contornar essa limitação ao reformular a tarefa como classificação multiclasse por meio de intervalos de tempo definidos por quantis, o que visa amortecer desvios marginais e fornecer estimativas úteis na prática para apoiar o planejamento do usuário.

A segunda lacuna refere-se à integração de metadados de domínio e infraestrutura: a maior parte dos modelos preditivos fundamenta-se puramente em métricas de *hardware* solicitadas ou no tamanho físico agregado dos dados de entrada. Contudo, variáveis específicas de domínio podem exercer impacto sobre o esforço computacional e o *makespan*. O `AranduFlow` avança ao integrar esses metadados de domínio aos atributos de infraestrutura em uma arquitetura extensível a diferentes *workflows*, embora os modelos estatísticos sejam especializados para cada *workflow*.

Por fim, a terceira lacuna consiste na falta de integração com mecanismos de submissão. Portais como o `BioinfoPortal` coletam e armazenam metadados ricos para análise *a posteriori*, mas não utilizam esses dados ativamente para fornecer suporte preditivo antes do disparo do *workflow*. Por outro lado, modelos preditivos avançados (*e.g.*, redes neurais de Yildirim *et al.* e regressão Bayesiana do `Lotaru`) operam de forma isolada, carecendo de um ecossistema transparente que os acople a interfaces *web* amigáveis e escalonadores subjacentes. A arquitetura modular do `AranduFlow` contribui para reduzir essa lacuna ao integrar a predição e a submissão em uma mesma ferramenta.

Capítulo IV Arquitetura e Funcionamento do *Gateway* `AranduFlow`

Este capítulo apresenta a arquitetura, a implementação e o funcionamento do `AranduFlow`, o principal artefato desenvolvido nesta dissertação. O sistema foi concebido como um *gateway* científico que integra, em um fluxo único, a configuração de *workflows*, a coleta e a validação de parâmetros, a estimativa prévia de faixas de tempo de execução, a submissão a infraestruturas de HPC e o armazenamento da configuração, da previsão e do tempo posteriormente observado.

A contribuição arquitetural do `AranduFlow` não se restringe à avaliação de modelos de aprendizado de máquina. A camada preditiva constitui um dos serviços internos do sistema e opera em conjunto com a interface de interação, o catálogo de *workflows*, o mecanismo de persistência, os conectores de infraestrutura e os executores responsáveis pelas aplicações científicas. Essa integração permite transformar dados históricos de execução em informações disponibilizadas ao usuário antes de uma nova submissão.

Inicialmente, são apresentados o escopo e os objetivos arquiteturais do *gateway*. Em seguida, descrevem-se a representação adotada para uma execução, a arquitetura geral do sistema, a interação do usuário e o funcionamento de cada componente. Também são detalhados o ciclo de submissão, a persistência dos dados, a comunicação com as infraestruturas, a execução das aplicações e o retorno do tempo observado.

IV.1 Escopo e Objetivos Arquiteturais

O `AranduFlow` atua como uma camada intermediária entre os usuários de aplicações científicas e as infraestruturas computacionais responsáveis por sua execução. Seu objetivo é reduzir a complexidade operacional associada à configuração e à submissão de *workflows* em ambientes de HPC, ao mesmo tempo que oferece informações preditivas para apoiar o planejamento do usuário antes do início do processamento.

Sem o apoio de um *gateway*, a preparação de uma execução pode exigir a edição manual de arquivos de configuração, a construção de comandos específicos do escalonador, a preparação dos dados no ambiente remoto e o acompanhamento individual dos arquivos de saída. Além disso, as informações produzidas após cada execução frequentemente permanecem dispersas em arquivos de configuração, registros do escalonador e diretórios de resultados, dificultando seu reaproveitamento.

O `AranduFlow` organiza a configuração e a submissão em um fluxo estruturado, mantendo a associação entre a configuração informada, a previsão apresentada, a infraestrutura selecionada e o tempo posteriormente observado.

Do ponto de vista do usuário, o fluxo inicia-se com a seleção de um *workflow* previamente cadastrado. A interface apresenta as atividades e os parâmetros correspondentes à aplicação selecionada, permitindo informar os parâmetros científicos, referenciar os arquivos, definir a quantidade de recursos e selecionar uma das infraestruturas disponíveis. A configuração pode ser preenchida por meio dos formulários da interface ou fornecida em um arquivo no formato *JavaScript Object Notation* (JSON).

Após o preenchimento, o sistema valida os valores informados e produz uma representação padronizada da execução. Essa representação reúne a identificação do *workflow*, as referências e os metadados dos arquivos de entrada, os parâmetros científicos, os recursos solicitados e as características da infraestrutura selecionada. Com base nessas informações, o serviço de predição carrega o modelo associado ao *workflow* e estima a faixa correspondente ao seu tempo de execução.

A faixa prevista é apresentada ao usuário antes da submissão. Nesse momento, o usuário pode retornar à configuração e modificar os parâmetros, descartar a previsão ou confirmar a execução. A previsão possui caráter informativo: o `AranduFlow` não substitui as políticas administrativas do escalonador e não altera automaticamente os recursos solicitados. A decisão final permanece com o usuário, enquanto a alocação e o gerenciamento da fila permanecem sob responsabilidade da infraestrutura de HPC.

Quando a submissão é confirmada, o *gateway* atribui um identificador único à execução, persiste a configuração e a previsão, transfere o documento JSON para o ambiente computacional selecionado e aciona o executor correspondente ao *workflow*. O conector de infraestrutura realiza a comunicação com o nó de submissão e encaminha a solicitação ao escalonador. O identificador é mantido durante o processamento para que o tempo posteriormente retornado possa ser associado ao registro correto.

Ao término bem-sucedido do processamento, o tempo real de execução é incorporado ao histórico. Esse mecanismo fecha o ciclo entre configuração, predição, execução e observação. As medições acumuladas podem ser utilizadas posteriormente para treinamento, validação e atualização dos modelos preditivos, sem que o processo de retreinamento interfira diretamente na execução em andamento.

A extensibilidade do `AranduFlow` está na separação entre os mecanismos compartilhados pelo *gateway* e os elementos específicos de cada aplicação. A interface, a persistência, a estrutura da configuração, os conectores de infraestrutura e o fluxo de submissão são reutilizados pelos diferentes *workflows*. Em contrapartida, cada aplicação possui seu próprio conjunto de atividades, parâmetros

de domínio, regras de validação, atributos preditivos, executor, histórico e modelo treinado. Assim, a integração de um novo *workflow* não exige a reconstrução do *gateway*, mas requer o cadastro de seus metadados e a implementação dos elementos específicos necessários à sua execução e predição.

IV.2 Representação de uma Execução e Entrada do Modelo Preditivo

O `AranduFlow` representa uma submissão a partir da identificação do *workflow* selecionado e do conjunto de informações disponíveis antes de sua execução. Essa representação é expressa pelo par ordenado, representado na Equação IV.1

$$\mathcal{K} = (\mathcal{W}, \mathbf{x}), \tag{IV.1}$$

em que $\mathcal{W}$ representa conceitualmente a estrutura lógica do *workflow*, conforme definido na Seção II.1, e $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ corresponde ao vetor de atributos utilizado para estimar sua faixa de tempo de execução.

A presença de $\mathcal{W}$ nessa representação não significa que o grafo do *workflow* seja fornecido diretamente ao modelo de aprendizado de máquina. No funcionamento do `AranduFlow`, $\mathcal{W}$ identifica a aplicação selecionada, determina suas atividades e seus parâmetros de configuração e permite localizar os componentes específicos associados a ela, como o esquema de configuração, o executor, o processo de engenharia de atributos e o modelo preditivo previamente treinado. O algoritmo de predição recebe como entrada apenas o vetor $\mathbf{x}$.

O vetor de atributos é construído a partir de três conjuntos de informações coletadas antes da submissão: os parâmetros de domínio e de configuração da aplicação, as características dos dados de entrada e as informações da infraestrutura computacional selecionada. Essa construção pode ser representada pela Equação IV.2

$$\mathbf{x} = (\mathbf{d}, \mathbf{e}, \mathbf{h}), \tag{IV.2}$$

em que $\mathbf{d}$ reúne os metadados de domínio e os parâmetros científicos, $\mathbf{e}$ descreve as características dos dados de entrada, $\mathbf{h}$ contém as informações da infraestrutura e dos recursos solicitados.

Os metadados de domínio incluem decisões que alteram o comportamento interno da aplicação, como a escolha de algoritmos ou opções de processamento. As características dos dados de entrada compreendem informações disponíveis antes da execução, como quantidade de arquivos, tamanho total, número de sequências ou outras medidas de volumetria. As informações da infraestrutura incluem o ambiente selecionado, a quantidade de *threads* solicitada e as características computacionais necessárias para diferenciar os ambientes cadastrados.

Como os *workflows* possuem parâmetros e comportamentos distintos, o `AranduFlow` mantém

um modelo preditivo específico para cada aplicação. Seja $f_{\mathcal{W}}$ o modelo associado ao *workflow* $\mathcal{W}$, a predição realizada pelo sistema é definida na Equação IV.3

$$\hat{y} = f_{\mathcal{W}}(\mathbf{x}), \tag{IV.3}$$

em que $\hat{y}$ pertence ao conjunto de classes temporais $\mathcal{Y}_{\mathcal{W}} = \{Q_1, Q_2, \ldots, Q_C\}$. Cada classe Q_j está associada a um intervalo de tempo cujas fronteiras são obtidas a partir do histórico de execuções utilizado no treinamento do respectivo *workflow*. Dessa forma, uma mesma classe, como Q_1, pode representar limites temporais diferentes em aplicações distintas.

Após calcular $\hat{y}$, o Gerenciador de Predição recupera os limites temporais correspondentes à classe prevista e apresenta esse intervalo ao usuário. A classe constitui a saída interna do modelo, enquanto a faixa de tempo constitui a informação exibida pelo *gateway*. Essa distinção permite que a interface apresente uma estimativa temporal compreensível sem expor ao usuário os rótulos empregados internamente pelo classificador.

Portanto, a generalidade do `AranduFlow` não decorre de um único modelo capaz de processar diretamente qualquer grafo de *workflow*. Ela está na existência de um fluxo comum para cadastrar a aplicação, coletar seus metadados, construir o vetor de atributos, carregar o modelo correspondente, apresentar a faixa prevista, realizar a submissão e registrar o resultado observado. Os atributos e o modelo $f_{\mathcal{W}}$ permanecem específicos de cada *workflow*.

IV.3 Arquitetura Geral do `AranduFlow`

O `AranduFlow` adota uma arquitetura modular orientada ao ciclo completo de uma execução científica, abrangendo a configuração do *workflow*, a produção da estimativa temporal, a submissão à infraestrutura computacional e a incorporação do resultado observado ao histórico. Essa organização busca desacoplar a interação com o usuário dos detalhes específicos dos modelos preditivos, dos escalonadores e das aplicações científicas executadas.

A arquitetura lógica do sistema é composta por quatro módulos principais: Interface *Web*, Gerenciador de Predição e Persistência, Seletor e Conector de Infraestrutura e Executores de *Workflows*. O documento no formato JSON constitui o principal contrato de comunicação entre a interface, os serviços internos do *gateway* e os executores instalados nos ambientes computacionais.

A Figura IV.1 apresenta a organização lógica do sistema em camadas. Os módulos numerados de 1 a 3 compõem o *gateway*, enquanto os Executores de *Workflows*, identificados pelo número 4, são instalados no ambiente computacional. As setas representam a configuração realizada pelo usuário, a produção da faixa estimada, a submissão da execução e o retorno assíncrono do tempo observado.

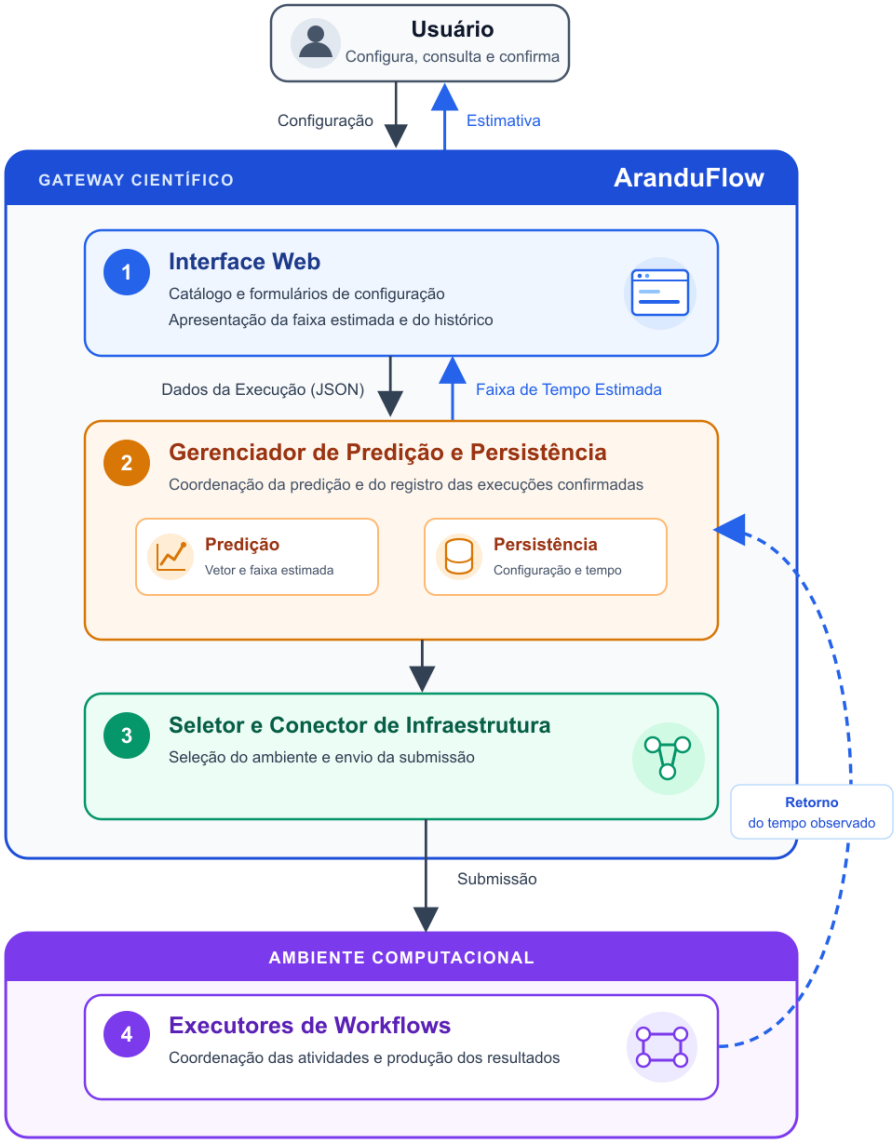


Figura IV.1: Arquitetura lógica do *gateway* `AranduFlow`.

IV.4 Interface *Web*

A Interface *Web*, identificada como módulo 1 na Figura IV.1, constitui o principal ponto de interação entre o usuário e o `AranduFlow`. Por meio dela, o usuário seleciona o *workflow* científico que deseja executar, informa os parâmetros necessários, referencia os dados de entrada, escolhe a infraestrutura computacional e solicita uma estimativa para o tempo de execução. Dessa forma, a interface abstrai parte da complexidade associada à configuração e à submissão manual de *workflows* em ambientes de processamento de alto desempenho.

A interação com a Interface *Web* compreende dois momentos distintos: o cadastro do *workflow* e a configuração de uma execução. No primeiro, define-se a estrutura reutilizável da aplicação e de seus parâmetros; no segundo, atribuem-se valores concretos a esses parâmetros e selecionam-se os dados de entrada e os recursos computacionais para uma execução específica.

IV.4.1 Cadastro do *Workflow* e Definição de seus Parâmetros

O cadastro reúne a identificação do *workflow*, suas atividades e os parâmetros científicos associados a cada atividade. Para cada parâmetro, especificam-se o nome, o tipo de dado e, quando aplicável, os valores permitidos, os limites numéricos e as condições de exibição. Essas informações constituem os metadados utilizados pela interface para apresentar dinamicamente o formulário de configuração de uma execução. A Tabela IV.1 descreve os campos envolvidos nesse processo.

Tabela IV.1: Campos de cadastro de *workflows*, atividades e parâmetros científicos no `AranduFlow`.

Grupo	**Campo**	**Descrição e preenchimento**
Workflow	Nome	Identifica o workflow no catálogo. Preenchimento obrigatório.
Workflow	Descrição	Apresenta a finalidade do processamento científico. Preenchimento obrigatório.
Workflow	Ícone	Representação visual: Ícone que é associado ao *workflow*.
Workflow	Palavras-chave	Termos de classificação separados por vírgulas. Campo opcional.
Atividade	Nome da atividade	Identifica uma etapa do workflow. Preenchimento obrigatório.
Atividade	Descrição	Explica o processamento realizado pela atividade. Campo opcional.
Atividade	Etapa opcional	Indica se a atividade é opcional. Caixa de seleção inicialmente desmarcada.
Parâmetro	Nome	Identifica o parâmetro científico da atividade. Obrigatório no cadastro.

Continua na próxima página.

Tabela IV.1 – continuação

Grupo	**Campo**	**Descrição e preenchimento**
Parâmetro	Descrição	Explica o significado do parâmetro. Campo opcional.
Parâmetro	Tipo	Categórico (padrão), inteiro, numérico real, texto ou booleano.
Regras	Valores permitidos	Opções adicionadas individualmente para parâmetros categóricos.
Regras	Valor mínimo	Limite inferior opcional para parâmetros inteiros ou numéricos reais.
Regras	Valor máximo	Limite superior opcional para parâmetros inteiros ou numéricos reais.
Condição de exibição	Parâmetro de referência	Outro parâmetro cuja resposta condiciona a exibição do parâmetro atual. Necessário ao adicionar uma condição.
Condição de exibição	Valor da condição	Valor do parâmetro de referência que permite exibir o parâmetro atual. Necessário ao adicionar uma condição.

Após a criação do *workflow*, são cadastradas suas atividades e, em cada uma delas, os respectivos parâmetros. O cadastro inicial de um parâmetro solicita nome, descrição e tipo; os valores permitidos, limites e condições são definidos no painel de detalhes. Um parâmetro sem condições permanece visível no formulário de configuração; quando existem várias condições, basta que uma delas seja atendida para que ele seja exibido. Esse cadastro descreve a aplicação, mas sua integração operacional também requer o executor e o modelo preditivo específicos, conforme o escopo arquitetural apresentado anteriormente.

IV.4.2 Configuração de uma Execução

Ao iniciar uma nova execução, o usuário seleciona um dos *workflows* cadastrados no *gateway*. A partir dessa escolha, a interface consulta as informações armazenadas sobre o *workflow* e apresenta dinamicamente o formulário correspondente. Os campos exibidos dependem das características de cada aplicação e seus parâmetros de domínio. Restrições e dependências entre parâmetros também podem ser representadas, permitindo que determinados campos sejam habilitados ou preenchidos de acordo com escolhas realizadas anteriormente.

Na configuração de uma execução, o usuário atribui valores aos parâmetros previamente definidos, sem precisar cadastrar novamente a aplicação. Por exemplo, no cadastro do NMFSt.P, o parâmetro `algorithm` é definido como categórico, com as opções `probcons` e `clustalw`; ao configurar uma execução, o usuário escolhe uma dessas opções e informa os parâmetros correspondentes. A Tabela IV.2 apresenta esse formulário como exemplo concreto do mecanismo de configuração do

AranduFlow.

Tabela IV.2: Campos de configuração de uma execução do NMFSt.P no AranduFlow.

Grupo	**Campo**	**Descrição e preenchimento**
Alinhamento	algorithm	Algoritmo de alinhamento múltiplo: probcons ou clustalw.
ClustalW	-TYPE	Tipo das sequências. Opção cadastrada: PROTEIN.
ClustalW	-ALIGN	Ativa o alinhamento completo. Valor booleano.
ClustalW	-OUTPUT	Formato de saída: CLUSTAL, FASTA, PHYLIP ou NEXUS. O executor do NMFSt.P requer CLUSTAL.
ClustalW	-OUTORDER	Ordem das sequências no resultado: ALIGNED ou INPUT.
ClustalW	-GAPOPEN	Penalidade de abertura de lacuna. Intervalo cadastrado: 5 a 20 (real).
ClustalW	-GAPEXT	Penalidade de extensão de lacuna. Intervalo cadastrado: 0 a 1 (real).
ClustalW	-QUIET	Modo silencioso, que reduz as mensagens do processamento. Valor booleano.
ProbCons	-clustalw	Saída em formato ClustalW. Valor booleano; deve ser verdadeiro para o executor do NMFSt.P.
ProbCons	-c	Iterações de transformação de consistência. Intervalo cadastrado: 0 a 5 (inteiro).
ProbCons	-ir	Iterações de refinamento do alinhamento. Intervalo cadastrado: 0 a 1000 (inteiro).
ProbCons	-pre	Iterações de pré-treinamento. Intervalo cadastrado: 0 a 20 (inteiro).
Árvore filogenética	evolutionary_model	Método de construção da árvore: upgma ou nj.
Árvore filogenética	tree_format	Formato da árvore: nexus ou newick.
Dados de entrada	Arquivos de entrada	Arquivos de sequências em formato FASTA utilizados pelo workflow.
Infraestrutura	Cluster de destino	Seleção de uma das infraestruturas cadastradas no sistema.
Infraestrutura	Threads paralelas	Quantidade de threads solicitadas: inteiro positivo, limitado pela capacidade cadastrada do cluster.
Configuração alternativa	Configuração avançada em JSON	Habilita a importação dos parâmetros científicos por arquivo JSON.

Os campos identificados como ClustalW ou ProbCons são exibidos de acordo com o algoritmo selecionado. Os intervalos apresentados correspondem aos limites cadastrados no sistema, enquanto as exigências de saída em formato ClustalW decorrem do executor do NMFSt.P. Um mesmo cadastro pode, portanto, originar diferentes configurações de execução, variando os parâmetros científicos, os arquivos e os recursos computacionais. O preenchimento do formulário não inicia o processamento: ele antecede a solicitação da estimativa temporal e a confirmação da submissão.

Após o preenchimento do formulário de configuração de uma execução, a interface organiza as informações em uma representação estruturada no formato JSON. Esse documento reúne a identificação do *workflow*, seus parâmetros de domínio, as referências aos dados de entrada, seus metadados e a configuração da infraestrutura selecionada. O uso de uma representação estruturada estabelece um contrato de comunicação entre a interface e os demais componentes do *gateway*, evitando que a lógica interna de execução fique diretamente acoplada aos elementos visuais da aplicação. A interface também permite importar e exportar essas configurações, favorecendo a reprodução e a reutilização de execuções previamente definidas.

Com base nas informações fornecidas, o usuário pode solicitar uma predição antes de submeter o *workflow*. Nesse momento, os atributos relevantes são extraídos da configuração e convertidos no vetor $\mathbf{x}$ esperado pelo modelo associado ao *workflow*. O intervalo de tempo estimado é então apresentado na própria interface, permitindo que o usuário examine a previsão juntamente com os parâmetros e a infraestrutura selecionados. A estimativa possui caráter informativo: cabe ao usuário decidir se mantém a configuração, modifica algum parâmetro ou prossegue com a submissão.

Uma vez confirmada a configuração, a interface encaminha o documento JSON ao Gerenciador de Predição e Persistência. Esse componente registra os dados da execução e aciona os módulos responsáveis pela comunicação com a infraestrutura escolhida. A interface permite consultar o histórico das submissões confirmadas, incluindo suas configurações, previsões e, quando disponível, o tempo observado. Essa integração reúne a definição da execução e a consulta posterior de seus registros em um único ambiente.

Na versão utilizada neste trabalho, a interface não implementa mecanismos de autenticação ou gerenciamento de contas de usuários. O acesso foi realizado diretamente no ambiente controlado empregado nos experimentos, uma vez que o objetivo da implementação foi validar o fluxo de configuração, predição, submissão e registro dos *workflows*. A incorporação de autenticação, autorização e gerenciamento de credenciais constitui uma extensão necessária para a utilização do *gateway* em ambientes compartilhados ou de produção.

IV.5 Gerenciador de Predição e Persistência

O Gerenciador de Predição e Persistência, identificado como módulo 2 na Figura IV.1, centraliza o tratamento das configurações produzidas pela Interface *Web*, a comunicação com os modelos preditivos e o armazenamento dos registros associados às execuções confirmadas. Embora essas responsabilidades possam ser implementadas por serviços distintos, elas são apresentadas como um único módulo lógico por participarem do mesmo fluxo de preparação e registro de uma execução no `AranduFlow`.

Quando o usuário solicita uma estimativa, o gerenciador recebe a configuração estruturada em JSON e identifica o *workflow* selecionado. Essa identificação, representada por $\mathcal{W}$, é utilizada para determinar o esquema de parâmetros, as regras de transformação dos dados e o modelo preditivo correspondente. Entretanto, conforme definido na Seção IV.2, $\mathcal{W}$ não é fornecido diretamente ao modelo. A entrada efetivamente processada é o vetor de características $\mathbf{x}$, construído a partir dos parâmetros de domínio, dos metadados dos dados de entrada e dos atributos da infraestrutura computacional.

A construção de $\mathbf{x}$ segue o mesmo procedimento de preparação utilizado durante o treinamento do modelo. Essa etapa pode envolver a seleção de atributos, a transformação de variáveis categóricas, a normalização de valores numéricos e a organização das características na ordem esperada pelo modelo. Essa correspondência é necessária para que a representação produzida durante o uso do *gateway* seja compatível com aquela empregada na construção do modelo preditivo.

Após o pré-processamento, o gerenciador encaminha $\mathbf{x}$ ao modelo associado ao *workflow*. O resultado corresponde a uma classe de tempo de execução, posteriormente convertida no intervalo temporal correspondente. Esse intervalo é devolvido à interface *Web* e apresentado ao usuário antes da submissão. Nesse momento, a previsão é considerada provisória e permanece apenas no contexto da interação corrente.

A utilização do modelo pressupõe que as características da nova execução estejam razoavelmente representadas pelo histórico empregado no treinamento. Na versão desenvolvida, o `AranduFlow` não implementa um mecanismo específico para detectar configurações fora da distribuição observada nem associa à faixa prevista uma medida de confiança baseada nessa condição. Portanto, quando uma nova infraestrutura ou configuração apresenta características substancialmente diferentes das utilizadas no treinamento, a estimativa pode sofrer degradação e deve ser interpretada com cautela.

A apresentação prévia da estimativa permite que o usuário examine o resultado e, caso necessário, altere parâmetros, metadados ou recursos computacionais antes de prosseguir. Cada alteração pode resultar na construção de um novo vetor $\mathbf{x}$ e, consequentemente, em uma nova estimativa. Essas consultas intermediárias não são armazenadas no banco de dados, pois não cor-

respondem a execuções efetivamente submetidas.

A persistência ocorre somente quando o usuário confirma a configuração e solicita a execução do *workflow*. Nesse momento, o gerenciador registra a configuração estruturada, o vetor de características utilizado e o intervalo de tempo previsto. Essa decisão evita que simulações, alterações de formulário ou previsões descartadas sejam contabilizadas como execuções reais no histórico do *gateway*.

Depois da confirmação, a configuração é encaminhada aos componentes responsáveis pela seleção da infraestrutura e pela submissão do *workflow*. Ao término do processamento, o registro é complementado com o tempo de execução observado retornado pelo executor. A associação entre a configuração submetida, a estimativa fornecida e o valor posteriormente observado permite manter um histórico dos experimentos e comparar o comportamento previsto com o comportamento real do *workflow*.

Na versão desenvolvida, os modelos são treinados externamente ao fluxo operacional do `AranduFlow` e posteriormente incorporados ao Gerenciador de Predição e Persistência. Para cada *workflow*, é mantido um artefato preditivo ativo, acompanhado das transformações necessárias para construir o vetor de características no formato utilizado durante o treinamento. A atualização do modelo ocorre pela substituição manual desse artefato após um novo processo de treinamento. O protótipo não implementa versionamento formal dos modelos nem associa aos registros das execuções um identificador da versão utilizada. Dessa forma, o histórico permite relacionar a configuração submetida, o vetor de características, a faixa prevista e o tempo observado, mas não preserva as versões anteriores dos modelos para reprodução integral de previsões após sua atualização.

O protótipo utiliza o `SQLite` como mecanismo de persistência. Essa escolha simplifica a implantação do *gateway*, uma vez que o banco pode ser executado localmente, não exige a configuração de um servidor de banco de dados independente e atende às necessidades do ambiente controlado empregado nos experimentos. No esquema utilizado, são armazenadas informações sobre os *workflows* cadastrados, seus parâmetros e restrições, as infraestruturas disponíveis e as execuções confirmadas.

A versão desenvolvida não mantém no banco de dados o estado corrente das execuções, como aguardando na fila, em processamento, concluída ou interrompida. O registro está concentrado na configuração submetida, no vetor de características, na previsão realizada e no tempo observado quando a execução é concluída com sucesso. Portanto, o histórico mantido pelo *gateway* não deve ser interpretado como um sistema de monitoramento do escalonador.

Embora adequado à validação experimental apresentada neste trabalho, o `SQLite` possui limitações para cenários com múltiplos usuários e elevado número de acessos simultâneos. Como evolução do `AranduFlow`, prevê-se a migração para um sistema gerenciador de banco de dados mul-

tiusuário, capaz de oferecer maior controle de concorrência, gerenciamento de acesso, escalabilidade e disponibilidade. Essa alteração pode ser realizada preservando o modelo lógico de persistência e o fluxo de interação dos demais componentes do *gateway*.

IV.6 Seletor e Conector de Infraestrutura

O Seletor e Conector de Infraestrutura, identificado como módulo 3 na Figura IV.1, é responsável por intermediar a comunicação entre o `AranduFlow` e os ambientes computacionais nos quais os *workflows* são executados. Esse módulo separa a configuração lógica de uma execução dos mecanismos específicos necessários para submetê-la a um *cluster*, reduzindo o acoplamento entre a Interface *Web* e as particularidades do ambiente de processamento de alto desempenho.

O termo seletor não indica a realização de uma escolha automática ou de uma otimização entre diferentes infraestruturas. As infraestruturas disponíveis são cadastradas no *gateway* e apresentadas ao usuário durante a configuração da execução. Cabe ao usuário selecionar o ambiente em que pretende executar o *workflow*, considerando os recursos computacionais disponíveis e as necessidades de seu experimento.

O cadastro de uma infraestrutura reúne as informações necessárias para sua identificação e utilização, como nome do ambiente, endereço do servidor remoto, recursos disponíveis e mecanismo de submissão de tarefas. Dessa maneira, a interface apresenta uma representação simplificada do ambiente, enquanto os detalhes necessários para estabelecer a comunicação remota permanecem encapsulados no conector.

Após a confirmação da execução, o Gerenciador de Predição e Persistência encaminha ao conector o documento JSON contendo a identificação do *workflow*, seus parâmetros, os nomes dos arquivos de entrada, seus metadados e os recursos selecionados. Esse documento é transferido para o *cluster* por meio do protocolo *SSH File Transfer Protocol* (SFTP).

Depois da transferência da configuração, o conector estabelece uma sessão *Secure Shell* (SSH) com o servidor remoto. Por meio dessa sessão, é executado o comando responsável pela submissão do *workflow* ao sistema de gerenciamento de filas do *cluster*. Em todos os experimentos realizados neste trabalho, foi utilizado o *Sun Grid Engine* (SGE), e as execuções foram submetidas por meio do comando `qsub`.

A partir da submissão, o SGE assume o gerenciamento da execução, incluindo sua permanência na fila, a alocação dos recursos solicitados e o início do processamento quando esses recursos se tornam disponíveis. Portanto, o intervalo estimado pelo modelo do `AranduFlow` corresponde ao tempo de processamento do *workflow* nas condições representadas pelo vetor de características, não ao tempo de espera introduzido pela fila do escalonador.

A utilização conjunta de SFTP, SSH e SGE representa a implementação empregada nos ex-

perimentos deste trabalho. Entretanto, a responsabilidade lógica do conector não está limitada a essas tecnologias. Outros protocolos de transferência, mecanismos de acesso remoto ou sistemas de gerenciamento de filas podem ser incorporados mediante a implementação de conectores específicos, sem exigir alterações no fluxo de configuração disponibilizado ao usuário.

Como o `AranduFlow` não armazena o estado corrente da execução, o conector é responsável pela submissão, mas não realiza o acompanhamento contínuo da fila do SGE. O histórico mantido pelo *gateway* concentra-se nas configurações confirmadas, nas previsões produzidas e nos tempos posteriormente observados sobre as execuções, não substituindo as ferramentas de monitoramento disponibilizadas pelo próprio escalonador.

IV.7 Executores de *Workflows*

Os Executores de *Workflows*, identificados como módulo 4 na Figura IV.1, constituem a camada responsável por transformar a configuração recebida do *gateway* em uma execução concreta no ambiente computacional. Enquanto os demais componentes do `AranduFlow` tratam os *workflows* por meio de uma representação comum, cada aplicação científica possui parâmetros, arquivos, comandos e dependências próprios. Por esse motivo, foi desenvolvido um programa executor específico para cada *workflow* integrado ao *gateway*.

Após a submissão pelo SGE, o programa correspondente ao *workflow* é iniciado no *cluster* e recebe o documento JSON transferido pelo conector. Esse documento contém os parâmetros informados pelo usuário, os nomes dos arquivos de entrada, seus metadados e a configuração dos recursos computacionais.

O executor interpreta os campos da configuração e os converte nos argumentos, comandos e estruturas exigidos pela aplicação científica. Essa etapa concentra o conhecimento específico do *workflow* e impede que a interface *Web* ou o conector de infraestrutura precisem conhecer os detalhes de execução de cada aplicação. Assim, alterações relacionadas a comandos, formatos de entrada ou organização interna das atividades ficam restritas ao respectivo executor.

Os executores não são responsáveis pela persistência dos dados do *gateway* nem pelo armazenamento do estado da execução. Sua responsabilidade termina após a realização do *workflow* e a devolução do tempo observado. Essa delimitação mantém separadas a lógica científica de cada aplicação e a lógica de gerenciamento mantida pelo `AranduFlow`.

Embora os *workflows* empregados nos experimentos possuam implementações específicas, eles funcionam como casos de uso da arquitetura proposta. A incorporação de um novo *workflow* requer a definição de seu esquema de configuração e a implementação de um executor capaz de interpretar esse esquema, organizar suas atividades e iniciar a aplicação correspondente. Os demais componentes do *gateway* podem continuar utilizando o mesmo fluxo de configuração, predição,

submissão e persistência.

IV.8 Registro do Tempo de Execução Observado

A submissão de um *workflow* ao SGE ocorre de maneira assíncrona, pois a solicitação pode permanecer na fila antes que os recursos computacionais sejam disponibilizados. Consequentemente, a conexão utilizada pelo *gateway* para executar o comando `qsub` não permanece aberta até o término do processamento. Para permitir que o tempo observado seja posteriormente associado à previsão, foi implementado um mecanismo de retorno assíncrono entre o executor e o Gerenciador de Predição e Persistência.

Quando o usuário confirma uma execução, o gerenciador cria seu registro no banco de dados e atribui a ele um identificador único. Esse identificador é incluído no documento JSON transferido ao *cluster* e acompanha a execução durante todo o processamento. Dessa forma, o executor não precisa reproduzir os parâmetros utilizados na busca pelo registro nem depender da ordem em que diferentes *workflows* são concluídos.

O tempo é medido no próprio programa executor por meio de um relógio monotônico. A medição é iniciada imediatamente antes do acionamento das atividades do *workflow* e encerrada depois que todas as atividades são concluídas. Portanto, o valor observado inclui a coordenação das atividades e o processamento realizado pelo *workflow*, mas exclui o tempo de espera na fila do SGE, a transferência do documento de configuração, o estabelecimento da conexão SSH e o processamento da predição.

Após a conclusão bem-sucedida, o executor realiza uma requisição HTTP para um ponto de acesso disponibilizado pelo Gerenciador de Predição e Persistência. O corpo da requisição contém somente o identificador da execução e o tempo observado, expresso em segundos, conforme o exemplo a seguir. Ao receber a requisição, o gerenciador valida os valores informados, localiza o registro correspondente no `SQLite` e atualiza o campo destinado ao tempo real de execução.

```
{
  "execution_id": "identificador-unico-da-execucao",
  "elapsed_time_seconds": 427.36
}
```

Esse mecanismo pode ser representado pela seguinte operação da Equação IV.4.

$$\text{atualizarTempo}(id_e, t_{\text{obs}}), \tag{IV.4}$$

em que id_e representa o identificador único atribuído à execução e t_{obs} corresponde ao tempo transcorrido entre o início e o término das atividades do *workflow*. A utilização do identificador

evita ambiguidades mesmo quando diversas execuções do mesmo *workflow* utilizam configurações semelhantes.

Se alguma atividade do *workflow* terminar com erro, o tempo observado não é encaminhado ao *gateway*, pois a execução não produziu uma medição completa comparável aos dados utilizados no treinamento e na avaliação dos modelos. As informações do erro permanecem disponíveis nos arquivos de saída e de erro gerados pelo SGE.

A ausência do retorno do tempo não é convertida em um estado de falha no banco de dados, uma vez que a versão desenvolvida não implementa o acompanhamento centralizado dos estados das execuções. O registro permanece com o campo de tempo observado não preenchido. A incorporação de estados, mensagens de erro e mecanismos de recuperação constitui uma possibilidade de evolução do *gateway*.

Ao final desse processo, cada execução concluída possui, no mesmo registro, sua configuração, o vetor de características, o intervalo previsto e o tempo efetivamente observado. Essa associação fornece os dados necessários para comparar previsões e observações, além de possibilitar a utilização de execuções anteriores em análises posteriores e em processos de atualização dos modelos.

IV.9 Considerações Finais

Este capítulo apresentou o `AranduFlow` como um *gateway* científico composto por módulos que abrangem a configuração de aplicações, a predição de faixas temporais, a persistência das submissões confirmadas, a comunicação com ambientes de HPC e a execução dos *workflows*. A separação entre as responsabilidades compartilhadas pelo *gateway* e os elementos específicos de cada aplicação permite incorporar diferentes *workflows* sem modificar o fluxo geral oferecido ao usuário.

A camada preditiva foi posicionada como parte desse processo operacional, e não como um componente isolado ou como a totalidade da contribuição.

Capítulo V Avaliação Experimental do `AranduFlow`

Este capítulo apresenta os experimentos conduzidos para avaliar o desempenho dos modelos de predição de tempo de execução utilizados no `AranduFlow`. O objetivo principal é investigar a capacidade do sistema de estimar, com base em dados históricos de execução, faixas de tempo de execução para *workflows* científicos executados em ambientes HPC.

Para avaliar a aplicação da abordagem preditiva a diferentes perfis de carga, a avaliação experimental baseia-se em dois estudos de caso de naturezas distintas: o *workflow* NMFSt.P, uma aplicação de bioinformática computacionalmente intensiva, apresentado nas Seções V.2.1 e V.3.1; e o *workflow* Montage, uma aplicação de astronomia intensiva em operações de E/S e manipulação matemática de arquivos de imagem, apresentado nas Seções V.2.2 e V.3.2.

A Seção V.1 apresenta o delineamento da avaliação, incluindo os ambientes, a configuração dos experimentos, a discretização do tempo, os modelos e a extensão experimental de calibração local. Em seguida, a Seção V.2 caracteriza os conjuntos de dados, e a Seção V.3 apresenta e discute os resultados. Ao final do capítulo, a Seção V.4 sintetiza as evidências obtidas, enquanto a Seção V.5 delimita as limitações à validade da avaliação.

V.1 Delineamento da Avaliação Experimental

Esta seção apresenta a metodologia da avaliação experimental do `AranduFlow`, com o objetivo de avaliar a acurácia das predições de tempo de execução produzidas pelo sistema. Para isso, a avaliação é feita a partir de dois *workflows* científicos com características distintas: o NMFSt.P, um *workflow* de bioinformática intensivo em CPU; e o Montage, um *workflow* de astronomia com alto consumo de recursos de E/S. Ambos os *workflows* foram submetidos utilizando o sistema proposto sob diversas configurações de recursos e tamanhos de dados de entrada.

V.1.1 Configuração do Ambiente

A avaliação experimental foi realizada utilizando dois tipos de máquinas distintas, denominadas *Cluster* A e *Cluster* B. Esses perfis diferem significativamente em termos de arquitetura de processador, poder de processamento *single-core* e capacidade de memória. As principais características das infraestruturas estão descritas na Tabela V.1.

Tabela V.1: Características de *hardware* e sistema operacional das classes de infraestrutura HPC avaliadas

Atributo	*Cluster* A (Xeon)	*Cluster* B (Core i7)
Sistema Operacional	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS
Processador (CPU)	2x Intel Xeon E5-2620 v3	1x Intel Core i7-13700
Threads por nó	24	24
Memória por nó	64 GB	128 GB
Número de nós	6	3

No *Cluster* A, cada nó é composto por dois processadores físicos da família Intel Xeon E5-2620 v3, totalizando 24 *threads* de processamento por nó. Sob a modelagem principal do `AranduFlow`, esse *hardware* é representado por um vetor de características do *hardware*, contendo: modelo do processador (`cpu_model` = *Intel Xeon E5-2620 v3*), número de núcleos físicos (`physical_cores` = 12) e lógicos (`logical_cores` = 24), tamanho das memórias caches L2 (`l2_cache_kb` = 3.072 KB) e L3 (`l3_cache_kb` = 30.720 KB), capacidade total de *Random Access Memory* (RAM) (`ram_gb` = 62,79 GB), número de canais de memória ativos (`ram_channels` = 8) e velocidade nominal da RAM (`ram_speed` = 0, indicando valor não especificado). No *Cluster* B, cada nó é composto por um único processador físico Intel Core i7-13700, também totalizando 24 *threads* de processamento por nó, cujas características são mapeadas no vetor como: `cpu_model` = *Intel Core i7-13700*, `physical_cores` = 16, `logical_cores` = 24, `l2_cache_kb` = 24.576 KB, `l3_cache_kb` = 30.720 KB, `ram_gb` = 125,49 GB, `ram_channels` = 0 (não especificado) e `ram_speed` = 0. Ambos os ambientes utilizam o SGE, versão 8.1.9 (revisão r192), para o gerenciamento de filas de submissão e alocação de recursos nos nós de computação.

Como abordagem de representação secundária e comparativa, são utilizadas pontuações de *benchmark*. Nessa estratégia, o *hardware* do *Cluster* A recebe as pontuações de `cpu_points = 1692`, `disk_points = 2583` e `ram_points = 3566`. Já o *hardware* do *Cluster* B recebe as pontuações de `cpu_points = 4095`, `disk_points = 5701` e `ram_points = 10609`. A definição dessas pontuações visa quantificar de forma agregada e simplificada a capacidade de processamento e transferência de cada nó computacional. A pontuação de CPU foi extraída diretamente da métrica *Single Thread Rating* da base de dados internacional `PassMark`[1], a qual avalia o desempenho de instruções sequenciais por *thread* de processamento. Para caracterizar o desempenho de E/S de disco e a largura de banda da memória RAM, foram executados *scripts* de *benchmark* em `Python`. O *benchmark* de disco mede a taxa de transferência sequencial física em MB/s, enquanto o *benchmark* de memória estima a largura de banda da memória RAM em MB/s. Em ambos os casos, a pontuação numérica atribuída corresponde à média aritmética arredondada das taxas observadas, consolidando uma escala de desempenho relativo entre as infraestruturas computacionais avaliadas.

[1] https://www.cpubenchmark.net

V.1.2 Configuração dos Experimentos

Os dois *workflows* foram executados em diferentes condições por meio do `AranduFlow`. Para cada execução concluída com sucesso, o sistema associou o tempo observado à configuração, aos dados de entrada e à infraestrutura utilizada. Os registros resultantes constituíram os conjuntos históricos empregados no treinamento e na avaliação dos modelos preditivos.

Para o *workflow* científico de filogenia NMFSt.P, as variações foram aplicadas tanto ao volume e características dos dados de entrada quanto aos parâmetros de configuração do *workflow*. Foram variados o número de sequências e de arquivos de entrada (em formato multi-FASTA [Ocaña and Dávila, 2011; Ferrari et al., 2023]), os algoritmos de alinhamento de sequências selecionados (`clustalw` ou `probcons`), os formatos de geração da árvore filogenética resultante (`nexus` ou `newick`) e a quantidade de *threads* alocadas para o processamento.

Para o *workflow* de processamento de imagens astronômicas Montage, a variabilidade dos experimentos abrange parâmetros específicos do domínio da astronomia e de desempenho físico. As variações contemplam o sistema de coordenadas astronômicas (`system`), o limiar de erro de projeção (`mproject_e`), o tipo de coadição de imagens (`coadd_type`), a tabela de cores utilizada, o algoritmo de estiramento de contraste (`stretch`) e o limiar dinâmico. Em termos de volumetria e paralelismo, variaram-se a quantidade de *threads*, o tamanho de entrada em MB (`size_mb`) e a quantidade de arquivos de imagem (`qtt_files`). Além disso, foram computadas variáveis derivadas de carga de trabalho concorrente, como arquivos por *thread* e MB por *thread*.

Para fins de modelagem preditiva, os atributos coletados nos dois estudos de caso foram organizados conforme a representação definida na Seção IV.2. O conjunto **d** reúne os metadados de domínio e os parâmetros de configuração específicos de cada aplicação, como os algoritmos de alinhamento e o formato da árvore no NMFSt.P, ou o sistema de coordenadas e o algoritmo de estiramento de contraste no Montage. O conjunto **e** descreve as características dos dados de entrada, incluindo a quantidade e o tamanho dos arquivos, bem como o número de sequências ou imagens processadas. Por sua vez, o conjunto **h** contém as características da infraestrutura e dos recursos solicitados, como quantidade de *threads*, capacidade de memória e atributos físicos do processador. Também foram construídos atributos derivados que relacionam a dimensão da entrada aos recursos solicitados, como arquivos por *thread* e MB por *thread*. Essa organização permite analisar tanto a contribuição de cada grupo quanto as interações entre dados, configuração e infraestrutura.

As bases utilizadas na avaliação foram constituídas exclusivamente por execuções concluídas com sucesso. Execuções interrompidas ou finalizadas com erro não foram incorporadas aos conjuntos de dados, pois não produziram uma medição completa do tempo de processamento comparável às observações utilizadas no treinamento e na avaliação dos modelos.

V.1.3 Discretização do Tempo de Execução

O `AranduFlow` reformula a predição temporal contínua como um problema de classificação multiclasse por meio da discretização da variável contínua de tempo de execução. Para que a avaliação dos dois *workflows* científicos siga exatamente o mesmo padrão metodológico, adotou-se a discretização por quantis para ambos os *workflows*.

Foram avaliados cinco cenários de discretização, dividindo o tempo de execução em 3, 5, 7, 10 e 15 classes. Em cada cenário, as fronteiras dos quantis foram calculadas exclusivamente a partir dos tempos de execução pertencentes ao conjunto de treinamento. Após essa definição, as mesmas fronteiras foram mantidas fixas e aplicadas às amostras do conjunto de teste, sem utilizar os tempos observados nesse conjunto para recalcular os limites das classes. A discretização por quantis divide os dados de treinamento de modo que cada classe de tempo de execução contenha frequências da maneira mais equilibrada possível. Essa escolha de projeto mitiga problemas de desbalanceamento de classes em distribuições assimétricas, facilitando o treinamento estável dos classificadores de aprendizado de máquina e evitando previsões enviesadas para faixas temporais curtas de maior ocorrência. Como representação metodológica, a Tabela V.2 ilustra os intervalos de tempo e a quantidade de amostras correspondente para o cenário de 5 classes do NMFSt.P, e a Tabela V.3 apresenta o cenário correspondente de 5 classes para o Montage. A distribuição completa dos intervalos de tempo de execução e frequências de amostras para os cenários restantes (3, 7, 10 e 15 classes) de ambos os *workflows* encontra-se detalhada no Apêndice B.

Os intervalos adotam a convenção $(a, b]$, de modo que cada classe inclui seu limite superior e exclui seu limite inferior, com exceção da primeira classe, que também inclui o menor valor observado.

Tabela V.2: Distribuição de frequências e intervalos de tempo das classes de tempo de execução (cenário com cinco classes) no NMFSt.P.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	286	4,84	59,57
Q2	285	59,57	222,75
Q3	286	222,75	1.040,48
Q4	285	1.040,48	10.114,91
Q5	286	10.114,91	221.539,98

V.1.4 Modelos de Aprendizado de Máquina no `AranduFlow`

Esta seção descreve os modelos de aprendizado de máquina, as estratégias de pré-processamento e a metodologia de avaliação adotados para prever o tempo de execução de *workflows* no `AranduFlow`.

Tabela V.3: Distribuição de frequências e intervalos de tempo das classes de tempo de execução (cenário com cinco classes) no Montage.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	554	5,07	25,44
Q2	554	25,44	46,21
Q3	553	46,21	171,41
Q4	554	171,41	790,19
Q5	554	790,19	92.602,43

A formulação da tarefa de predição retoma a representação introduzida na Seção IV.2, na qual uma execução é descrita por $\mathcal{K} = (\mathcal{W}, \mathbf{x})$. Nessa representação, $\mathcal{W}$ identifica o *workflow* e determina os elementos específicos associados à aplicação, enquanto somente o vetor $\mathbf{x}$ é processado pelo modelo preditivo. Conforme definido na Equação IV.2, esse vetor reúne os metadados de domínio e configuração $\mathbf{d}$, as características dos dados de entrada $\mathbf{e}$ e as informações da infraestrutura e dos recursos solicitados $\mathbf{h}$. A predição é formulada como um problema de classificação multiclasse, no qual o tempo de execução contínuo observado é discretizado em faixas temporais baseadas em quantis.

Foram avaliados quatro algoritmos de aprendizado supervisionado: Regressão Linear (do inglês, *Linear Regression*, LiR), Regressão Logística (do inglês, *Logistic Regression*, LoR), Classificador por Árvore de Decisão (do inglês, *Decision Tree Classifier*, DTC) e Classificador de Florestas Aleatórias (do inglês, *Random Forest Classifier*, RFC). A seleção desses algoritmos justifica-se pelas suas características complementares: os modelos baseados em árvores (DTC e RFC) são reconhecidamente eficazes para mapear interações não lineares complexas em dados tabulares heterogêneos sem exigir hipóteses estritas sobre a distribuição dos dados, oferecem recursos de interpretação, como importância de atributos, especialmente no caso das árvores de decisão; por sua vez, os modelos lineares tradicionais (LiR e LoR) foram incorporados para permitir avaliar se abordagens mais simples e de baixo custo computacional seriam suficientes para o problema proposto. Embora a LiR seja, inerentemente, um modelo baseado em regressão, ele foi intencionalmente incluído na avaliação para investigar a adequação de preditores contínuos para estimar faixas de tempo de execução. Para permitir uma comparação justa com as abordagens baseadas em classificação, as saídas contínuas produzidas por LiR foram discretizadas utilizando os mesmos intervalos de tempo de execução empregados para a variável alvo. Esta escolha de projeto reflete o objetivo prático do `AranduFlow`, que visa fornecer estimativas em faixas do tempo de execução antes da submissão do *workflow*, em vez de predições exatas de tempo de execução.

Diferentes *pipelines* de pré-processamento foram empregados para acomodar as características de cada algoritmo de aprendizado. Para LiR e LoR, os atributos numéricos foram normalizados

por meio de padronização *standard scaling*, e as variáveis categóricas foram codificadas via *one-hot encoding*. Em contraste, para DTC e RFC, apenas *one-hot encoding* foi aplicado, sem normalização dos atributos. Esta escolha é motivada pelo fato de que modelos baseados em árvores são invariantes à escala dos atributos, enquanto modelos baseados em regressão normalmente se beneficiam de atributos de entrada normalizados para garantir estabilidade numérica e melhor comportamento de convergência.

Todas as etapas de pré-processamento foram encapsuladas em *pipelines* de aprendizado de ponta a ponta, a fim de evitar vazamento de dados e garantir uma avaliação justa. Esses *pipelines* foram ajustados exclusivamente com os dados de treinamento, preservando, assim, a integridade do protocolo experimental. O treinamento e a avaliação dos modelos foram realizados utilizando a biblioteca `scikit-learn`, seguindo suas práticas padrão de implementação. Para estabelecer uma comparação reprodutível sem otimização específica para os conjuntos avaliados, todos os modelos foram treinados com as configurações padrão de hiperparâmetros fornecidas pela biblioteca. Essa escolha permite analisar o comportamento dos algoritmos em suas configurações de referência, sem realizar busca ou seleção adicional de hiperparâmetros.

Como *baseline* operacional, utilizou-se um classificador majoritário que sempre prediz a classe majoritária observada no conjunto de treinamento. Como a discretização por quantis produz classes aproximadamente balanceadas, esse *baseline* representa o desempenho esperado de uma política que ignora os atributos de domínio e infraestrutura e escolhe apenas a faixa temporal mais frequente. Para cada cenário com C classes, sua acurácia esperada aproxima-se de $1/C$, enquanto o F1-Macro permanece baixo por concentrar previsões em uma única classe.

O desempenho dos modelos foi avaliado por meio de um protocolo em duas etapas. Inicialmente, aplicou-se uma divisão *hold-out* aleatória, alocando 80% dos dados para treinamento e 20% para teste. Após essa separação, os limites dos quantis foram estimados exclusivamente sobre os tempos observados nos 80% destinados ao treinamento e utilizados para atribuir os rótulos das amostras de ambos os conjuntos. Em seguida, apenas sobre o conjunto de treinamento, realizou-se uma validação cruzada estratificada com 5 *folds*. Para cada combinação de *workflow*, número de classes e algoritmo, foram treinados cinco modelos candidatos, cada um usando quatro *folds* para ajuste e um *fold* para validação. O modelo candidato com maior F1-Macro no respectivo *fold* de validação, usando a acurácia como critério secundário de desempate, foi selecionado para a avaliação final. O conjunto de teste do *hold-out* permaneceu isolado durante esse processo e foi utilizado apenas uma vez, para calcular as métricas finais reportadas neste capítulo. A qualidade das predições de tempo de execução do `AranduFlow` foi avaliada utilizando acurácia e F1-Macro como métricas de desempenho preditivo.

V.1.5 Estratégias de Calibração Dinâmica Local

Como extensão experimental da abordagem preditiva do `AranduFlow`, investigou-se uma estratégia de calibração dinâmica local para situações em que um modelo previamente treinado é aplicado a uma infraestrutura ainda não representada adequadamente pelo histórico de treinamento. Essa extensão foi avaliada com o objetivo de adaptar as previsões ao ambiente computacional de destino sem exigir a reconstrução do modelo exclusivamente a partir de um novo conjunto de dados. A estratégia baseia-se em conceitos de aprendizado por transferência [Pan and Yang, 2009] e adaptação de domínio sob desvio de distribuição de dados. Por se tratar de uma investigação experimental, a calibração não integra o fluxo operacional regular do *gateway* descrito no Capítulo IV.

O procedimento inicia-se com uma sondagem composta por $N_{cal} = 15$ execuções sob configurações variadas do *workflow*. Essas execuções fornecem a amostra rotulada local $D_{cal} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{15}$, utilizada para representar o perfil de desempenho da infraestrutura de destino.

No Ajuste Fino Ponderado (*Weighted Fine-Tuning*), o conjunto de sondagem local D_{cal} é combinado ao conjunto de treino global original $D_{treino} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$, composto pelas execuções históricas nos *clusters* A e B. O preditor é ajustado novamente utilizando ponderação diferenciada de amostras (*instance re-weighting*) [Shimodaira, 2000]. Nos classificadores, o ajuste preserva a mesma família de modelo avaliada no preditor base, alterando o peso relativo das amostras de origem e de destino durante o treinamento. Para a abordagem linear de regressão, utiliza-se um regressor Ridge regularizado. Define-se um vetor de pesos λ_k, conforme a Equação V.1.

$$\lambda_k = \begin{cases} \lambda_{origem} = 0,1, & \text{se } k \in D_{treino} \\ \lambda_{destino} = 5,0, & \text{se } k \in D_{cal} \end{cases} \tag{V.1}$$

No caso linear, a regressão Ridge minimiza a soma dos resíduos quadráticos ponderados sujeita a uma penalidade L2 sobre os coeficientes ($\alpha = 1,0$), conforme a Equação V.2.

$$\min_{\boldsymbol{\beta}} \sum_{k \in D_{treino} \cup D_{cal}} \lambda_k (y_k - \mathbf{x}_k^T \boldsymbol{\beta})^2 + \alpha \|\boldsymbol{\beta}\|_2^2 \tag{V.2}$$

Para aplicar a regressão linear à classificação multiclasse por quantis temporais, os rótulos categóricos, como Q1 e Q2, são mapeados para valores inteiros ordenados $0, 1, \ldots, C-1$. Após obter a predição contínua $\hat{z} = \mathbf{x}^T \boldsymbol{\beta}$, realiza-se o arredondamento e o confinamento (*clipping*) ao intervalo $[0, C-1]$ para remapear o resultado à classe de quantil correspondente.

Como alternativa, a Calibração de Probabilidade Local (*Platt Scaling*) ajusta uma camada de calibração sobre as previsões dos modelos classificadores que fornecem escores ou probabilidades [Platt, 1999]. Em vez de retreinar o modelo base, utiliza-se o conjunto D_{cal} com 15 amostras para

ajustar um calibrador sigmoide um contra todos (*One-vs-Rest*) sobre as saídas geradas pelo modelo original. Para contornar a ausência eventual de algumas classes de quantis na pequena amostra de calibração, o ajuste é realizado diretamente sobre todo o conjunto D_{cal}, mantendo congelado o modelo original previamente treinado.

Para garantir o rigor metodológico e a validade científica da avaliação trans-infraestrutura, a divisão entre os dados de calibração e teste no novo ambiente computacional segue regras estritas de isolamento. O conjunto de calibração D_{cal} é extraído por amostragem pseudoaleatória reprodutível, com semente fixa, contendo 15 execuções locais do ambiente computacional de destino. As predições reportadas são calculadas exclusivamente sobre as execuções restantes (conjunto de teste local $D_{teste_local} = D_{destino} \setminus D_{cal}$), as quais permanecem completamente ocultas durante o ajuste da Calibração de Platt ou do Ajuste Fino Ponderado. Dessa forma, o modelo base e sua camada de adaptação podem utilizar o conjunto de calibração, mas nunca acessam as amostras reservadas para teste local, impedindo vazamento de informação (*data leakage*) na avaliação final. A ponderação assimétrica entre histórico global ($\lambda_{origem} = 0,1$) e sondagem local ($\lambda_{destino} = 5,0$), combinada à regularização L2 no caso linear, atua como limitador de variância e mitiga o risco de sobreajuste (*overfitting*) à pequena amostra de calibração.

A seleção das 15 execuções constitui um procedimento retrospectivo adotado para avaliar a viabilidade da calibração com uma pequena amostra local, e não um mecanismo operacional implementado pelo `AranduFlow`. Em uma implantação real, como os tempos de execução ainda não são conhecidos, a amostra de sondagem não poderia ser estratificada previamente pelas classes temporais. Sua seleção deveria considerar atributos disponíveis antes da execução, buscando abranger diferentes tamanhos de entrada, parâmetros do *workflow* e configurações de recursos computacionais.

V.1.6 Disponibilidade dos Artefatos e Reprodutibilidade

Os códigos-fonte, os dados e os programas utilizados no processamento, no treinamento e na avaliação dos modelos preditivos, bem como na geração das tabelas e figuras apresentadas nesta dissertação, estão disponíveis no repositório público do `AranduFlow`. Para identificar de forma inequívoca os artefatos utilizados neste trabalho, foi disponibilizada a versão `v1.0-dissertacao`, associada ao *commit* `73bbc9c8d476aa62536c571d2685b04b87c61cbb`[2]. Essa versão contém as dependências fixadas, os dados de entrada, as sementes pseudoaleatórias, as instruções de configuração e os programas necessários para reproduzir os experimentos e regenerar os resultados apresentados.

[2] `https://github.com/AnndreIgor/AranduFlow/releases/tag/v1.0-dissertacao`

V.2 Análise Exploratória dos Dados

Realizou-se uma análise exploratória em ambos os conjuntos de dados para compreender a correlação entre as variáveis explicativas e o tempo de execução observado.

V.2.1 *Workflow* NMFSt.P

O conjunto de dados utilizado para o *workflow* NMFSt.P é composto por 1.428 execuções (tuplas) coletadas e distribuídas em 740 no *Cluster* B e 688 no *Cluster* A. A estrutura dos atributos utilizados para representar essas execuções é descrita no dicionário de dados apresentado na Tabela A.1. Essas execuções contemplam diferentes combinações de metadados de domínio e metadados de infraestrutura.

A variável alvo, *tempo de execução*, é contínua e apresenta variabilidade entre as execuções. A Figura V.1 ilustra a distribuição dos tempos de execução observados nos experimentos, exibindo assimetria à direita, com maior frequência de execuções em faixas temporais menores. O tempo mínimo de execução foi de 4,84 segundos, a média foi de 6.984,27 segundos e o tempo máximo registrado atingiu 221.539,98 segundos (aproximadamente 61,5 horas).

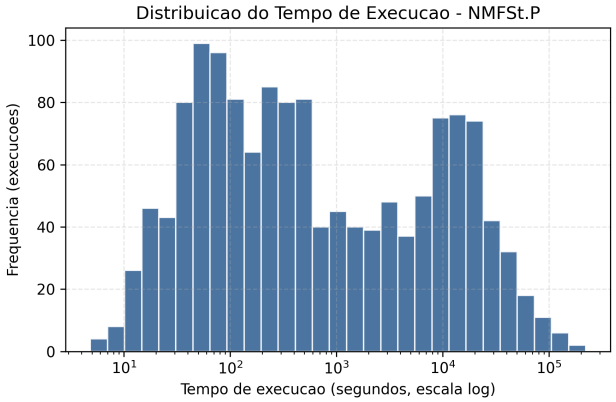


Figura V.1: Distribuição do tempo de execução das múltiplas execuções do *workflow* NMFSt.P.

A Figura V.2 mostra a relação entre o tamanho dos arquivos de sequência de entrada e o tempo de execução observado para o NMFSt.P, mantendo todos os outros atributos constantes. Os resultados sugerem uma tendência aproximadamente linear de aumento no tempo de execução à medida que o tamanho do arquivo cresce no conjunto observado. Essa correlação aponta o tamanho do arquivo de entrada como um fator relevante para a predição.

A Figura V.3 ilustra o tempo de execução em relação ao número de *threads* utilizadas. Observa-se que, inicialmente, o tempo tende a reduzir com o acréscimo de *threads*. Contudo, a partir de aproximadamente 14 *threads*, o tempo de execução passa a apresentar um ponto de saturação, comportamento que pode estar associado à sobrecarga de paralelização, aos custos de sincronização ou à contenção de recursos no ambiente compartilhado.

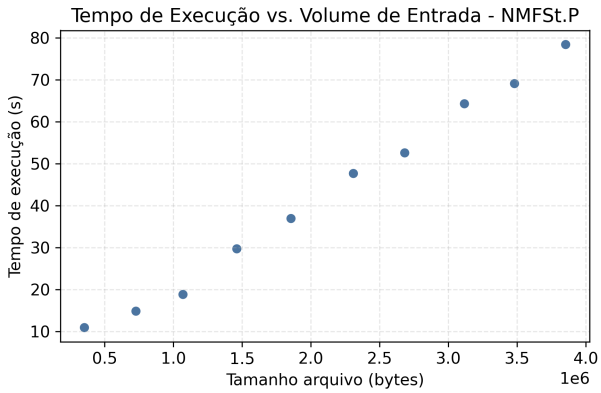


Figura V.2: Tempo de execução do NMFSt.P (em segundos) em função do tamanho do arquivo de entrada (em *bytes*).

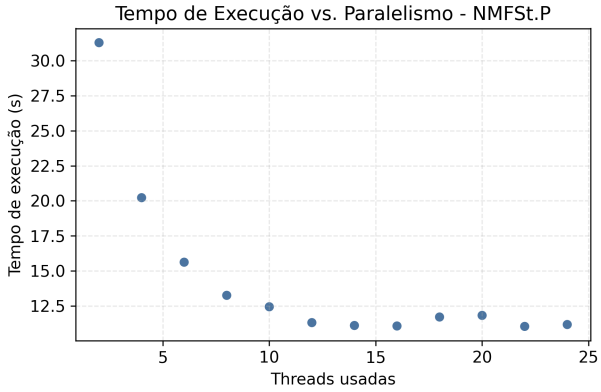


Figura V.3: Tempo de execução do NMFSt.P (em segundos) em função do número de *threads* alocadas.

V.2.2 *Workflow* Montage

O segundo conjunto de dados é composto por registros de execução do *workflow* Montage, uma aplicação de processamento de imagens astronômicas para a geração de mosaicos celestes. A estrutura dos atributos utilizados para representar essas execuções é descrita no dicionário de dados apresentado na Tabela A.2. O conjunto consolidado contém 2.769 execuções, das quais 770 foram executadas no *Cluster* A e 1.999 no *Cluster* B.

A variável-alvo do Montage também apresenta uma distribuição assimétrica à direita, com maior concentração de execuções nas faixas de menor duração e uma cauda prolongada em direção aos tempos mais elevados, como ilustrado na Figura V.4. O tempo mínimo de execução foi de 5,07 segundos, a média foi de 481,93 segundos e o tempo máximo registrado atingiu 92.602,43 segundos (aproximadamente 25,7 horas).

No caso do Montage, a análise exploratória revela padrões específicos de processamento de dados astronômicos. Ao contrário do NMFSt.P, que é intensivo em CPU e opera sobre arquivos individuais compactos de árvores, o Montage processa uma quantidade elevada de arquivos de

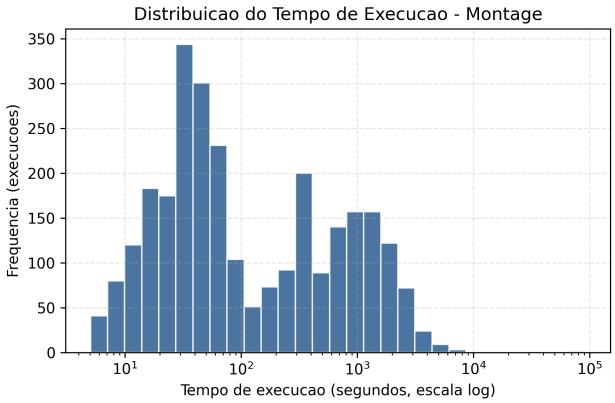


Figura V.4: Distribuição do tempo de execução das múltiplas execuções do *workflow* Montage.

imagens astronômicas (de dezenas a milhares), sendo caracterizado como intensivo em dados.

A relação entre o volume total dos dados de entrada (`size_mb`) e a quantidade de arquivos (`qtt_files`) apresenta correlação com o tempo de execução, conforme ilustrado na Figura V.5. Entretanto, o ganho com o uso de múltiplas *threads* parece sofrer perda de eficiência em cenários com leituras e escritas concorrentes efetuadas pelas tarefas de projeção de imagem (`mProjectPP`) e coadição (`mBgModel` e `mImgtbl`).

Adicionalmente, os atributos de domínio da astronomia parecem desempenhar papel relevante. O sistema de coordenadas astronômicas (`system`) e o algoritmo de contraste (`stretch`) estão associados a variações nas demandas computacionais, uma vez que a reprojeção e o estiramento de escala envolvem operações trigonométricas *pixel* a *pixel*, gerando variabilidade de tempo não capturada apenas pelo tamanho das imagens de entrada.

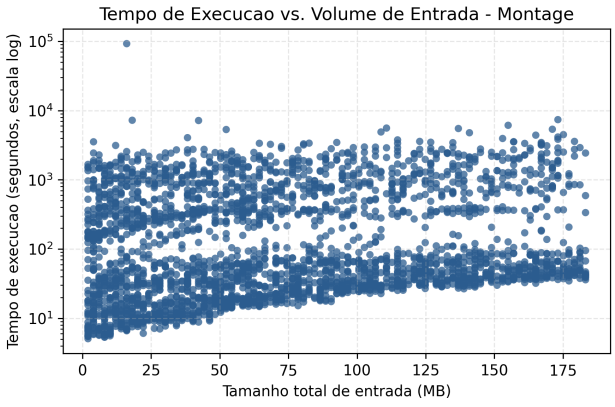


Figura V.5: Tempo de execução do Montage (em segundos) em função do tamanho total de entrada (em MB).

V.3 Resultados e Discussão

A avaliação dos modelos preditivos seguiu o protocolo descrito na Seção V.1.4: primeiro, uma divisão *hold-out* aleatória separou 80% dos dados para treinamento e 20% para teste; em seguida,

o conjunto de treinamento foi submetido a uma validação cruzada estratificada com 5 *folds* para selecionar o melhor modelo candidato para cada *workflow*. As métricas de acurácia e F1-Macro reportadas nesta seção foram calculadas exclusivamente sobre o conjunto de teste mantido isolado pelo *hold-out*. O *baseline* reportado corresponde ao classificador majoritário que prediz sempre a classe majoritária observada no conjunto de treinamento, conforme definido na Seção V.1.4.

V.3.1 Resultados para o *Workflow* NMFSt.P

Os resultados de desempenho preditivo para o NMFSt.P sob a metodologia de discretização por quantis são descritos na Tabela V.4. Foram avaliados quatro modelos de aprendizado de máquina: LoR, DTC, RFC e LiR.

Tabela V.4: Acurácia e F1-Macro para os modelos preditivos no *workflow* NMFSt.P

Classes	*Baseline*		LoR		DTC		RFC		LiR	
	Acc	F1-Macro	Acc	F1-Macro	Acc	F1-Macro	Acc	F1-Macro	Acc	F1-Macro
3	0,3322	0,1662	0,6119	0,5826	0,9441	0,9439	**0,9615**	**0,9615**	0,7413	0,7001
5	0,1993	0,0665	0,4476	0,4174	0,7972	0,7977	**0,8357**	**0,8346**	0,4371	0,3333
7	0,1399	0,0351	0,4476	0,4088	0,7203	0,7195	**0,7657**	**0,7659**	0,3252	0,2196
10	0,0979	0,0178	0,3636	0,3002	0,5839	0,5825	**0,7028**	**0,7036**	0,2517	0,1629
15	0,0664	0,0083	0,3042	0,2688	0,5175	0,5168	**0,5629**	**0,5619**	0,1783	0,1103

Os resultados na Tabela V.4 mostram que todos os modelos superam o *baseline*. RFC obteve o maior desempenho preditivo em todos os cenários, com acurácia de 96,15% para 3 classes e 56,29% para 15 classes. O classificador DTC obteve desempenho próximo em vários cenários, registrando acurácia de 94,41% (3 classes) e 51,75% (15 classes).

O vetor de características estáticas de microarquitetura forneceu aos modelos informações físicas detalhadas sobre as capacidades computacionais relativas entre o *Cluster* A e o *Cluster* B. O modelo LiR obteve desempenho moderado a baixo nas classes maiores (*e.g.* acurácia de 17,83% para 15 classes), o que sugere dificuldade de modelar linearmente relações que possuem saturação e não linearidade.

A Figura V.6 apresenta os valores de importância de atributos derivados do modelo DTC treinado no cenário de 5 classes. Embora o RFC apresente melhor desempenho preditivo em vários cenários, o DTC foi utilizado nessa análise por oferecer uma interpretação mais direta das decisões. Para manter a interpretabilidade das características, as variáveis *dummy* geradas pela codificação *one-hot* (como as diferentes arquiteturas de CPU) foram reagrupadas de volta para seus atributos originais. Os resultados indicam que as métricas de carga de trabalho por *thread* aparecem entre os principais fatores associados à variabilidade do tempo de execução: a quantidade de sequências por *thread* (`total_sequencias_per_thread`) possui importância de aproximadamente 26,00%, seguida pelo algoritmo de alinhamento selecionado (`algoritmo`) com 19,76% e pelo volume de dados de

entrada por *thread* (`size_arquivos_per_thread`) com 15,04%. Em seguida, aparecem a quantidade de núcleos físicos (`physical_cores`) com 12,99% e o volume total dos arquivos (`size_arquivos`) com 7,04%.

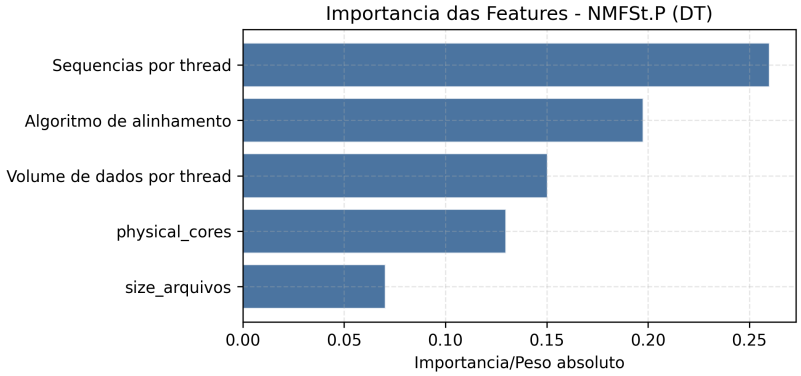
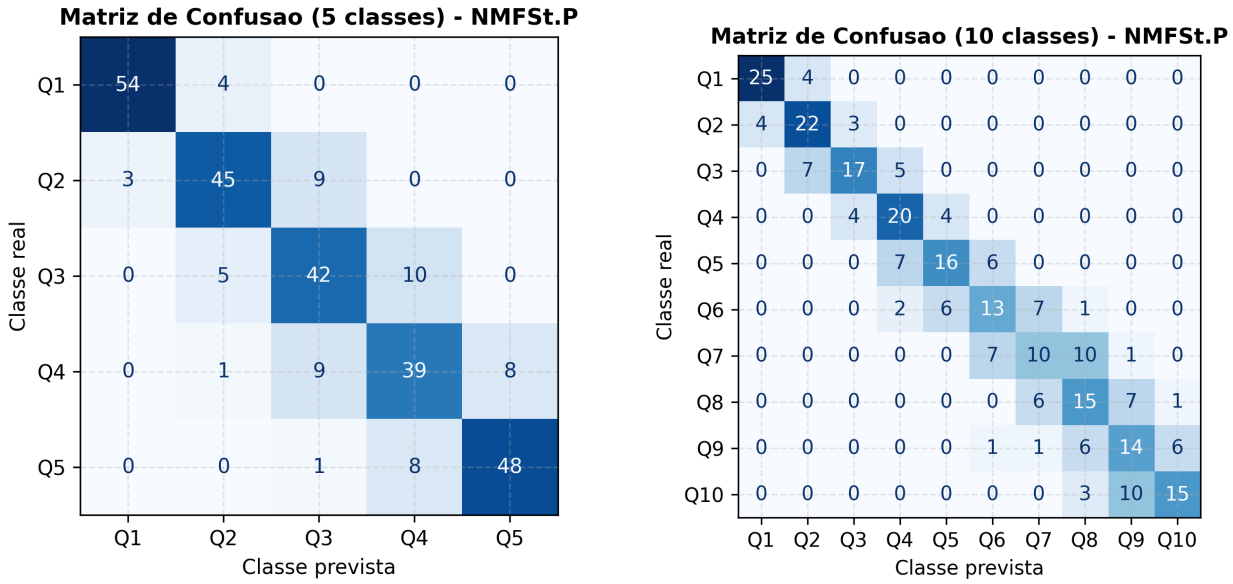


Figura V.6: Atributos mais bem ranqueados de acordo com a importância de Gini no modelo Árvore de Decisão (cenário com cinco classes) no NMFSt.P.

Para avaliar o comportamento preditivo do modelo sob diferentes resoluções de discretização, a Figura V.7 apresenta, de forma comparativa, as matrizes de confusão do classificador DTC nos cenários com 5 classes (Figura V.7a) e 10 classes (Figura V.7b).

A análise das matrizes revela que, em ambos os cenários, os erros de predição concentram-se predominantemente em classes imediatamente adjacentes à diagonal principal (comportamento conhecido como erro suave ou *off-by-one*). No cenário com 5 classes, por exemplo, das 57 execuções reais pertencentes à classe Q2, 45 foram corretamente preditas como Q2, enquanto os erros limitaram-se a 3 instâncias classificadas como Q1 e 9 instâncias como Q3. O mesmo comportamento é observado no cenário mais granular de 10 classes: para a classe Q8, das 29 instâncias de teste, 15 foram classificadas corretamente, 6 foram classificadas como Q7, 7 como Q9 e 1 como Q10.

Essa característica de adjacência indica menor distância entre as classes preditas e observadas na ordenação dos quantis, mas não demonstra, por si só, a redução de erros severos de planejamento. Como as faixas possuem larguras temporais distintas, o benefício operacional desse comportamento depende dos limites dos intervalos e da decisão apoiada pela previsão, aspectos que não foram avaliados diretamente neste estudo. A taxa de acerto é elevada nas classes extremas no cenário de 5 classes, em que as classes de menor tempo (Q1) e maior tempo (Q5) registraram acertos em 54 de 58 e 48 de 57 amostras, respectivamente. No cenário com 10 classes, a classe Q1 registrou 25 acertos em 29 amostras, enquanto a classe Q10 registrou 15 acertos em 28 amostras, com seus erros concentrados nas faixas temporais superiores próximas (Q8 e Q9), preservando uma concentração de erros nas regiões vizinhas da discretização.



(a) Cenário com 5 classes.

(b) Cenário com 10 classes.

Figura V.7: Matrizes de confusão para o classificador Árvore de Decisão no *workflow* NMFSt.P.

V.3.2 Resultados para o *Workflow* Montage

Os resultados preditivos para o *workflow* Montage estão consolidados na Tabela V.5. Foram avaliados os mesmos quatro modelos preditivos e o *baseline* sob o mesmo protocolo experimental.

Tabela V.5: Acurácia e F1-Macro para os modelos preditivos no *workflow* Montage

Classes	*Baseline*		LoR		DTC		RFC		LiR	
	Acc	F1-Macro	Acc	F1-Macro	Acc	F1-Macro	Acc	F1-Macro	Acc	F1-Macro
3	0,3321	0,1662	0,7022	0,7046	0,6931	0,6924	**0,8032**	**0,8016**	0,7130	0,7016
5	0,2004	0,0668	0,5415	0,5247	0,5668	0,5678	**0,6733**	**0,6643**	0,5289	0,4695
7	0,1426	0,0357	0,4242	0,3927	0,4639	0,4631	**0,5343**	**0,5218**	0,4079	0,3417
10	0,0993	0,0181	0,3556	0,3214	0,3791	0,3796	**0,4747**	**0,4622**	0,2852	0,2233
15	0,0668	0,0083	0,2347	0,1884	0,2816	0,2820	**0,3646**	**0,3555**	0,2166	0,1589

Os resultados na Tabela V.5 mostram que os modelos preditivos superam o *baseline* em todos os cenários de discretização. RFC obteve o melhor desempenho em todos os cenários, alcançando 80,32% de acurácia com 3 classes e 36,46% com 15 classes. O classificador DTC apresentou desempenho intermediário, com acurácia de 69,31% para 3 classes e 28,16% para 15 classes, mantendo o apelo de interpretabilidade das decisões. O modelo LiR também superou o *baseline* em todos os cenários (*e.g.*, obtendo acurácia de 71,30% para 3 classes), embora com menor precisão em relação aos modelos baseados em árvores.

Para viabilizar uma análise interpretável sobre a importância dos atributos no Montage, as variáveis *dummy* resultantes da codificação *one-hot* foram reagrupadas de volta em seus respectivos atributos originais. No modelo DTC sob o cenário de 5 classes (Figura V.8), as importâncias

consolidadas indicam que as variáveis mais relevantes na predição do tempo são o número de arquivos por *thread* `files_per_thread` (aproximadamente 21,76%), a capacidade física de memória RAM `ram_gb` (15,44%), o sistema de coordenadas astronômicas `system` (14,91%), o algoritmo de contraste astronômico `stretch` (7,37%) e a quantidade de *threads* (6,88%).

O reagrupamento das variáveis *dummy* permitiu observar que os metadados específicos do domínio (astronomia) e a quantidade de trabalho por *thread* aparecem entre os principais fatores associados à variabilidade do tempo no Montage.

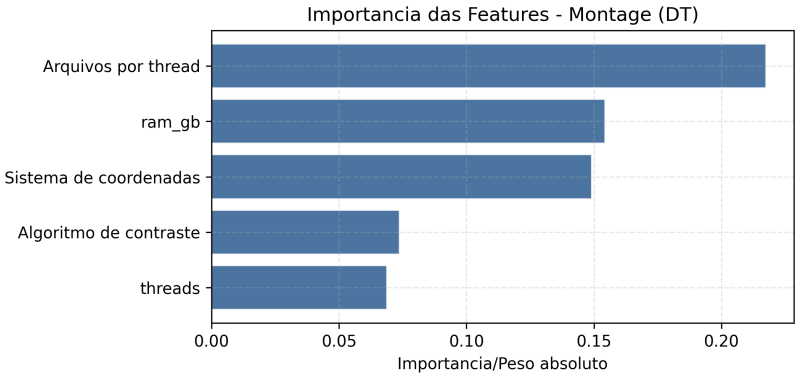


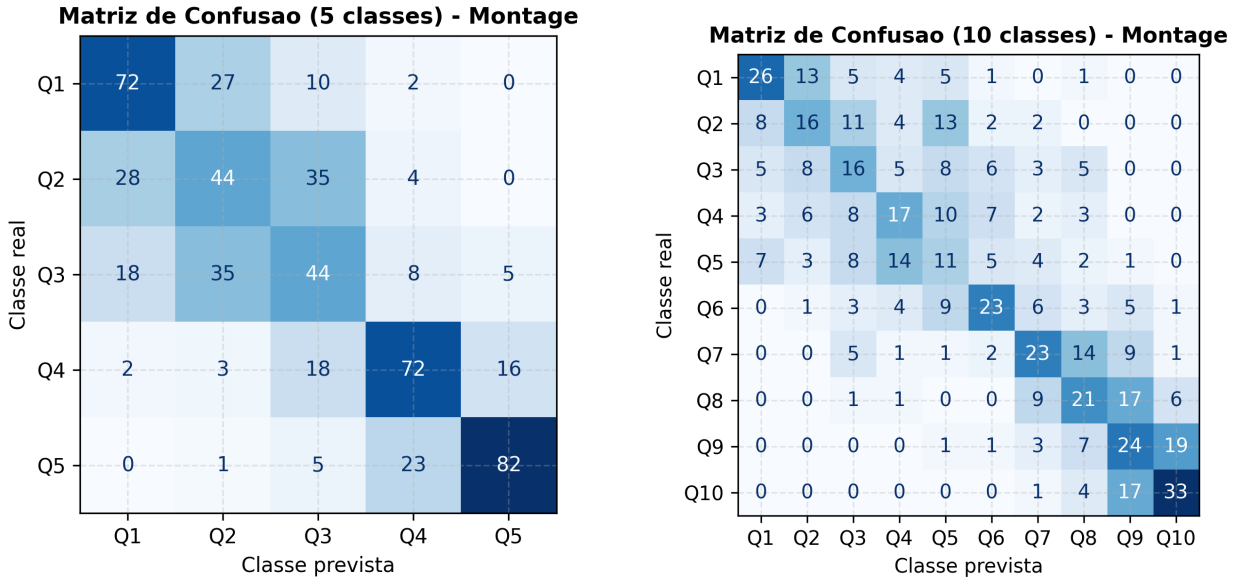
Figura V.8: Atributos mais bem ranqueados de acordo com a importância de Gini no modelo Árvore de Decisão (cenário com cinco classes) no Montage.

Para analisar a distribuição de acertos e erros do modelo preditivo sob diferentes níveis de granularidade no Montage, a Figura V.9 exibe as matrizes de confusão obtidas para o classificador Árvore de Decisão nos cenários de 5 classes (Figura V.9a) e 10 classes (Figura V.9b), também com os eixos rotulados em português.

O Montage, por se caracterizar como um *workflow* intensivo em dados executado em infraestruturas concorrentes compartilhadas, apresenta maior dispersão nos erros de classificação em relação ao NMFSt.P, o que reflete a volatilidade intrínseca dos tempos de transferência de rede e disco. No entanto, mesmo sob essa maior variabilidade, parte relevante das previsões incorretas concentra-se nas vizinhanças da diagonal principal. No cenário de 5 classes, as classes extremas exibem maior estabilidade relativa: a classe Q1 registrou acerto em 72 das 111 amostras (64,9% de *recall*) e a classe crítica Q5 obteve acerto em 82 das 111 amostras (73,9% de *recall*). Entre os erros de Q5, a maior parcela foi deslocada para Q4 (23 casos), com poucos erros adicionais para Q3 (5 casos) e Q2 (1 caso). O maior índice de confusão ocorre nas classes intermediárias (Q2, Q3 e Q4), onde a sobreposição das faixas de tempo é acentuada.

No cenário de 10 classes (Figura V.9b), essa dispersão intermediária torna-se ainda mais evidente devido ao estreitamento das faixas de quantis, com a taxa de acerto exato caindo significativamente (*e.g.* classe Q4 registrando 30,4% de *recall*). Entretanto, o comportamento de erro suave permanece em parte preservado nas classes superiores: para a classe mais longa Q10, das 55 execuções de

teste, 33 foram corretamente classificadas como Q10, 17 como Q9, 4 como Q8 e 1 como Q7, sem deslocamentos para classes baixas. Esse resultado indica proximidade na ordenação das classes, mas não permite concluir que o erro temporal seja pequeno ou que haja redução do risco de interrupção por estouro do limite de tempo nos agendadores HPC. A verificação desse benefício potencial exige considerar a largura das faixas, a regra utilizada para definir o limite solicitado e a frequência de execuções que ultrapassariam esse limite.



(a) Cenário com 5 classes.

(b) Cenário com 10 classes.

Figura V.9: Matrizes de confusão para o classificador Árvore de Decisão no *workflow* Montage.

Como análise complementar do modelo base, avaliou-se o desempenho do melhor classificador no teste agregado (*Cluster* A + *Cluster* B) quando o mesmo conjunto de teste é particionado por infraestrutura. Essa verificação é relevante porque o modelo base é treinado com dados de ambas as infraestruturas e também avaliado originalmente sobre um teste combinado. Assim, a métrica agregada pode mascarar assimetrias de desempenho entre os *clusters*, sobretudo quando a quantidade de amostras de teste é desbalanceada.

Os resultados da Tabela V.6 mostram que, no NMFSt.P, o desempenho do modelo base permanece consistente quando o teste é particionado por infraestrutura. Como os subconjuntos de teste do *Cluster* A e do *Cluster* B possuem tamanhos próximos, a métrica agregada é menos sujeita a ser dominada por uma única infraestrutura. O RFC mantém desempenho elevado no cenário de 3 classes em ambos os *clusters* e, nos cenários mais granulares, apresenta F1-Macro relativamente próximo entre as duas infraestruturas. Em 10 classes, por exemplo, o *Cluster* A alcança F1-Macro de 0,6967 e o *Cluster* B atinge 0,6446, sugerindo que o vetor de características de microarquitetura contribui para capturar diferenças estruturais entre os ambientes sem depender de um único *cluster* dominante.

Tabela V.6: Desempenho do melhor modelo base no teste agregado e nos subconjuntos de teste por infraestrutura.

Workflow	Classes	Modelo	Geral A+B		*Cluster* A			*Cluster* B		
			Acc	F1-Macro	n	Acc	F1-Macro	n	Acc	F1-Macro
NMFSt.P	3	RFC	0,9615	0,9615	132	0,9697	0,8711	154	0,9545	0,9299
NMFSt.P	5	RFC	0,8357	0,8346	139	0,8058	0,7934	147	0,8639	0,7788
NMFSt.P	7	RFC	0,7657	0,7659	138	0,7609	0,6756	148	0,7703	0,6920
NMFSt.P	10	RFC	0,7028	0,7036	135	0,6815	0,6967	151	0,7219	0,6446
NMFSt.P	15	RFC	0,5629	0,5619	141	0,5319	0,5050	145	0,5931	0,4889
Montage	3	RFC	0,8032	0,8016	159	0,8868	0,6582	395	0,7696	0,7657
Montage	5	RFC	0,6733	0,6643	147	0,7347	0,2731	407	0,6511	0,5287
Montage	7	RFC	0,5343	0,5218	154	0,5779	0,2545	400	0,5175	0,4112
Montage	10	RFC	0,4747	0,4622	160	0,4562	0,1988	394	0,4822	0,4381
Montage	15	RFC	0,3646	0,3555	150	0,3600	0,2489	404	0,3663	0,3021

No Montage, a diferença entre acurácia e F1-Macro é mais evidente. O *Cluster* A apresenta acurácia superior à do *Cluster* B nos cenários de 3, 5 e 7 classes, mas seu F1-Macro é substancialmente menor em todos os cenários avaliados. Nos cenários de 10 e 15 classes, o *Cluster* B também supera ligeiramente o *Cluster* A em acurácia. Esse padrão indica que a acurácia isolada superestima o desempenho no *Cluster* A, pois os acertos estão concentrados em poucas classes e o classificador perde capacidade de distinguir todas as faixas temporais de maneira balanceada. Portanto, a análise por infraestrutura reforça a necessidade de reportar F1-Macro juntamente com acurácia e evidencia que o desempenho agregado do modelo base não descreve integralmente a robustez do preditor em cada ambiente de execução.

V.3.3 Generalização Trans-Ambiente

Com o propósito de avaliar o comportamento e a robustez dos modelos preditivos treinados quando submetidos a um cenário de transferência fora de domínio (*out-of-distribution*), realizou-se uma avaliação experimental utilizando um ambiente computacional externo e de arquitetura distinta. Essa avaliação de generalização trans-ambiente permite analisar a capacidade dos modelos preditivos em manter desempenho quando aplicados a um ambiente computacional distinto daquele utilizado durante o treinamento. Essa análise é relevante porque, em cenários reais, modelos treinados para estimar o tempo de execuções de *workflows* científicos podem ser aplicados a infraestruturas com características diferentes das observadas nos dados de treinamento.

É importante destacar também, que treinar o modelo no *cluster* A e testar no *cluster* B, e vice-versa, faz com que os atributos relacionados ao domínio da infraestrutura computacional apresentem pouca ou nenhuma variabilidade nos dados de treinamento. Dessa forma, o modelo não dispõe de exemplos suficientes para aprender de maneira robusta como atributos da infraestrutura

computacional influenciam no tempo de execução dos *workflows*. Portanto, a avaliação trans-ambiente foi incluída como etapa complementar às avaliações realizadas dentro do mesmo ambiente, permitindo analisar não somente a capacidade preditiva dos modelos sob condições conhecidas, mas também sua capacidade de adaptação quando submetidos a uma infraestrutura computacional de arquitetura distinta.

O ambiente computacional selecionado para essa validação, denominado *Host* C, consiste em um computador pessoal equipado com 16 GB de RAM, operando sob sistema operacional Ubuntu 22.04 em modo WSL2. Por não constituir uma infraestrutura de HPC, o *Host* C foi utilizado como ambiente externo para analisar a portabilidade dos modelos diante de uma mudança acentuada de infraestrutura, e não como evidência de generalização para um terceiro *cluster* de produção. Foram coletadas 225 amostras de execuções do *workflow* Montage no *Host* C e discretizadas adotando-se os mesmos quantis calculados sobre o conjunto de dados do *cluster* de treinamento (*Cluster* A e *Cluster* B).

Para este ambiente computacional de validação externa, o vetor de características de microarquitetura é mapeado com os seguintes atributos físicos: o modelo do processador (`cpu_model` = *AMD Ryzen 5 1600X*), o número de núcleos físicos (`physical_cores` = 6) e lógicos (`logical_cores` = 12), o tamanho das memórias caches L2 (`l2_cache_kb` = 3.072 KB) e L3 (`l3_cache_kb` = 8.192 KB) e a capacidade de memória principal alocada para o ambiente virtual (`ram_gb` = 7,73 GB), com os canais e a velocidade da RAM não especificados no vetor (`ram_channels` = 0 e `ram_speed` = 0). Paralelamente, sob a abordagem de representação secundária, foram mapeadas as respectivas pontuações de *benchmark* do *host*, as quais totalizam: `cpu_points = 2189` para o desempenho de *thread* única, `disk_points = 1230` para a taxa de escrita sequencial em disco (MB/s) e `ram_points = 6526` correspondente à largura de banda da memória RAM (MB/s).

Investigaram-se duas abordagens para a modelagem dos metadados de *hardware* nas variáveis de aprendizado de máquina. A primeira abordagem é a representação por vetor de características, onde o *hardware* é descrito por um vetor de atributos físicos detalhados do processador e barramento, abrangendo quantidade de núcleos físicos e lógicos, tamanhos das memórias caches L2 e L3, capacidade real de memória principal, quantidade física de canais de memória ativos e velocidade nominal da RAM. Trata-se da modelagem principal proposta para o `AranduFlow`. A segunda abordagem é a representação por pontuações de *benchmark*, em que o *hardware* das máquinas de treinamento é representado por escores brutos agregados obtidos a partir de bancos de dados internacionais e testes sintéticos de barramento (`cpu_points`, `disk_points`, `ram_points`). Trata-se da representação de comparação descrita na Seção V.1.1

A Tabela V.7 apresenta de forma comparativa o desempenho preditivo (Acurácia e F1-Macro) dos modelos nos cenários com 3, 5, 7, 10 e 15 classes sob as duas formas de representação, para o

workflow Montage.

Tabela V.7: Comparação de Acurácia e F1-Macro no *Host* C usando as representações por Pontuações de *benchmark* vs. vetor de características

Classes	Modelo	Acurácia		Diff. Acurácia	F1-Macro		Diff. F1-Macro
		Bench.	Vetor		Bench.	Vetor	
3	Regressão Logística	0,5333	0,3644	-0,1689	0,4300	0,2163	-0,2138
	Random Forest	0,3689	0,3689	0,0000	0,1879	0,1879	0,0000
	Árvore de Decisão	0,3689	0,3733	+0,0044	0,1937	0,1969	+0,0032
	Regressão Linear	0,3467	0,3600	+0,0133	0,1716	0,1765	+0,0049
5	Regressão Logística	0,0356	0,2400	+0,2044	0,0372	0,0900	+0,0528
	Random Forest	0,2756	0,2667	-0,0089	0,1080	0,1052	-0,0029
	Árvore de Decisão	0,2756	0,2667	-0,0089	0,1226	0,1114	-0,0112
	Regressão Linear	0,2356	0,2444	+0,0089	0,0824	0,0818	-0,0006
7	Regressão Logística	0,0444	0,2222	+0,1778	0,0122	0,1042	+0,0920
	Random Forest	0,2578	0,2622	+0,0044	0,1412	0,1479	+0,0066
	Árvore de Decisão	0,2578	0,2622	+0,0044	0,1297	0,1324	+0,0028
	Regressão Linear	0,2489	0,2222	-0,0267	0,0998	0,0978	-0,0020
10	Regressão Logística	0,0311	0,1644	+0,1333	0,0254	0,0729	+0,0475
	Random Forest	0,2133	0,2222	+0,0089	0,0979	0,1364	+0,0385
	Árvore de Decisão	0,2178	0,2089	-0,0089	0,0931	0,0857	-0,0074
	Regressão Linear	0,1956	0,2089	+0,0133	0,0658	0,0660	+0,0002
15	Regressão Logística	0,0000	0,0667	+0,0667	0,0000	0,0359	+0,0359
	Random Forest	0,2044	0,2089	+0,0044	0,0885	0,0852	-0,0033
	Árvore de Decisão	0,1867	0,1911	+0,0044	0,0843	0,0828	-0,0015
	Regressão Linear	0,1733	0,1911	+0,0178	0,0642	0,0628	-0,0014

Os resultados comparativos sugerem que o enriquecimento do espaço de atributos com o vetor de características físicas estáticas apresenta indícios de maior estabilidade na generalização trans-infraestrutura, especialmente em cenários mais granulares. Sob a representação baseada em pontuações de *benchmark*, o modelo de LoR aparenta um desempenho artificialmente alto no cenário de 3 classes (53,33% de acurácia), porém colapsa nos cenários mais refinados de 5, 7, 10 e 15 classes, alcançando acurácias próximas a zero e escores F1-Macro também próximos de zero devido ao desvio de classes severo. Ao adotar a representação por vetor de características estáticas de microarquitetura, o modelo LoR passa a demonstrar um comportamento preditivo mais estável, alcançando acurácias como 24,00% para 5 classes (um ganho líquido de +20,44%), 22,22% para 7 classes (+17,78%) e 16,44% para 10 classes (+13,33%), além de ganhos consistentes no F1-Macro. Isso indica que as fronteiras lineares se beneficiam diretamente de dados físicos mais finos para descrever a diferença estrutural entre as CPUs Intel do treinamento e a CPU AMD do teste.

Para os demais modelos, os efeitos da representação estática foram mais modestos e dependentes do cenário. No classificador RFC, a representação estática propiciou ganho de 0,89 ponto percentual em acurácia e incremento de 3,85 pontos percentuais no F1-Macro no cenário de 10 classes,

embora apresente diferenças pequenas ou negativas em outros cenários. O modelo DTC apresentou melhorias discretas em 3, 7 e 15 classes, mas perdas em 5 e 10 classes. A LiR demonstrou ganhos de acurácia em 3, 5, 10 e 15 classes, ainda que com F1-Macro baixo nos cenários mais granulares.

Entretanto, nota-se que o desempenho obtido no *Host* C é substancialmente menor se comparado ao obtido na avaliação apenas com os *clusters* A e B (Tabela V.5). Essa queda de desempenho é compatível com a ocorrência de mudanças nas distribuições dos atributos e dos rótulos entre os ambientes de treinamento e teste, fenômenos associados a *covariate shift* e *label shift*. Ao discretizar os dados de execução coletados no *Host* C utilizando as fronteiras de quantis estimadas sobre os dados de treinamento (*Cluster* A e *Cluster* B), observou-se que a distribuição de rótulos real do teste ficou severamente desbalanceada. As diferenças de *hardware*, sistema operacional, virtualização e carga computacional entre o *Host* C e os nós de HPC podem ter contribuído para que mais de 65% das execuções do ambiente externo se concentrassem em apenas um dos quantis calculados no treinamento. Assim, o classificador balanceado errou a maior parte das amostras, revelando os limites operacionais de modelos preditivos fixos e fundamentando a investigação de estratégias de calibração local.

Avaliação Preditiva sob Calibração Dinâmica Local no *Host* C

Conforme discutido na Seção V.3.3, a transposição direta do preditor global para o *Host* C sem calibração resulta em degradação acentuada de desempenho. Para mensurar experimentalmente a viabilidade e os ganhos da extensão de calibração dinâmica local descrita na Seção V.1.5, executaram-se os testes utilizando um conjunto de sondagem contendo $N_{cal} = 15$ execuções coletadas localmente no *Host* C. O restante das execuções locais serviu como conjunto de teste para avaliação do preditor calibrado.

A Tabela V.8 apresenta os resultados comparativos de acurácia obtidos para as diferentes estratégias de calibração local aplicadas a todos os algoritmos avaliados. Adicionalmente, para analisar o comportamento dos modelos sob o desbalanceamento inerente às classes temporais, a Tabela V.9 expõe os resultados obtidos para a métrica F1-Macro.

Para complementar a análise agregada das Tabelas V.8 e V.9, a Figura V.10 apresenta as matrizes de confusão da Árvore de Decisão no cenário com 3 classes para as duas estratégias de calibração local avaliadas. Esse modelo foi selecionado para a visualização por apresentar o maior F1-Macro após o ajuste fino ponderado no cenário de 3 classes e por tornar mais evidente a diferença entre uma calibração probabilística conservadora e uma readaptação efetiva dos parâmetros do classificador.

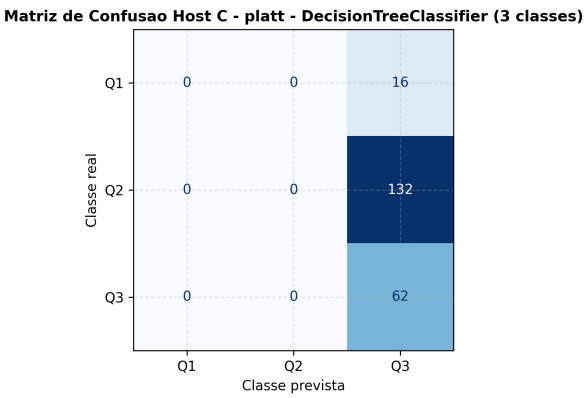
As matrizes indicam que a Calibração de Platt não foi capaz de recuperar a separação entre as faixas temporais no *Host* C. No cenário com 3 classes, todas as amostras de teste foram deslocadas

Tabela V.8: Comparação de Acurácia obtida no *Host* C sob as diferentes estratégias de calibração local.

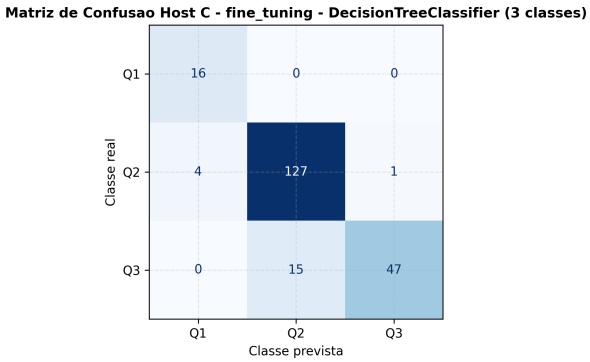
Classes	Modelo	Sem Calibração	Calibração de Platt	Ajuste Fino Ponderado
3	Regressão Logística	0,3644	0,2952	0,4476
	Random Forest	0,3689	0,3000	**0,9048**
	Árvore de Decisão	0,3733	0,2952	**0,9048**
	Regressão Linear/Ridge	0,3600	–	0,8524
5	Regressão Logística	0,2400	0,1905	0,2381
	Random Forest	0,2667	0,1905	**0,7857**
	Árvore de Decisão	0,2667	0,1905	0,6524
	Regressão Linear/Ridge	0,2444	–	0,6667
10	Regressão Logística	0,1644	0,1857	0,1952
	Random Forest	0,2222	0,2286	**0,6381**
	Árvore de Decisão	0,2089	0,2190	0,6333
	Regressão Linear/Ridge	0,2089	–	0,4048

Tabela V.9: Comparação de F1-Macro obtido no *Host* C sob as diferentes estratégias de calibração local.

Classes	Modelo	Sem Calibração	Calibração de Platt	Ajuste Fino Ponderado
3	Regressão Logística	0,2163	0,1520	0,5135
	Random Forest	0,1879	0,1575	0,8810
	Árvore de Decisão	0,1969	0,1520	**0,8901**
	Regressão Linear/Ridge	0,1765	–	0,6221
5	Regressão Logística	0,0900	0,0640	0,1567
	Random Forest	0,1052	0,0640	**0,5904**
	Árvore de Decisão	0,1114	0,0640	0,4853
	Regressão Linear/Ridge	0,0818	–	0,4371
10	Regressão Logística	0,0729	0,0473	0,0549
	Random Forest	0,1364	0,0594	**0,3101**
	Árvore de Decisão	0,0857	0,0544	0,2922
	Regressão Linear/Ridge	0,0660	–	0,2167



(a) Calibração de Platt.



(b) Ajuste fino ponderado.

Figura V.10: Matrizes de confusão da Árvore de Decisão calibrada no *Host* C para o cenário com três classes.

para a classe Q3, resultando em acerto apenas das 62 execuções originalmente pertencentes a essa faixa e erro sistemático nas classes Q1 e Q2. No cenário com 5 classes, a camada de Platt também concentrou as predições em Q5. Em 10 classes, a concentração ocorreu principalmente nas classes Q6 e Q10. Esse padrão explica o baixo F1-Macro da estratégia, pois a métrica penaliza fortemente modelos que ignoram classes inteiras.

Por outro lado, o Ajuste Fino Ponderado produziu uma matriz de confusão substancialmente mais concentrada na diagonal principal. No cenário com 3 classes, a classe Q1 foi inteiramente recuperada (16 acertos em 16 amostras), enquanto Q2 apresentou 127 acertos em 132 amostras e Q3 apresentou 47 acertos em 62 amostras. Nos cenários mais granulares, o modelo ainda apresenta confusões, mas os erros passam a ocorrer de forma mais informativa do que no colapso observado em Platt. Portanto, a análise por matriz de confusão reforça que o ganho do ajuste fino ponderado não decorre apenas de aumento global de acurácia, mas de uma redistribuição mais útil das predições entre as faixas temporais do *Host* C.

A análise comparativa indica que o Ajuste Fino Ponderado (*Weighted Fine-Tuning*) apresentou melhorias expressivas em relação às demais abordagens nos cenários avaliados. No cenário com 3 classes, a acurácia da Árvore de Decisão subiu de 0,3733 (sem calibração) para 0,9048 com a aplicação do ajuste fino ponderado, enquanto o F1-Macro passou de 0,1969 para **0,8901**. O RFC apresentou comportamento semelhante no mesmo cenário, também atingindo 0,9048 de acurácia. Em cenários com maior granularidade, o melhor desempenho passou a ser obtido pelo RFC ajustado: 0,7857 de acurácia e **0,5904** de F1-Macro em 5 classes, e 0,6381 de acurácia e **0,3101** de F1-Macro em 10 classes.

Esse comportamento superior decorre da capacidade do método de reajustar os parâmetros internos (coeficientes lineares ou caminhos de decisão das árvores) sintonizando-os à escala de tempo real do *host* de destino. A ponderação assimétrica ($\lambda_{destino} = 5,0$) faz com que a otimização priorize as amostras do *Host* C, enquanto a manutenção dos dados históricos com peso reduzido ($\lambda_{origem} = 0,1$) busca preservar parte dos padrões aprendidos nos ambientes de origem.

A estratégia de Calibração de Platt local não se mostrou uma adaptação efetiva na avaliação realizada. Nos cenários de 3 e 5 classes, ela reduziu a acurácia e o F1-Macro dos classificadores em relação ao uso sem calibração; no caso da Árvore de Decisão, por exemplo, a acurácia em 3 classes passou de 0,3733 para 0,2952. Mesmo nos casos em que houve pequeno ganho de acurácia em 10 classes, o F1-Macro permaneceu baixo porque as predições ficaram concentradas em poucas classes. Esse resultado indica que, com apenas 15 amostras locais, a camada sigmoidal de Platt atua como uma correção probabilística conservadora, mas não corrige o deslocamento estrutural de distribuição (*covariate shift*) entre os *clusters* A/B e o *Host* C, nem altera a estrutura de decisão aprendida no modelo base. Essa limitação reforça a relevância do Ajuste Fino Ponderado, que

efetivamente reajusta os parâmetros internos do modelo para a escala temporal do *host* de destino. Para a Regressão Linear/Ridge, a Calibração de Platt não é aplicável (indicada por – nas tabelas), por tratar-se de um regressor e não de um classificador probabilístico.

Os resultados consolidados nas Tabelas V.8 e V.9 reforçam que o ganho de desempenho no *Host* C está associado principalmente ao ajuste fino ponderado. Em comparação com o uso sem calibração e com a calibração de Platt, essa estratégia apresentou os melhores valores de acurácia e F1-Macro nos cenários avaliados, especialmente no cenário com 3 classes. Nos cenários com maior granularidade, os ganhos permanecem mais limitados, mas ainda indicam que o reajuste supervisionado com amostras locais é efetivo, nos experimentos realizados.

Em suma, os resultados sugerem que a utilização de 15 execuções locais para calibrar o preditor proporcionou melhorias expressivas de acurácia no ambiente avaliado. Contudo, a validação efetuada no *Host* C constitui uma demonstração experimental preliminar, restrita ao *workflow* Montage, sendo necessários testes adicionais em outras infraestruturas e aplicações para examinar a generalização da estratégia.

Para justificar cientificamente a escolha do parâmetro de calibração $N_{cal} = 15$, realizou-se um estudo de sensibilidade variando-se o tamanho do conjunto de sondagem local no *Host* C em $N_{cal} \in \{2, 5, 10, 15, 20, 25, 30, 40, 50\}$. Para cada valor de N_{cal}, foram executadas 30 rodadas de simulação utilizando sementes de divisão aleatória estratificada, permitindo mensurar tanto o desempenho médio (acurácia) quanto a variabilidade (desvio padrão) decorrentes da escolha das amostras de calibração. A Figura V.11 ilustra os resultados desse estudo de sensibilidade para cenários de discretização com 3, 5 e 10 classes, comparando-os com as respectivas acurácias da *baseline* sem calibração (linhas tracejadas).

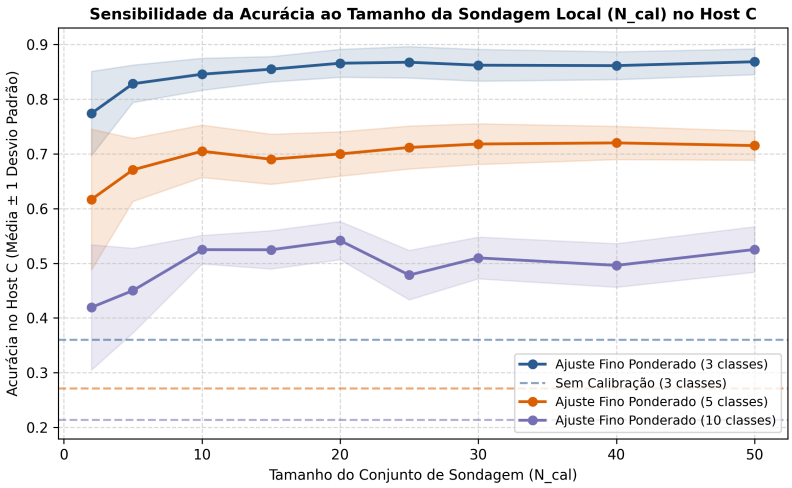


Figura V.11: Sensibilidade da acurácia média e desvio padrão em função do tamanho do conjunto de sondagem local (N_{cal}) para 3, 5 e 10 classes no *Host* C.

Como pode ser observado na Figura V.11, a acurácia do modelo calibrado apresenta um com-

portamento de curva de aprendizado característico, no qual se destaca o efeito cotovelo (*elbow effect*) em torno de $N_{cal} = 15$. No cenário de 3 classes, a acurácia média salta rapidamente de 36,00% (sem calibração) para 77,41% com apenas $N_{cal} = 2$ amostras, progredindo para 82,83% com $N_{cal} = 5$, 84,57% com $N_{cal} = 10$, e estabilizando-se em 85,49% com $N_{cal} = 15$. A partir de $N_{cal} = 15$, o ganho de desempenho obtido com o aumento do conjunto de sondagem torna-se marginal e entra em patamar de saturação, oscilando dentro da margem de erro amostral (por exemplo, atingindo 86,58% com $N_{cal} = 20$, o que representa uma melhora menor que 1,1% frente ao custo extra de execução). O mesmo comportamento se repete para os cenários de 5 e 10 classes, onde a estabilização ocorre em torno de 70,00% e 52,50%, respectivamente, a partir de $N_{cal} = 15$.

O valor médio de 85,49% não contradiz a acurácia de 90,48% apresentada anteriormente. O primeiro corresponde à média de 30 rodadas com diferentes amostras de calibração, enquanto o segundo foi obtido na divisão fixa utilizada na comparação detalhada entre os métodos. A diferença evidencia a influência da composição da pequena amostra local sobre o desempenho da calibração.

No contexto da extensão experimental avaliada, a definição de $N_{cal} = 15$ representa um equilíbrio entre a quantidade de execuções de sondagem e o desempenho preditivo observado. Aumentar o conjunto para $N_{cal} \geq 20$ ou $N_{cal} \geq 30$ exigiria mais execuções do Montage na infraestrutura de destino, enquanto os resultados da análise de sensibilidade indicam ganhos adicionais reduzidos de acurácia.

V.4 Síntese da Avaliação

A comparação dos resultados obtidos nos dois estudos de caso fornece evidências sobre a aplicabilidade da abordagem preditiva do `AranduFlow` a *workflows* heterogêneos executados em ambientes de computação de alto desempenho.

A natureza das duas aplicações avaliadas é distinta. O NMFSt.P é um *workflow* de bioinformática com característica majoritariamente *CPU-bound*, que processa arquivos de sequências e árvores em `Newick` e exige processamento na comparação de árvores filogenéticas. O Montage, por sua vez, é um *workflow* astronômico de processamento de imagens com perfil intensivo em operações de E/S, envolvendo etapas de reprojeção e coadição das imagens que demandam leitura e escrita frequentes de arquivos.

Apesar dessas diferenças em termos de domínio científico, volume de dados e perfil de utilização dos recursos, os dois *workflows* foram integrados por meio da mesma arquitetura modular e do fluxo comum descrito no Capítulo IV. Em ambos os casos, o `AranduFlow` recebeu a configuração estruturada em JSON, construiu o vetor de atributos, produziu a estimativa temporal, submeteu a execução à infraestrutura selecionada e registrou o tempo posteriormente observado. Permaneceram específicos de cada aplicação o esquema de configuração, o processo de engenharia de atributos, o

executor e o modelo preditivo, conforme previsto pela arquitetura.

O treinamento e a avaliação dos modelos foram realizados pelo procedimento experimental descrito neste capítulo. Os artefatos resultantes foram então incorporados ao Gerenciador de Predição e Persistência para produzir as estimativas apresentadas pelo *gateway*. Em ambos os estudos de caso, os classificadores baseados em árvores superaram o *baseline* operacional na predição das faixas de tempo. Esses resultados indicam que o fluxo arquitetural comum pode ser reutilizado em aplicações com características distintas, desde que sejam implementados os elementos específicos exigidos por cada *workflow*.

Os resultados evidenciam um compromisso entre o desempenho classificatório e a resolução temporal das previsões. Cenários com menor número de classes apresentam maior acurácia, mas fornecem faixas mais abrangentes, que podem ser pouco informativas para decisões que exigem maior precisão. Como a discretização por quantis equilibra a frequência das observações, e não a largura temporal dos intervalos, classes adjacentes podem representar diferenças expressivas de duração. Portanto, tanto o acerto da classe quanto a concentração dos erros em classes vizinhas devem ser interpretados em conjunto com os limites temporais correspondentes. A escolha da granularidade não deve se apoiar exclusivamente na maior acurácia, mas também na resolução necessária para a decisão pretendida.

V.5 Limitações e Ameaças à Validade

Apesar dos resultados obtidos, algumas limitações devem ser consideradas na interpretação da avaliação experimental. A primeira refere-se à validade externa: os experimentos foram conduzidos com dois *workflows* científicos e duas classes de infraestrutura, o que fornece evidências de adaptabilidade da arquitetura, mas não permite afirmar generalização irrestrita para qualquer domínio científico, escalonador ou arquitetura de *hardware*. Novos *workflows* podem exigir seleção de atributos, coleta de histórico e retreinamento específicos.

A avaliação também pressupõe que o histórico de treinamento represente adequadamente as configurações encontradas durante o uso do modelo. A análise realizada no *Host* C mostrou que mudanças relevantes na infraestrutura e na distribuição dos tempos podem reduzir substancialmente o desempenho preditivo. Como o protótipo não detecta automaticamente submissões fora da distribuição, os resultados obtidos nos *clusters* de treinamento não devem ser diretamente extrapolados para ambientes ou configurações não representados nos dados históricos.

A segunda diz respeito à interpretação dos atributos. As importâncias baseadas em decréscimo de impureza de Gini são úteis para análise exploratória, mas podem favorecer atributos contínuos ou variáveis categóricas com maior número de divisões possíveis. Por isso, os ranqueamentos apresentados devem ser interpretados como evidências associativas sobre a relevância dos metadados,

e não como demonstrações causais isoladas.

Os dados foram coletados em ambientes compartilhados de produção, sujeitos a concorrência de usuários, variações de carga, contenção de E/S e flutuações de rede. Essas condições aumentam o realismo experimental, mas também introduzem ruído não controlado que pode afetar tanto o treinamento quanto a avaliação dos modelos e a reprodutibilidade dos experimentos.

A avaliação quantitativa concentrou-se no componente preditivo do `AranduFlow`. Embora os demais módulos tenham sido utilizados na configuração, submissão e registro das execuções, não foram avaliados quantitativamente aspectos como usabilidade da interface, sobrecarga introduzida pelo *gateway* ou confiabilidade da comunicação entre seus componentes. A calibração local, por sua vez, foi investigada somente como extensão experimental, utilizando o Montage no *Host* C.

A avaliação realizada não quantificou a utilidade das faixas para decisões como selecionar uma infraestrutura, definir um limite de tempo ou verificar a viabilidade de conclusão dentro de um prazo. Assim, os resultados demonstram a capacidade de classificação das execuções nos cenários estudados, mas não permitem concluir, isoladamente, que a configuração com maior acurácia seja a mais adequada ao planejamento do usuário. A comparação com a regressão direta do tempo e a avaliação orientada à decisão são propostas na Seção VI.2.

Embora o estudo de sensibilidade permita analisar o efeito do tamanho e da composição da amostra de calibração, a seleção pseudoaleatória de 15 execuções não garante representatividade em qualquer ambiente de produção. Os ganhos observados dependem das configurações presentes na amostra local, e o protótipo não implementa uma política automática para selecionar essas execuções.

V.6 Considerações Finais

Este capítulo apresentou o delineamento e os resultados da avaliação experimental da abordagem preditiva integrada ao `AranduFlow`. Os experimentos com o NMFSt.P e o Montage mostraram que os modelos avaliados superaram o *baseline* operacional nas diferentes granularidades de discretização, embora o desempenho tenha diminuído à medida que o número de classes aumentou. A avaliação trans-ambiente evidenciou as limitações da aplicação direta dos modelos a uma infraestrutura não representada no treinamento, enquanto a calibração local apresentou resultados promissores como extensão experimental. Essas evidências fundamentam a resposta à questão de pesquisa e a síntese das contribuições apresentadas no Capítulo VI.

Capítulo VI Conclusão

Esta dissertação apresentou o `AranduFlow`, um *gateway* científico voltado ao suporte à submissão de *workflows* e à predição de tempo de execução em ambientes de HPC. A proposta visa apoiar usuários e administradores de sistemas na tomada de decisão informada antes do disparo de execuções em infraestruturas compartilhadas. Ao utilizar dados históricos de proveniência e técnicas de aprendizado de máquina supervisionado, o sistema estima faixas de tempo de execução, reduzindo a dependência de previsões pontuais sensíveis a ruídos operacionais típicos de *clusters* de produção.

Inicialmente, o Capítulo II estabeleceu os fundamentos teóricos necessários, apresentando o formalismo matemático de *workflows* representados por grafos direcionados acíclicos (DAG), a arquitetura de ambientes HPC multiusuário e a formulação teórica de modelos de aprendizado de máquina lineares (LiR e LoR) e baseados em árvores (DTC e RFC), bem como o detalhamento de métricas de avaliação voltadas a cenários multiclasse, como o F1-Macro.

Em seguida, o Capítulo III mapeou sistematicamente o estado da arte na literatura por meio de revisão sistemática e *snowballing*. Foram definidos critérios formais de inclusão e exclusão para a seleção dos artigos representativos, evidenciando que as soluções existentes abordavam o problema de forma parcial, frequentemente focando em regressão contínua, em métricas exclusivas de infraestrutura ou operando de forma isolada sem acoplamento a portais de submissão.

O Capítulo IV detalhou a especificação arquitetural modular do `AranduFlow`, descrevendo a Interface *Web*, o Gerenciador de Predição e Persistência, o Seletor e Conector de Infraestrutura e os Executores de *Workflows*. O capítulo mostrou como esses módulos integram a configuração, a produção da estimativa temporal, a submissão à infraestrutura selecionada e o registro do tempo posteriormente observado.

O Capítulo V apresentou a metodologia de avaliação experimental baseada em dois estudos de caso reais com perfis contrastantes: o *workflow* NMFSt.P, intensivo em CPU, e o *workflow* Montage, intensivo em E/S. Ambos os *workflows* foram avaliados sob um protocolo experimental que combina *hold-out* aleatório (80% treino e 20% teste) com validação cruzada estratificada de 5 *folds* no conjunto de treinamento, em múltiplos cenários de discretização (3, 5, 7, 10 e 15 classes de quantis de igual frequência) e em nós de especificações físicas distintas (*Cluster* A e *Cluster* B).

Os resultados mostraram que os modelos avaliados superaram o *baseline* operacional nos dois

estudos de caso e em todas as granularidades consideradas. O RFC apresentou o melhor desempenho global, com acurácia entre 96,15% e 56,29% no NMFSt.P e entre 80,32% e 36,46% no Montage, considerando, respectivamente, os cenários com 3 e 15 classes. As matrizes de confusão analisadas também mostraram, em cenários específicos, maior concentração dos erros em faixas temporais próximas, indicando um potencial de apoio ao planejamento cuja utilidade depende da largura das faixas e ainda requer avaliação orientada à decisão. A análise de importância baseada no decréscimo de impureza de Gini indicou que metadados de domínio e medidas de volumetria por *thread* aparecem entre os principais atributos associados à variabilidade do tempo de execução.

A avaliação trans-ambiente realizada com o Montage em um *host* externo mostrou que o desempenho dos modelos diminui quando a infraestrutura de destino não está adequadamente representada pelos dados de treinamento. A representação por características estáticas de microarquitetura apresentou maior estabilidade em alguns dos cenários mais granulares, especialmente para a LoR, embora seus efeitos tenham variado entre modelos e números de classes. Como extensão experimental, investigou-se também a calibração dinâmica local. Com uma amostra fixa de 15 execuções locais, o ajuste fino ponderado elevou a acurácia da Árvore de Decisão de 37,33% para 90,48% no cenário com 3 classes. Na análise de sensibilidade com diferentes amostras de calibração, a acurácia média para 15 execuções foi de 85,49%, indicando o potencial da estratégia no ambiente avaliado. Esses resultados estão restritos ao Montage e ao *host* utilizado, e a calibração não integra o fluxo operacional regular do `AranduFlow`.

Em resposta à questão de pesquisa, os resultados indicam que a submissão e a estimativa de faixas de tempo de execução de *workflows* científicos heterogêneos podem ser integradas por meio de uma arquitetura modular que separa o fluxo comum do *gateway* dos elementos específicos de cada aplicação. No `AranduFlow`, a configuração informada pelo usuário é convertida no vetor $\mathbf{x}$, composto por metadados de domínio e configuração, características dos dados de entrada e informações da infraestrutura e dos recursos solicitados. Esse vetor é processado por um modelo específico para o *workflow*, e a faixa estimada é apresentada antes da confirmação da submissão. A adaptabilidade observada decorre, portanto, da reutilização do fluxo de configuração, predição, submissão e persistência, associada à implementação de esquemas de configuração, processos de engenharia de atributos, executores e modelos próprios para cada aplicação, e não da utilização de um único modelo universal. Como evidência complementar de relevância acadêmica, parte da arquitetura do ecossistema proposto foi descrita em um artigo publicado no *International Conference on Enterprise Information Systems* (ICEIS) 2026 [Pereira et al., 2026], sem substituir a avaliação experimental apresentada nesta dissertação.

VI.1 Limitações

Os resultados devem ser interpretados considerando o escopo da avaliação realizada. Os experimentos envolveram dois *workflows* e duas infraestruturas de HPC, enquanto a avaliação no *Host* C foi conduzida somente com o Montage. Além disso, os dados foram coletados em ambientes compartilhados, sujeitos a variações de carga e contenção de recursos. A avaliação quantitativa concentrou-se no componente preditivo e não abrangeu aspectos como usabilidade da interface, sobrecarga introduzida pelo *gateway*, autenticação, concorrência entre múltiplos usuários ou acompanhamento dos estados das execuções. Dessa forma, os resultados fornecem evidências de viabilidade nos cenários considerados, mas não permitem afirmar a generalização irrestrita da abordagem para outros domínios, escalonadores e infraestruturas.

Outra limitação refere-se ao compromisso entre acurácia e resolução temporal. O acerto de uma classe ampla pode fornecer informação insuficiente para o planejamento, e a proximidade entre classes não garante proximidade em minutos ou horas. Não foram avaliados os efeitos das previsões sobre as decisões dos usuários, de modo que a maior acurácia observada não implica, necessariamente, maior utilidade operacional.

VI.2 Trabalhos Futuros

Como trabalho futuro, propõe-se comparar a classificação por faixas com a regressão direta do tempo de execução e com abordagens que produzam intervalos preditivos. A comparação deverá utilizar os mesmos dados, atributos e partições de avaliação, considerando tanto o erro temporal das estimativas pontuais quanto a cobertura e a largura dos intervalos.

Propõe-se também uma avaliação orientada à decisão do usuário, por meio de cenários de planejamento e estudos com usuários. Essa avaliação deverá investigar em que medida as previsões auxiliam na seleção da infraestrutura, na definição de limites de tempo e no atendimento de prazos, considerando os custos distintos da subestimação e da superestimação. Dessa forma, a escolha da granularidade poderá considerar conjuntamente o desempenho preditivo, a amplitude das faixas e a utilidade para a decisão pretendida.

Como desdobramento da avaliação preditiva, propõe-se investigar o comportamento temporal dos modelos sob o efeito de *concept drift*, decorrente de mudanças no perfil de carga do *cluster*, atualizações de *hardware* ou envelhecimento dos dados. Esse estudo pode subsidiar mecanismos que acompanhem o desempenho a partir das execuções concluídas e indiquem a necessidade de retreinamento quando a degradação ultrapassar limites previamente definidos. Também se propõe realizar um estudo de ablação por grupos de atributos, removendo separadamente os metadados de domínio, as características dos dados de entrada e os atributos da infraestrutura, a fim de quan-

tificar a contribuição preditiva de cada grupo. Adicionalmente, podem ser avaliados modelos de aprendizado profundo e ferramentas de Aprendizado de Máquina Automatizado (*AutoML*) para seleção de algoritmos e ajuste de hiperparâmetros, principalmente nos cenários com maior granularidade temporal.

Em relação à extensão da avaliação, trabalhos futuros devem incorporar outros *workflows* científicos e novas infraestruturas de produção, permitindo examinar a validade externa da abordagem em diferentes domínios e arquiteturas computacionais. A implementação de novos conectores de infraestrutura também permitirá avaliar o sistema com outros gerenciadores de recursos, como o `Slurm`, e em ambientes de nuvem híbrida ou baseados em orquestradores de contêineres, como o `Kubernetes`.

No âmbito da extensão experimental de calibração local, propõe-se investigar o emprego de Aprendizado Ativo (*Active Learning*) para selecionar as configurações que devem compor a amostra de sondagem. O objetivo é aumentar a representatividade das execuções locais e avaliar se é possível reduzir a quantidade de observações necessária para adaptar os modelos a uma nova infraestrutura. Como trabalho futuro, propõe-se também investigar a quantidade mínima de execuções históricas necessária para incorporar um novo *workflow* ao `AranduFlow`. Esse estudo deverá analisar a evolução do desempenho preditivo em função do número e da diversidade das execuções utilizadas no treinamento, permitindo estabelecer critérios para determinar quando o histórico coletado é suficiente para disponibilizar predições confiáveis para um novo *workflow*.

Do ponto de vista da evolução do protótipo, permanecem como trabalhos futuros a incorporação de autenticação e autorização, a migração do `SQLite` para um banco de dados multiusuário e a implementação do acompanhamento dos estados das execuções. Também se prevê a adoção de um mecanismo de versionamento dos modelos que preserve os artefatos anteriores e associe cada previsão à versão utilizada, ampliando a rastreabilidade e a capacidade de auditoria do sistema. Essas extensões são necessárias para avaliar o `AranduFlow` em cenários compartilhados de produção com maior número de usuários e submissões concorrentes.

Entre as evoluções propostas, considera-se prioritário o amadurecimento operacional da arquitetura do `AranduFlow`, incluindo autenticação e autorização, persistência multiusuário, acompanhamento dos estados das execuções e versionamento dos modelos. Essa evolução é necessária para viabilizar o uso continuado do *gateway* em ambientes compartilhados de produção e fornecer uma base mais robusta para avaliações posteriores de usabilidade, confiabilidade e sobrecarga. A ampliação dos ambientes de validação, o aperfeiçoamento do protocolo experimental e a investigação de modelos mais sofisticados permanecem relevantes, podendo avançar com os dados históricos disponíveis. Já os estudos com usuários em situações reais de submissão dependem de uma infraestrutura operacional suficientemente estável para sustentar avaliações representativas.

Referências

Camila Ferrari, João Vitor Moraes, and Daniel de Oliveira. Nmfst. p: um notebook para identificação em paralelo de subárvores frequentes em conjuntos de árvores filogenéticas. In *Brazilian e-Science Workshop (BreSci)*, pages 1–8. SBC, 2023. , 6, 41

Daniel C. M. de Oliveira, Ji Liu, and Esther Pacitti. *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2019. ISBN 978-3-031-00744-6. doi: 10.2200/S00915ED1V01Y201904DTM060. URL `https://doi.org/10.2200/S00915ED1V01Y201904DTM060`. 1

Ewa Deelman et al. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015. 1, 5, 7, 22

Paolo Di Tommaso et al. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4):316–319, 2017. 1, 5, 22

Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012. 1, 5, 22

Peter Amstutz et al. Common workflow language, v1.0. *Specification, Common Workflow Language working group*, 2016. 1, 5

Enis Afgan, Dannon Baker, Benjamin Batut, Marius Van den Beek, David Bouvier, Martina Cech, et al. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1):W537–W544, 2018. 1

Philipp A. Friese, Ahmed Eleliemy, Utz-Uwe Haus, and Martin Schulz. Closing the hpc-cloud convergence gap: Multi-tenant slingshot RDMA for kubernetes. In *IEEE International Conference on Cluster Computing, CLUSTER 2025, Edinburgh, United Kingdom, September 2-5, 2025*, pages 1–10. IEEE, 2025. doi: 10.1109/CLUSTER59342.2025.11186471. URL `https://doi.org/10.1109/CLUSTER59342.2025.11186471`. 1, 2, 8

Francieli Boito, Luan Teylo, Mihail Popov, Théo Jolivel, François Tessier, Jakob Lüttgau, Julien Monniot, Ahmad Tarraf, André Ramos Carneiro, and Carla Osthoff. A deep look into the temporal I/O behavior of HPC applications. In *IEEE International Parallel and Distributed Processing Symposium, IPDPS 2025, Milano, Italy, June 3-7, 2025*, pages 749–762. IEEE, 2025. doi: 10.1109/IPDPS64566.2025.00072. URL https://doi.org/10.1109/IPDPS64566.2025.00072. 2, 8

Luis C. R. Alvarenga, Yuri Frota, Daniel de Oliveira, and Rafaelli Coutinho. Optimizing resource estimation for scientific workflows in HPC environments: A layered-bucket heuristic approach. *Concurr. Comput. Pract. Exp.*, 37(4-5), 2025. doi: 10.1002/CPE.8381. URL https://doi.org/10.1002/cpe.8381. 2, 21, 23

Esma Yildirim, Mohab Hussein, Mikhail Titov, and Ozgur Ozan Kilic. Predicting runtime and resource utilization of jobs on integrated cloud and hpc systems. *Future Generation Computer Systems*, 176:108230, 2026. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2025.108230. URL https://www.sciencedirect.com/science/article/pii/S0167739X25005242. 2, 20, 23

Raslan Ribeiro, Rafaelli Coutinho, and Daniel de Oliveira. Apoena: Towards a cloud dimensioning approach for executing sql-like workloads using machine learning and provenance. In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS)*, pages 289–296, Rome, Italy, 2024. SCITEPRESS – Science and Technology Publications. ISBN 978-989-758-692-7. doi: 10.5220/0012633000003690. 2, 22, 23

Kary A. C. S. Ocaña, Marcelo Galheigo, Carla Osthoff, Luiz M. R. Gadelha Jr., Fábio Porto, Antônio Tadeu A. Gomes, Daniel de Oliveira, and Ana Tereza Ribeiro de Vasconcelos. Bioinfoportal: A scientific gateway for integrating bioinformatics applications on the brazilian national high-performance computing network. *Future Gener. Comput. Syst.*, 107:192–214, 2020. doi: 10.1016/J.FUTURE.2020.01.030. URL https://doi.org/10.1016/j.future.2020.01.030. 2, 22, 23

Alexandre H. L. Porto, Micaella Coelho, Hiago Mayk G. de A. Rocha, Carla Osthoff, Kary A. C. S. Ocaña, and Douglas de O. Cardoso. Assuming the best: Towards a reliable protocol for resource usage prediction for high-performance computing based on machine learning. *Future Gener. Comput. Syst.*, 175:108070, 2026. doi: 10.1016/J.FUTURE.2025.108070. URL https://doi.org/10.1016/j.future.2025.108070. 2, 20, 23

Rafaelli de C. Coutinho, Yuri Frota, Kary A. C. S. Ocaña, Daniel de Oliveira, and Lúcia M. A. Drummond. A dynamic cloud dimensioning approach for parallel scientific workflows: a case

study in the comparative genomics domain. *J. Grid Comput.*, 14(3):443–461, 2016. doi: 10.1007/S10723-016-9367-X. URL https://doi.org/10.1007/s10723-016-9367-x. 2, 21, 23

Proyecto Montoya. Arandu: verbete do nuevo tesoro hispanoguaraní. https://www.proyectomontoya.org.py/nthg/arandu. Acesso em: 11 jul. 2026. 3

Naruya Saitou and Masatoshi Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987. 6

Shishir Bharathi, Ann Chervenak, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. Characterization of scientific workflows. In *2008 Third Workshop on Workflows in Support of Large-Scale Science*, pages 1–10, 2008. doi: 10.1109/WORKS.2008.4723958. 7

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4 edition, 2020. 9

Matheus Sobrinho and Aletéia Paumgartten. Serviço de predição de recursos para execução eficiente de workflows de bioinformática em nuvem federada com aprendizado de máquina. In *Anais da IV Escola Regional de Alto Desempenho do Centro-Oeste*, pages 22–26, Porto Alegre, RS, Brasil, 2021. SBC. doi: 10.5753/eradco.2021.18419. URL https://sol.sbc.org.br/index.php/eradco/article/view/18419. 9, 20, 23

Jonathan Bader, Fabian Lehmann, Lauritz Thamsen, Ulf Leser, and Odej Kao. Lotaru: Locally predicting workflow task runtimes for resource management on heterogeneous infrastructures. *Future Generation Computer Systems*, 150:171–185, 2024. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2023.08.022. URL https://www.sciencedirect.com/science/article/pii/S0167739X23003229. 9, 20, 23

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 9, 11

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. 9

G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Springer International Publishing, 2023. ISBN 9783031387470. URL https://books.google.com.br/books?id=ygzJEAAAQBAJ. 9, 10, 12, 13, 14

Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc.", 2022. 9, 11

Trevor Hastie, Robert Tibshirani, Jerome Friedman, et al. The elements of statistical learning, 2009. 9, 10, 11, 12, 13

David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020. 14

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. 15, 45

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009. 15, 45

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999. 15, 45

Daniel de Oliveira, Kary A. C. S. Ocaña, Fernanda Araújo Baião, and Marta Mattoso. A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *J. Grid Comput.*, 10(3):521–552, 2012. doi: 10.1007/S10723-012-9227-2. URL `https://doi.org/10.1007/s10723-012-9227-2`. 16

Juliana Freire and Cláudio T. Silva. Making computations and publications reproducible with vistrails. *Comput. Sci. Eng.*, 14(4):18–25, 2012. doi: 10.1109/MCSE.2012.76. URL `https://doi.org/10.1109/MCSE.2012.76`. 16

Michel J.F. Rosa, Célia Ghedini Ralha, Maristela Holanda, and Aleteia P.F. Araujo. Computational resource and cost prediction service for scientific workflows in federated clouds. *Future Generation Computer Systems*, 125:844–858, 2021. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2021.07.030. URL `https://www.sciencedirect.com/science/article/pii/S0167739X21002922`. 18, 23

André Nascimento, Victor Olimpio, Vítor Silva, Aline Paes, and Daniel de Oliveira. A reinforcement learning scheduling strategy for parallel cloud-based workflows. In *IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2019, Rio de Janeiro, Brazil, May 20-24, 2019*, pages 817–824. IEEE, 2019. doi: 10.1109/IPDPSW.2019.00134. URL `https://doi.org/10.1109/IPDPSW.2019.00134`. 21, 23

Kary AcS Ocaña and Alberto Mr Dávila. Phylogenomics-based reconstruction of protozoan species tree. *Evolutionary Bioinformatics*, 7:EBO–S6861, 2011. 41

André Igor Pereira, Ubiratam de Paula Junior, Daniel de Oliveira, and Rafaelli Coutinho. AranduFlow: Towards using performance and domain metadata to predict and optimize dataflow

execution in HPC environments. In *Proceedings of the 28th International Conference on Enterprise Information Systems – Volume 1: ICEIS*, pages 272–279. INSTICC, SciTePress, 2026. ISBN 978-989-758-834-1. doi: 10.5220/0014832800004018. 66

Apêndice A Dicionários de Dados

Neste apêndice são descritos os atributos utilizados para representar as execuções dos *workflows* NMFSt.P e Montage coletadas durante os experimentos realizados. Cada linha da base corresponde a uma execução observada. Os campos combinam metadados do domínio do *workflow*, parâmetros de execução, variáveis derivadas da carga de entrada, informações da infraestrutura e a variável-alvo de tempo de execução.

A.1 Dicionário de Dados do NMFSt.P

No NMFSt.P, os atributos de domínio descrevem o algoritmo de alinhamento, o formato da árvore filogenética e as características dos arquivos de sequências biológicas utilizados como entrada. A Tabela A.1 consolida os campos específicos do NMFSt.P. Os atributos de infraestrutura comuns aos dois estudos de caso são apresentados separadamente na Tabela A.3.

Tabela A.1: Dicionário de dados do NMFSt.P.

Campo	Tipo/Unidade	Descrição
`algoritmo`	Categórico	Algoritmo utilizado na execução do *workflow*, como `clustalw` ou `probcons`.
`formato_arvore`	Categórico	Formato de saída da árvore filogenética, por exemplo `newick` ou `nexus`.
`cluster`	Categórico	Ambiente de execução de origem da amostra, como *Cluster* A ou *Cluster* B.
`threads_usados`	Inteiro	Quantidade de processos ou *threads* solicitada na execução do *workflow*.
`tempo_execucao`	Segundos	Tempo total observado da execução, medido pelo programa executor entre o início e a conclusão das atividades do *workflow*.
`qtd_sequencias`	Inteiro	Quantidade de arquivos FASTA de entrada considerados na execução.
`total_sequencias`	Inteiro	Soma da quantidade de sequências contidas em todos os arquivos de entrada.
`size_arquivos`	Bytes	Soma do tamanho, em bytes, dos arquivos de entrada.
`media_tamanho_sequencias`	Bases ou resíduos	Média dos tamanhos médios das sequências informados para os arquivos de entrada.
`size_arquivos_per_thread`	Bytes por *thread*	Razão entre `size_arquivos` e `threads_usados`.
`total_sequencias_per_thread`	Sequências por *thread*	Razão entre `total_sequencias` e `threads_usados`.
`time_class`	Categórico	Classe do tempo de execução após discretização por quantis, representada por rótulos `Q1`, `Q2`, ..., `Qn`.

A.2 Dicionário de Dados do Montage

No Montage, os atributos de domínio descrevem as opções de reprojeção, combinação, visualização e a coleção de imagens astronômicas em formato FITS processadas em cada execução. A Tabela A.2 consolida os campos específicos do Montage. Assim como no NMFSt.P, os atributos de infraestrutura usados pelos modelos são descritos no vetor comum apresentado na Tabela A.3.

Tabela A.2: Dicionário de dados do Montage.

Campo	Tipo/Unidade	Descrição
`system`	Categórico	Sistema de coordenadas astronômicas utilizado na geração do mosaico.
`mproject_e`	Booleano	Indica se a opção `-e` foi utilizada na etapa `mProject`.
`mproject_X`	Booleano	Indica se a opção `-X` foi utilizada na etapa `mProject`.
`coadd_type`	Categórico	Método de combinação das imagens, como `mean`, `median` ou `count`.
`madd_e`	Booleano	Indica se a opção `-e` foi utilizada na etapa `mAdd`.
`color_table`	Inteiro	Código da tabela de cores aplicada na geração da visualização do mosaico.
`stretch`	Categórico	Função de escala visual aplicada ao mosaico, como `linear`, `log`, `loglog`, `asinh` ou `gaussian-log`.
`range`	Categórico	Intervalo de intensidade utilizado na visualização, por exemplo `0 max` ou `-1s 99.999%`.
`threads`	Inteiro	Quantidade de *threads* solicitada para a execução do Montage.
`total_time_seconds`	Segundos	Tempo total observado da execução do *workflow*.
`total_size`	Bytes	Soma do tamanho dos arquivos FITS de entrada.
`qtt_files`	Inteiro	Quantidade de arquivos FITS utilizados como entrada.
`avg_file_size`	Bytes	Tamanho médio dos arquivos FITS de entrada.
`size_mb`	MB	Conversão de `total_size` para megabytes.
`avg_file_size_mb`	MB	Conversão de `avg_file_size` para megabytes.
`files_per_thread`	Arquivos por *thread*	Razão entre `qtt_files` e `threads`.
`size_mb_per_thread`	MB por *thread*	Razão entre `size_mb` e `threads`.
`cluster`	Categórico	Ambiente de execução de origem da amostra, como *Cluster* A ou *Cluster* B.
`time_class`	Categórico	Classe do tempo de execução após discretização por quantis, representada por rótulos `Q1`, `Q2`, ..., `Qn`.

A.3 Vetor de Características da Infraestrutura

A Tabela A.3 apresenta os atributos utilizados para representar a infraestrutura computacional nas execuções do NMFSt.P e do Montage. Esse conjunto corresponde ao vetor de características da infraestrutura utilizado de forma igual para ambos os *workflows*, permitindo representar os ambientes computacionais por meio dos mesmos metadados.

Tabela A.3: Vetor de características da infraestrutura comum ao NMFSt.P e ao Montage.

Campo	Tipo/Unidade	Descrição
`cpu_model`	Categórico	Modelo de processador.
`physical_cores`	Inteiro	Quantidade de núcleos físicos do processador.
`logical_cores`	Inteiro	Quantidade de núcleos lógicos do processador.
`l2_cache_kb`	KB	Tamanho total de cache L2.
`l3_cache_kb`	KB	Tamanho total de cache L3.
`ram_gb`	GB	Capacidade de memória principal.
`ram_channels`	Inteiro	Número de canais de memória disponíveis.
`ram_speed`	MHz	Frequência da memória principal; valor zero indica informação não disponível.

Apêndice B Intervalos de Discretização por Quantis

Este apêndice apresenta de forma detalhada a distribuição de frequências, a quantidade de amostras e os limites dos intervalos de tempo de execução (tempo mínimo e tempo máximo) definidos para os cenários de discretização por quantis contendo 3, 7, 10 e 15 classes.

Os dados referem-se aos dois estudos de caso avaliados: o *workflow* científico de bioinformática NMFSt.P e o *workflow* de astronomia Montage.

B.1 Intervalos de Discretização para o Cenário de 3 Classes

A Tabela B.1 apresenta os intervalos para as três classes do NMFSt.P e a Tabela B.2 detalha os respectivos intervalos para o Montage.

Tabela B.1: Distribuição de frequências e intervalos de tempo (cenário com três classes) no NMFSt.P.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	476	4,84	131,78
Q2	476	131,78	2.533,32
Q3	476	2.533,32	221.539,98

Tabela B.2: Distribuição de frequências e intervalos de tempo (cenário com três classes) no Montage.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	923	5,07	37,98
Q2	923	37,98	325,85
Q3	923	325,85	92.602,43

B.2 Intervalos de Discretização para o Cenário de 7 Classes

A Tabela B.3 apresenta os intervalos para as sete classes do NMFSt.P e a Tabela B.4 descreve o mapeamento correspondente para o Montage.

Tabela B.3: Distribuição de frequências e intervalos de tempo (cenário com sete classes) no NMFSt.P.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	204	4,84	43,85
Q2	204	43,85	95,46
Q3	204	95,46	264,65
Q4	204	264,65	679,28
Q5	204	679,28	4.657,90
Q6	204	4.657,90	15.512,53
Q7	204	15.512,53	221.539,98

Tabela B.4: Distribuição de frequências e intervalos de tempo (cenário com sete classes) no Montage.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	396	5,07	18,82
Q2	395	18,82	33,52
Q3	396	33,52	50,58
Q4	395	50,58	108,73
Q5	396	108,73	394,57
Q6	395	394,57	1.109,49
Q7	396	1.109,49	92.602,43

B.3 Intervalos de Discretização para o Cenário de 10 Classes

A Tabela B.5 detalha as dez faixas obtidas para o NMFSt.P e a Tabela B.6 apresenta as dez faixas para o Montage.

Tabela B.5: Distribuição de frequências e intervalos de tempo (cenário com dez classes) no NMFSt.P.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	143	4,84	34,85
Q2	143	34,85	59,57
Q3	143	59,57	102,46
Q4	142	102,46	222,75
Q5	143	222,75	413,77
Q6	143	413,77	1.040,48
Q7	142	1.040,48	3.640,96
Q8	143	3.640,96	10.114,91
Q9	143	10.114,91	20.942,92
Q10	143	20.942,92	221.539,98

Tabela B.6: Distribuição de frequências e intervalos de tempo (cenário com dez classes) no Montage.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	277	5,07	15,20
Q2	277	15,20	25,44
Q3	277	25,44	34,94
Q4	277	34,94	46,21
Q5	277	46,21	65,53
Q6	276	65,53	171,41
Q7	277	171,41	373,66
Q8	277	373,66	790,19
Q9	277	790,19	1.417,12
Q10	277	1.417,12	92.602,43

B.4 Intervalos de Discretização para o Cenário de 15 Classes

A Tabela B.7 e a Tabela B.8 contêm a distribuição detalhada dos limites de tempo de execução para quinze classes do NMFSt.P e Montage, respectivamente.

Tabela B.7: Distribuição de frequências e intervalos de tempo (cenário com quinze classes) no NMFSt.P.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	96	4,84	23,22
Q2	95	23,22	41,72
Q3	95	41,72	59,57
Q4	95	59,57	83,61
Q5	95	83,61	131,78
Q6	95	131,78	222,75
Q7	95	222,75	323,27
Q8	96	323,27	525,97
Q9	95	525,97	1.040,48
Q10	95	1.040,48	2.533,32
Q11	95	2.533,32	5.784,46
Q12	95	5.784,46	10.114,91
Q13	95	10.114,91	16.245,08
Q14	95	16.245,08	27.696,03
Q15	96	27.696,03	221.539,98

Tabela B.8: Distribuição de frequências e intervalos de tempo (cenário com quinze classes) no Montage.

Classe	Quantidade de Amostras	Tempo Mínimo (s)	Tempo Máximo (s)
Q1	185	5,07	11,59
Q2	185	11,59	18,10
Q3	184	18,10	25,44
Q4	185	25,44	31,75
Q5	184	31,75	37,98
Q6	185	37,98	46,21
Q7	184	46,21	58,59
Q8	185	58,59	76,53
Q9	184	76,53	171,41
Q10	185	171,41	325,85
Q11	184	325,85	496,74
Q12	185	496,74	790,19
Q13	184	790,19	1.179,08
Q14	185	1.179,08	1.802,95
Q15	185	1.802,95	92.602,43

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO/DIVULGAÇÃO DE DOCUMENTO ELETRÔNICO NO CATÁLOGO DO SISTEMA DE BIBLIOTECAS DO CEFET/RJ E NO REPOSITÓRIO INSTITUCIONAL

Na qualidade de titular dos direitos de autor da publicação abaixo citada, de acordo com a Lei nº 9610/98, autorizo, a partir da presente data, o CEFET/RJ, a disponibilizar gratuitamente, sem ressarcimento de direitos autorais, conforme permissões assinadas abaixo, no Catálogo Online do Sistema de Bibliotecas e no Repositório Institucional do Cefet/RJ, ou qualquer outro formato de disseminação da informação que a instituição implemente, no formato PDF, para fins de leitura, impressão e/ou download pela internet, a título de divulgação da produção científica gerada por esta Instituição.
Declaro, ainda, que a versão definitiva do trabalho a ser disponibilizada é de inteira responsabilidade do(a) autor(a) e do(a) orientador(a), que atestam sua conformidade com a versão aprovada e autorizam sua divulgação nos meios institucionais do CEFET/RJ, eximindo a Instituição de qualquer responsabilidade quanto ao conteúdo, à integridade ou a eventuais alterações posteriores ao depósito.

1- Tipo de trabalho: [ ] Projeto Final (Graduação) [ ] Especialização
[x] Dissertação [ ] Tese

2- Identificação do trabalho/autor(es):
Curso: Mestrado Em Ciência da Computação
Título: ARANDUFLOW: RUMO AO USO DE METADADOS DE DESEMPENHO E DOMÍNIO PARA PREVER TEMPO DE EXECUÇÃO DE WORKFLOWS EM AMBIENTES DE HPC
Autor(es): André Igor Pereira
Orientador: Rafaelli Coutinho
Co-orientador: Daniel de Oliveira

3- Informações de acesso ao trabalho:

Este trabalho é confidencial? [ ] Sim [x] Não
Pode ser liberado para publicação? [x] Total [ ] Parcial [ ] Não
Justifique:_____
_____
Em caso de publicação parcial, assinale as permissões:
[ ] Sumário
[ ] Capítulos. Especifique: _____
[ ] Bibliografia
[ ] Outras partes do trabalho: _____

Rio de Janeiro, 24/09/2026

Documento assinado digitalmente
gov.br ANDRE IGOR PEREIRA
Data: 24/09/2026 12:10:03-0300
Verifique em https://validar.iti.gov.br

_____
Assinatura do(s) autor(es)

RAFAELLI DE CARVALHO COUTINHO:12132776717
Assinado de forma digital por RAFAELLI DE CARVALHO COUTINHO:12132776717
Dados: 2026.09.24 15:01:49 -03'00'

Assinatura do orientador

Documento assinado digitalmente
gov.br DANIEL CARDOSO MORAES DE OLIVEIRA
Data: 25/09/2026 15:16:21-0300
Verifique em https://validar.iti.gov.br

_____
Assinatura do co-orientador