Modelos de Predição do Tempo de Jobs Aplicados a um Ambiente de Produção de Alto Desempenho

Miguel de Lima[1], Bernardo Gallo[1], Luciano Andrade[1], Felipe A. Portella[2], Paulo J. B. Estrela[2], Renzo Q. Malini[2], Alan L. Nunes[1], José Viterbo[1], Lúcia M. A. Drummond[1]

{miguel_flj, bgallo, lucianoandrade, alan_lira}@id.uff.br
{viterbo, lucia}@ic.uff.br
{felipeportella, paulo.estrela, renzo}@petrobras.com.br

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

[2]Petróleo Brasileiro S.A. (PETROBRAS) – Rio de Janeiro – RJ – Brazil

Resumo. *Este artigo tem como objetivo avaliar o impacto da utilização do tempo de execução de jobs, previstos pelos modelos de aprendizado de máquina J48, Linear Regression e Random Forest, no escalonamento em sistemas computacionais de alto desempenho. Os tempos previstos por esses modelos foram usados pela política SJF (Shortest Job First) em uma simulação de escalonamento baseada em um conjunto de milhares de jobs de aplicações reais de alto desempenho que foram executados em um ambiente de produção da Petrobras. As métricas de desempenho de escalonamento throughput e tempo médio de espera foram examinadas adicionalmente às tradicionais métricas teóricas de modelos preditores. Demonstramos que o efeito prático das predições pode divergir do resultado teórico dos preditores, destacando a importância de avaliações empíricas para a otimização do escalonamento de jobs.*

1. Introdução

Em sistemas de computação de alto desempenho (HPC), *jobs* são executados tipicamente de forma exclusiva em um subconjunto de nós computacionais organizados em *clusters* ou *data centers* distribuídos. Escalonadores de *jobs*, como o *Slurm* [Yoo et al. 2003], são responsáveis pelo gerenciamento desses ambientes complexos. Em geral, quando há escassez de recursos para atender à demanda, *jobs* ficam retidos em uma fila de espera até que haja novamente recursos suficientes para cumprir os requisitos de escalonamento.

Nesse contexto, o conhecimento antecipado do tempo de execução de cada *job* permite que escalonadores implementem políticas de escalonamento mais sofisticadas [Feitelson and Weil 1998]. Inúmeros benefícios podem ser observados a partir de um escalonamento mais eficiente dos *jobs*, tais como a redução do tempo médio de espera, o aumento da quantidade de *jobs* concluídos por fração de tempo (*throughput*), o aumento da taxa de utilização de recursos, a redução de custos operacionais e energéticos, a melhoria do planejamento a longo prazo do escalonamento, entre outros, permitindo que sistemas de HPC se tornem ainda mais eficazes [Gaussier et al. 2018, Nichols et al. 2022].

A adoção de algoritmos de aprendizado de máquina (ML) tem emergido como uma abordagem promissora para a tarefa de predição do tempo de execução de *jobs* em

sistemas de HPC [Witt et al. 2019], visto que diversas características podem ser consideradas durante a geração dos modelos preditores, tais como o tipo de aplicação executada, a configuração dos recursos solicitados e as informações do usuário responsável pelo *job*. As técnicas de ML são capazes de capturar padrões complexos e não lineares que estejam presentes nos dados [Tanash et al. 2019]. Além disso, modelos preditores podem ser treinados continuamente com novos dados, permitindo que se ajustem melhor às mudanças eventuais do sistema e do perfil de *jobs* executados.

Em geral, preditores são avaliados a partir de métricas teóricas que fornecem uma compreensão inicial do desempenho que pode ser obtido para as predições. Apesar disso, devido à dinamicidade de sistemas computacionais e de *jobs* executados neles e à ausência de diversidade de dados em vários sistemas reais, não há garantias de que as predições sejam suficientes para tornar um sistema de HPC mais eficaz [Kuchnik et al. 2019]. Por essa razão, uma avaliação prática dos preditores permite a identificação e correção de problemas de desempenho não evidenciados em cenários controlados.

O objetivo deste trabalho é analisar o impacto no desempenho de um sistema de HPC ao serem considerados diferentes modelos de predição do tempo de execução, gerados a partir da metodologia proposta em [Nunes et al. 2023]. Para isso, milhares de *jobs* de simulação de reservatórios de petróleo, executados em um ambiente de produção da Petrobras, foram utilizados como dados de entrada para a nossa implementação de simulador de escalonamento de *jobs*. Neste trabalho, os *jobs* a serem escalonados são avaliados em intervalos (janelas) de tempo e são ordenados considerando duas políticas tradicionais: i) *Shortest Job First* (SJF), que ordena-os por tempos estimados de execução e ii) *First-Come First-Served* (FCFS), que ordena-os por tempos de submissão. Durante a simulação, as métricas *throughput* e tempo médio de espera dos *jobs* foram calculadas para diferentes configurações. A análise dos resultados considerando as métricas teóricas de qualidade dos preditores destacou a necessidade de escolha de preditor adaptada aos aspectos do sistema de HPC e objetivos de otimização.

O restante deste artigo está organizado da seguinte forma. A Seção 2 expõe os conceitos básicos de simulação de reservatórios de petróleo, de escalonamento de *jobs* e de predição do tempo de execução de *jobs*. A Seção 3 destaca os trabalhos relacionados. A Seção 4 apresenta a construção dos preditores de tempo de execução de *jobs*. A Seção 5 detalha o algoritmo desenvolvido para simular um escalonador de *jobs*. A Seção 6 apresenta a avaliação experimental e, por fim, as conclusões são expostas na Seção 7.

2. Conceitos Preliminares

2.1. Simulação de Reservatórios de Petróleo

A simulação de reservatórios é uma ferramenta fundamental para o setor de óleo e gás que ajuda a minimizar riscos e a otimizar tomadas de decisão durante o desenvolvimento de reservatórios de petróleo. Tem por objetivo reproduzir o histórico de produção e prever a produção futura, fornecendo informações valiosas sobre o comportamento de petróleo, gás e água ao longo do tempo. As simulações são conduzidas a partir de modelos de reservatórios, permitindo que engenheiros explorem diversos cenários de forma mais econômica e eficiente do que por meio de operações reais [Portella et al. 2022].

Modelos de reservatórios geralmente empregam grades tridimensionais, que compreendem milhares ou milhões de células para representar um reservatório físico, e

suas equações matemáticas são derivadas de princípios físicos como conservação de massa, equilíbrio termodinâmico, transferência de calor e fluxo em meios porosos (Lei de Darcy) [Coats 1982]. Em razão da complexidade computacional requerida por esses modelos numéricos, simulações de reservatórios são normalmente executadas em *clusters* de computação de alto desempenho (HPC). Para lidar com as incertezas associadas à caracterização de um reservatório e ao comportamento de seus fluídos, múltiplas simulações devem ser executadas para um mesmo cenário.

Neste trabalho, *jobs* de simulação de reservatórios executam uma aplicação multiprocesso de memória compartilhada, implementada via *OpenMP*. Embora a execução da aplicação esteja limitada a um nó computacional, o problema de simulação de reservatórios compreende diversos fluxos de trabalho que demandam múltiplas simulações individuais, da ordem de centenas a milhares por usuário. Cada simulação é executada como um *job* particular em um *cluster*, de modo que o processamento de simulações de reservatórios pode ser tipificado como massivamente paralelo. Por fim, os *jobs* considerados possuem as seguintes características de execução: i) tempo de duração: entre 300 e 9.389.026 segundos; ii) quantidade de nós: entre 1 e 40; iii) quantidade de *CPU cores*: entre 1 e 1.600; e iv) quantidade de memória: entre 7,5 e 12.000 *GB*.

2.2. Escalonamento de Jobs

Em geral, um problema de escalonamento especifica três componentes: carga de trabalho, recursos e requisitos [Lopes and Menascé 2016]. A carga de trabalho define os consumidores dos recursos. Normalmente compreende *jobs*, que são coleções de tarefas computacionais. Os recursos, necessários para executar a carga de trabalho, são constituídos por um conjunto de nós organizados em *clusters*, geralmente conectados por rede de alta velocidade. Os requisitos determinam o objetivo do escalonamento de *jobs*, que deve ser atendido pela solução. Usualmente, o foco é a otimização de uma ou mais métricas de desempenho que são afetadas pelas decisões de escalonamento, tais como a taxa de utilização dos recursos, a quantidade de *jobs* concluídos por fração de tempo (*throughput*) e o tempo médio de espera de *jobs* na fila aguardando a disponibilidade de recursos.

Devido a complexidade desses três componentes, a maioria dos problemas de escalonamento é classificada como computacionalmente difícil [Pinedo 2016]. Por essa razão, várias ferramentas que auxiliam no gerenciamento de recursos e escalonamento de *jobs*, tais como *Slurm* e *Torque*, têm sido amplamente adotadas. Além da otimização de métricas de desempenho, o uso eficiente desses escalonadores de *jobs* pode promover a redução de custos financeiros e energéticos em sistemas de HPC.

2.3. Predição do Tempo de Execução de Jobs

O uso de algoritmos de aprendizado de máquina que sejam capazes de prever o tempo de execução de *jobs* potencialmente aprimora a eficiência de *clusters* de HPC. Escalonadores de *jobs* poderiam utilizar as informações de predição para refinar suas políticas de decisão e redefinir a ordem de execução dos *jobs*, por exemplo. No entanto, preditores têm sido implantados de forma limitada por três razões principais: a falta de diversidade de dados, que pode afetar negativamente o desempenho do preditor; a alteração dos perfis de *jobs* ao longo do tempo, o que é típico na maioria dos sistemas; e a falta de garantia que predições com alta precisão promoverão a eficiência do sistema [Kuchnik et al. 2019]. Em geral, preditores do tempo de execução de *jobs* são avaliados a partir das métricas Erro Médio

Absoluto (MAE), Erro Percentual Médio Absoluto (MAPE) e Erro Quadrático Médio (RMSE) para modelos de regressão, e Acurácia, para modelos de classificação.

3. Trabalhos Relacionados

Escalonadores de *jobs*, como o Slurm, gerenciam recursos computacionais em sistemas paralelos e distribuídos a partir de políticas que determinam a ordem de execução dos *jobs* [Reuther et al. 2018]. Algumas políticas como EASY [Lifka 1998], EASY++ [Tsafrir et al. 2007] e SJF requerem, no entanto, estimativas do tempo de execução dos *jobs*. Assim, diversos estudos concentram-se na predição de tempo de *jobs* em sistemas de HPC e na nuvem [Tanash et al. 2019, Kim et al. 2020, Nunes et al. 2023].

Técnicas de aprendizado de máquina são frequentemente utilizadas para a obtenção de predições, com a maioria dos estudos avaliando o desempenho a partir das métricas acurácia e MAPE. Não obstante, estudos recentes consideram o uso de simulações de escalonamento de *jobs* para a avaliação da utilidade das predições. Um estudo observou que abordagens híbridas de predição usando filtros de Kalman (FMKF e MLKF) reduziram significativamente o tempo médio de espera, comparado a outras técnicas [Naghshnejad and Singhal 2018]. Simulações da política de escalonamento FCFS com *backfilling* demonstraram que a combinação de métodos de aprendizado supervisionado e não supervisionado para a predição do tempo de execução de aplicações de aerodinâmica, usando dados históricos reais, reduziu o tempo médio de espera em 29%, quando comparada ao uso apenas de modelos de regressão [Wang et al. 2021]. Outro estudo [Yang et al. 2023] propôs o uso da informação do diretório de execução dos *jobs*, agrupando-os por semelhança de diretório de execução. Cada grupo foi associado a um preditor específico, e quando estes foram baseados em árvores de decisão, a abordagem alcançou precisão geral de 88,5%. Através de uma simulação que também considerou a política FCFS com *backfilling*, os autores verificaram que a abordagem proposta foi capaz de obter o menor tempo médio de espera quando comparada a outras de referência.

Em nosso trabalho anterior [Nunes et al. 2023], utilizamos um conjunto de dados de *jobs* de Simulação de Reservatórios de Petróleo reais para construir um modelo de predição de tempo de execução. Diferentes técnicas de modelagem, como regressão linear e árvore de decisão, foram consideradas, e como atributos foram utilizadas as informações gerais sobre a execução dos *jobs*, tais como o nome do usuário, o projeto associado, o diretório de execução, o nome e a versão do simulador de reservatório selecionado, o número total de núcleos de CPU solicitados, a hora e o dia da semana de execução, entre outros. O melhor modelo foi selecionado a partir de avaliação experimental com base em métricas tradicionais, como MAE, MAPE, acurácia e coeficiente Kappa.

Neste trabalho, a contribuição é estendida ao apresentar a simulação do escalonamento considerando as predições do tempo de execução dos *jobs* fornecida pelos diferentes modelos, de forma semelhante ao realizado por Naghshnejad e Singhal [2018], Wang *et al.* [2021] e Yang *et al.* [2023]. Entretanto, enquanto no primeiro trabalho os autores implementaram sua simulação a partir de um simulador baseado em eventos de uso geral e os outros dois trabalhos utilizaram o *Slurm Simulator* [Simakov et al. 2018], neste trabalho a simulação foi implementada com o desenvolvimento de um algoritmo específico. Além disso, todos os trabalhos citados anteriormente executaram simulações com a finalidade de demonstrar que o modelo de predição selecionado a partir das

métricas tradicionais de avaliação de desempenho apresentava bom desempenho. O grande diferencial deste trabalho é a proposta de que o impacto dos diferentes modelos de predição do tempo de execução de *jobs* seja avaliado através da simulação do escalonamento considerando a política SJF com diferentes janelas de ordenação dos *jobs* e diferentes configurações do ambiente de execução, permitindo identificar o modelo mais adequado para cada cenário comparando-se o *throughput* e o tempo médio de espera, métricas específicas para avaliação de escalonamento.

4. Preditores do Tempo de Execução de Jobs

Esta seção apresenta o conjunto de dados de *jobs* de simulação de reservatórios, os atributos selecionados e os modelos empregados para a geração e avaliação dos preditores.

4.1. Conjunto de Dados de Jobs de Simulação de Reservatórios

O conjunto de dados explorado neste trabalho caracteriza as atividades diárias de HPC realizadas por mais de 300 engenheiros da *Petrobras*, abrangendo cerca de 5,1 milhões de registros de *jobs* executados ao longo de três anos pelo Slurm. Porém, como a infraestrutura interna é regularmente atualizada, o período de análise dos dados foi limitado a um ponto estável de recursos oferecidos para a execução dos *jobs*. Por essa razão, o conjunto de dados brutos compreendeu apenas os *jobs* gerenciados pelo Slurm de 1º de junho de 2022 a 30 de junho de 2023, totalizando 2.240.400 *jobs* de simulação de reservatórios, dos quais 1.083.224 (48,35%) foram executados em 2022 e 1.157.176 (51,65%) em 2023.

Uma técnica de filtragem foi aplicada ao conjunto de dados brutos considerando as seguintes regras: (i) descartar *jobs* com status de erro, (ii) descartar *jobs* de teste e (iii) descartar *jobs* que não executaram especificamente simulações de reservatórios de petróleo. Após a aplicação da filtragem, o conjunto de dados foi reduzido de 2.240.400 para 1.305.452 *jobs*. Finalmente, o conjunto de treinamento, utilizado para o treinamento dos preditores, correspondeu a 1.174.186 *jobs* executados entre 1º de junho de 2022 e 30 de maio de 2023 (89,94%), enquanto que o conjunto de teste, utilizado para a avaliação da qualidade das predições, correspondeu a 131.266 *jobs* executados em junho de 2023 (10,06%). Do conjunto de testes, somente dois *jobs* foram executados em mais de um nó.

4.2. Seleção de Atributos

O objetivo principal da modelagem preditiva é projetar com precisão as chances de que algo aconteça ou não, analisando dados históricos relevantes [Kuhn and Johnson 2013]. Considerando que o conjunto de dados de *jobs* de simulação de reservatórios é composto por mais de 40 atributos de diferentes tipos (nominais, ordinais, discretos e contínuos), a seleção apropriada de atributos é crucial para aprimorar a qualidade das predições.

Em um estudo anterior [Nunes et al. 2023], a análise exploratória do conjunto de dados compreendeu três etapas principais: i) investigação de picos de uso do ambiente HPC em termos de dias da semana e horas do dia, na busca por eventos de ocupação do *cluster* espaçados em janelas de observação de trinta minutos; ii) análise visual (*clustering*) para a obtenção de perfis de *jobs* mais frequentes em termos de quantidade de núcleos de processamento e tempo de execução; e iii) uso da ferramenta *Ranker* do *Weka* [Hall et al. 2009] para a medição da importância (ganho de informação) de cada atributo em relação a um atributo alvo, particularmente a classe de duração dos *jobs*. Na ocasião, 23 classes foram definidas para o atributo alvo *job_elapsed_class*, indicadas na Tabela 1.

Tabela 1. Classes do atributo-alvo *job_elapsed_class* (classificadores).

Classe	Intervalo de Duração (em segundos)	Classe	Intervalo de Duração (em segundos)
0	$0 \leq$ elapsed < 60	12	$21.600 \leq$ elapsed < 25.200
1	$60 \leq$ elapsed < 300	13	$25.200 \leq$ elapsed < 28.800
2	$300 \leq$ elapsed < 600	14	$28.800 \leq$ elapsed < 32.400
3	$600 \leq$ elapsed < 900	15	$32.400 \leq$ elapsed < 36.000
4	$900 \leq$ elapsed < 1.800	16	$36.000 \leq$ elapsed < 39.600
5	$1.800 \leq$ elapsed < 2.700	17	$39.600 \leq$ elapsed < 43.200
6	$2.700 \leq$ elapsed < 3.600	18	$43.200 \leq$ elapsed < 86.400
7	$3.600 \leq$ elapsed < 7.200	19	$86.400 \leq$ elapsed < 172.800
8	$7.200 \leq$ elapsed < 10.800	20	$172.800 \leq$ elapsed < 259.200
9	$10.800 \leq$ elapsed < 14.400	21	$259.200 \leq$ elapsed < 345.600
10	$14.400 \leq$ elapsed < 18.000	22	elapsed ≥ 345.600
11	$18.000 \leq$ elapsed < 21.600	—	—

A Tabela 2 descreve o subconjunto de 8 atributos (não-alvos) selecionados para a composição dos preditores ao final da análise exploratória, e que foram reutilizados durante este trabalho. Em relação ao atributo alvo, os modelos classificadores utilizaram o atributo nominal *job_elapsed_class*, enquanto que os modelos regressores utilizaram o atributo discreto *elapsed*, que expressa o tempo de execução de um *job* em segundos.

4.3. Modelos Preditores

Três modelos tradicionais de aprendizado de máquina, disponibilizados no Weka, foram utilizados para a construção dos preditores: *J48*, *LinearRegression* e *RandomForest*. O modelo *J48* implementa o algoritmo C4.5 [Quinlan 1993], um dos algoritmos mais renomados de árvores de decisão e que utiliza o conceito de entropia de informação. As árvores de decisão geradas podem ser empregadas para o problema de classificação e, por esta razão, este algoritmo é frequentemente referido como um classificador estatístico. O modelo *LinearRegression* implementa uma das ferramentas de maior destaque na análise estatística de dados, a regressão linear, que determina a relação entre uma variável dependente e um conjunto de variáveis independentes, assumindo que esta relação pode ser descrita por uma reta. A reta que se ajusta mais adequadamente aos dados é determinada a partir da minimização do erro associado às variáveis. O modelo *RandomForest* implementa um algoritmo versátil que pode ser utilizado tanto para problemas de regressão quanto para problemas de classificação . *Random forests* são construídas a partir da combinação de várias árvores de decisão, cada qual treinada isoladamente, e suas predições são obtidas pela média das predições individuais das árvores.

Todos os preditores foram treinados usando o conjunto de treinamento, tendo por método de avaliação a validação cruzada (*cross-validation*) 10-*fold*. Em seguida, foram testados usando o conjunto de teste, de modo a serem avaliados quanto ao desempenho de predição do tempo de execução de *jobs* não pertencentes à etapa de treinamento.

5. Simulador de Escalonamento de Jobs

Foi desenvolvido para este trabalho um algoritmo que simula o escalonamento de *jobs* em sistemas de HPC (Algoritmo 1). Três ações principais são executadas durante a simulação: (i) a ordenação da lista de *jobs*, considerando a política de escalonamento definida pelo usuário; (ii) a alocação de nós computacionais que estejam disponíveis para

Tabela 2. Subconjunto de atributos não-alvos selecionados para os preditores.

Atributo	Tipo	Descrição
username	Nominal	Usuário que submeteu o job.
account	Nominal	Conta associada ao job.
total_cpus	Discreto	Número de núcleos de CPU alocados para o job.
work_dir_parent	Nominal	'Pai' (caminho lógico) do diretório de trabalho associado ao job.
nodes_prefix	Nominal	Prefixo (4 caracteres iniciais) dos nós que executaram o job.
simulator	Nominal	Nome e versão do simulador de reservatórios utilizado pelo job.
job_start_day_of_week_class	Nominal	Classe associada ao dia da semana no qual o job foi iniciado. Classes definidas: 0 (Segunda-feira), 1 (Terça-feira), 2 (Quarta-feira), 3 (Quinta-feira), 4 (Sexta-feira), 5 (Sábado) e 6 (Domingo).
job_start_hour_of_day_class	Nominal	Classe associada à hora do dia na qual o job foi iniciado. Classes definidas: 0 (das 00h às 06h), 1 (das 06h às 12h), 2 (das 12h às 18h) e 3 (das 18h às 00h).

a execução dos próximos *jobs* da fila; e (iii) o cálculo das métricas de desempenho para o escalonamento, sobretudo o tempo total de execução (*makespan*), a quantidade de *jobs* concluídos por fração de tempo (*throughput*) e o tempo médio de espera dos *jobs* na fila (*waitingTimeAve*). Pressupõe-se que cada *job* será executado em um único nó e utilizará todos os núcleos de processamento, o que impede a aplicação de técnicas como o *backfilling*. Essa limitação reduz a flexibilidade do escalonador em otimizar o uso de recursos, representando uma restrição importante deste trabalho. Na política SJF, é comum ocorrer *starvation*; no entanto, ao utilizar a abordagem de janelas, esse problema é mitigado, pois, no pior cenário, o *job* será o último a ser executado dentro do grupo de *jobs* ao qual pertence.

No primeiro *loop* (linhas 1–3), os *jobs* são agrupados por tempos de submissão que estejam contidos em uma janela de execução, cujo tamanho é indicado pelo parâmetro *windowSize*, definido pelo usuário. A seguir, a política de escalonamento a ser adotada durante a simulação é verificada (linhas 4–9). Se for *First-Come First-Served* (FCFS), os *jobs* serão mantidos em sua ordenação original por tempo de submissão. Se for *Shortest Job First* (SJF), os *jobs* pertencentes à cada agrupamento serão ordenados por ordem crescente do tempo de execução previsto pelo preditor. Após, a lista de nós do *cluster* é inicializada com o menor tempo de submissão da lista de *jobs* (linhas 10–11).

No *loop* principal (linhas 12–22), a lista de *jobs* será processada na ordem definida pela política de escalonamento. Sejam j o próximo *job* da lista, n o nó que ficará disponível mais cedo e t_n o tempo no qual o nó n ficará disponível. Primeiro, obtém-se t_n e o índice associado ao nó n (linha 13). A seguir, define-se o tempo de início da execução do *job* j (linhas 14–18). Se j foi submetido antes de t_n, o tempo de início de j será postergado (linha 15). Caso contrário, o *job* j será executado no tempo t_n (linha 17). Os tempos de espera e de conclusão de j são obtidos em seguida (linhas 19–20). Na sequência, t_n é atualizado com o tempo no qual n ficará disponível novamente (linha 21). Finalmente, são calculadas as métricas de desempenho obtidas com o escalonamento: *makespan* (linhas 23–24), *throughput* (linhas 25–26) e *waitingTimeAve* (linha 27).

Algoritmo 1 Simulador de Escalonamento de Jobs

Require: windowSize: int; schedulingPolicy: string; nodesSize: int; listNodes: int vector[nodesSize]; job: float vector[6] (submit_time, predicted_time, elapsed_time, waiting_time, completing_time, start_time); listJobs: list of jobs

Ensure: makespan: float, throughput: float, waitingTimeMean: float

{Obtém o tempo inicial da janela de execução do job}
for job ∈ listJobs **do**
  group_submit[job] ← **GetWindowTime**(job[submit_time], windowSize)
end for
switch (schedulingPolicy)
  case FCFS**:**
    listJobs ← **Sort**(listJobs, **by**=submit_time) {Ordena jobs por tempos de submissão}
  case SJF**:**
    listJobs ← **Sort**(listJobs, **by**=predicted_time) {Ordena jobs por tempos estimados de execução}
end switch
firstJobSubmitted ← **MinSub**(listJobs) {Obtém o tempo de submissão do job mais antigo}
listNodes ← **Initialize**(firstJobSubmitted) {Inicializa o vetor de nós com o menor tempo de submissão dos jobs}
for job ∈ listJobs **do**
  nodeStartTime, index ← **MinTime**(listNodes) {Obtém o tempo e índice do nó que ficará vago mais cedo}
  if group_submit[job] ≥ nodeStartTime **then**
    job[start_time] ← group_submit[job] {Determina o tempo de início do job}
  else
    job[start_time] ← nodeStartTime
  end if
  job[waiting_time] ← job[start_time] - group_submit[job] {Calcula o tempo de espera do job}
  job[completing_time] ← job[start_time] + job[elapsed_time] {Calcula o tempo de conclusão do job}
  listNodes[index] ← job[completion_time] {Atualiza o próximo tempo no qual o nó 'index' ficará disponível}
end for
lastJobCompleted ← **MaxComp**(listJobs) {Obtém o tempo de conclusão do último job}
makespan ← lastJobCompleted - firstJobSubmitted {Calcula o tempo total para a execução dos jobs}
total_jobs ← |listJobs| {Obtém a quantidade total de jobs}
throughput ← $\frac{\text{total\_jobs}}{\text{makespan}}$ {Calcula a quantidade de jobs concluídos por dia}
waitingTimeAve ← **AveWait**(listJobs) {Calcula o tempo médio de espera dos jobs}
return makespan, throughput, waitingTimeAve

6. Resultados Experimentais

Com o objetivo de avaliar o impacto da qualidade das predições do tempo de execução de *jobs*, foram utilizados três modelos preditores: *J48*, *LinearRegression* e *RandomForest* (Subseção 4.3). A simulação considerou 131.266 *jobs* executados em junho de 2023 (Subseção 4.1), com a adição de suas respectivas durações estimadas pelos preditores. Em relação às políticas de escalonamento adotadas, FCFS foi utilizada como a base para as comparações com a política SJF. A diferença entre essas políticas é que FCFS não utiliza estimativas do tempo de execução de *jobs*, enquanto que SJF prioriza a execução de *jobs* cujas durações estimadas sejam menores. Três casos foram definidos para SJF: (i) **SJF-J48**, que utilizou as predições do *J48*; (ii) **SJF-Linear**, que utilizou as predições do *LinearRegression*; e (iii) **SJF-Forest**, que utilizou as predições do *RandomForest*.

Quatro tamanhos de janelas de tempo (*windowSize*) foram utilizadas: 15 minutos,

30 minutos, 60 minutos e 120 minutos. Note que o tamanho da janela determina a quantidade de *jobs* prontos para a execução, de modo que quanto maior a janela utilizada, maior a quantidade de *jobs* em espera. Além disso, foram utilizadas duas configurações de capacidade do *cluster*: 25 e 150 nós. A combinação de parâmetros resultou em 32 execuções distintas e não repetidas de simulação, visto que o simulador é determinístico. O desempenho dos diferentes experimentos foi comparado a partir de duas métricas retornadas pelo simulador: (i) *throughput*, que fornece a quantidade de *jobs* concluídos por dia; e (ii) *waitingTimeAve*, que fornece o tempo médio de espera dos *jobs*. Esses parâmetros foram escolhidos para observar o impacto das previsões no escalonamento, uma vez que testes preliminares mostraram que essas combinações permitiam melhor análise desse efeito.

Um *cluster* com 25 nós foi escolhido para representar um sistema de pequeno porte, enquanto 150 nós representam um *cluster* de tamanho médio. As janelas de tempo foram selecionadas para permitir a formação de uma fila, ao mesmo tempo que minimizam o impacto no tempo de espera.

A Figura 1 apresenta os valores de *throughput* obtidos pelas políticas FCFS e SJF considerando várias configurações de janelas de tempo e capacidade do *cluster*. É importante ressaltar que 64.804 (49%) dos *jobs* não possuem a informação de duração estipulada pelo usuário (*time_limit*). Portanto, não seria possível aplicar o SJF utilizando essa estimativa. Assim, só foi possível aplicar o SJF com as predições fornecidas pelo modelo de *machine learning* para todos os *jobs*. Nesse contexto, utilizamos o *throughput* como uma métrica para avaliar o impacto das predições no escalonamento. A política SJF combinada às predições obteve *throughput* superior à FCFS, com diferença mais acentuada à medida em que o tamanho da janela aumentava, indicando a vantagem de utilização de modelos preditores. Para janelas pequenas, no entanto, é possível que predições sejam insuficientes para melhorar o escalonamento, comparado ao FCFS.

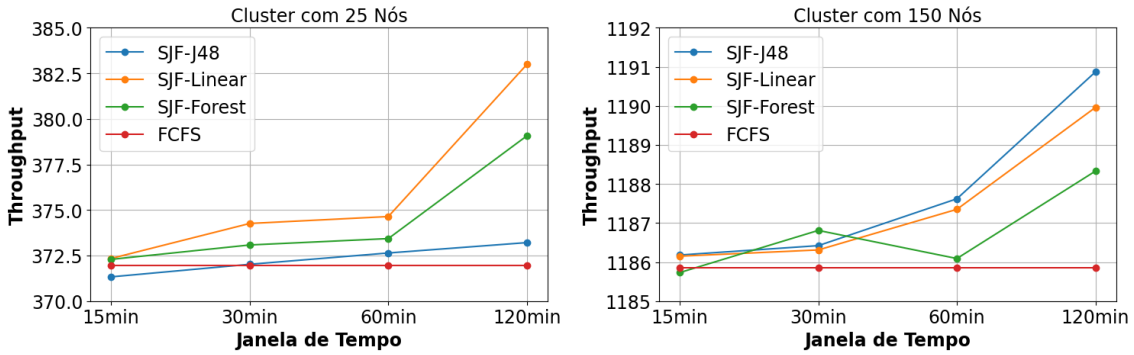

Figura 1. Throughput em clusters simulados com 25 e 150 nós.

A Figura 2 apresenta os valores de tempo médio de espera obtidos pelas políticas FCFS e SJF considerando as mesmas configurações de simulação. Observe que quanto menor o tempo de espera, melhor o desempenho do escalonamento. Não foi possível observar uma melhoria dessa métrica com o uso do SJF para a maioria dos casos. Foi observado, no entanto, que para a maior janela (120 minutos) em um cluster de 25 nós, todos os escalonamentos realizados pela política SJF combinada às predições obtiveram um desempenho melhor do que FCFS. O SJF-Forest foi o único que apresentou melhoria em relação ao FCFS, tanto no tempo médio de espera quanto no *throughput*.

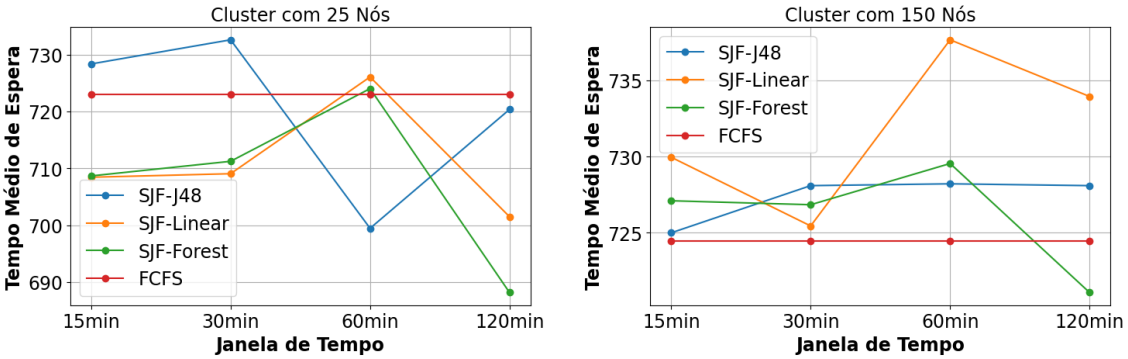

Figura 2. Tempo de espera em clusters simulados com 25 e 150 nós.

6.1. Comparativo entre as métricas teóricas e as métricas da simulação

As métricas teóricas MAE, MAPE e RMSE (Subseção 2.3) foram calculadas para os modelos preditores para fins de comparação com os resultados de desempenho de escalonamento de *jobs* obtidos pelo simulador. Visto que o modelo *J48* prediz uma classe associada ao intervalo de duração de cada *job*, para este modelo, foram adotados os valores mínimos de cada classe (Tabela 1) para a computação dos erros numéricos de predição.

A Tabela 3 sintetiza os valores obtidos por cada modelo preditor. Particularmente, *J48* destacou-se com os menores valores para MAE e MAPE enquanto que *RandomForest* obteve o menor valor para RMSE. No entanto, foi observado a partir das simulações que o uso das predições oferecidas por esses modelos não necessariamente contribuiu para um melhor desempenho no escalonamento dos *jobs*. Fatores dinâmicos, como o tamanho da janela de execução, a quantidade de nós do *cluster* e as medidas de desempenho a serem otimizadas, podem influenciar a eficiência do escalonamento. Em geral, para janelas de execução superiores a 30 minutos, o uso dos preditores combinados à política SJF superou o FCFS em relação ao *throughput*. Além disso, à medida em que o tamanho do *cluster* aumentou, a diferença de eficiência entre as estratégias FCFS e SJF também aumentou. Por fim, os resultados da simulação relevaram que as métricas teóricas são insuficientes para investigar o efeito das predições do tempo de execução dos *jobs* no escalonamento. Particularmente, *LinearRegression*, que não obteve os melhores valores para as métricas teóricas, atingiu o melhor *throughput* para todas as janelas de execução quando foram considerados *clusters* formados por 25 nós.

Tabela 3. Comparativo de métricas teóricas entre diferentes modelos preditores.

Modelo Preditor	Métricas Teóricas		
	MAE	MAPE	RMSE
J48	9.694,93	84,09%	28.876,15
LinearRegression	14.378,54	779,51%	28.242,98
RandomForest	10.382,05	582,15%	26.631,86

7. Conclusões e Direções Futuras

Este trabalho permitiu observar que métricas teóricas são insuficientes na definição do modelo preditor que seja mais adequado em um contexto de escalonamento de *jobs* em

sistemas de HPC. Os resultados apontam a relevância de adaptar a escolha do preditor do tempo de duração dos *jobs* às características específicas do ambiente computacional e aos objetivos de otimização. As métricas teóricas não capturam a complexidade de um sistema de escalonamento, e desse modo, é crucial a realização de avaliações práticas que considerem diferentes configurações operacionais, tais como a variação da quantidade de nós, e metas de desempenho, tais como *throughput* e tempo médio de espera dos *jobs*.

Embora a política de escalonamento *Shortest Job First* (SJF) seja eficiente para a redução do tempo médio de espera, constatamos que possui limitações de otimização do escalonamento de *jobs* em ambientes de HPC que sofram variações na carga de trabalho. Como trabalho futuro, pretende-se explorar outras políticas de escalonamento, como o SJF e FCFS com *backfilling*, além da *Shortest Area First* (SAF). Também serão consideradas outras métricas, como o *bounded slowdown*, que introduz um limite inferior para o tempo de duração e fornece uma avaliação mais justa ao comparar *jobs* de diferentes durações. Além disso, serão explorados outros objetivos de escalonamento, como a minimização de custos energéticos e a maximização da taxa de utilização de recursos.

Agradecimentos

Os autores agradecem à Petrobras pelo financiamento deste trabalho.

Referências

Coats, K. H. (1982). Reservoir Simulation: State of the Art. *Journal of Petroleum Technology*, 34(8):1633–1642.

Feitelson, D. and Weil, A. (1998). Utilization and Predictability in Scheduling the IBM SP2 with Backfilling. In *First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pages 542–546.

Gaussier, E., Lelong, J., Reis, V., and Trystram, D. (2018). Online Tuning of EASY-Backfilling using Queue Reordering Policies. *IEEE Transactions on Parallel and Distributed Systems*, 29(10):2304–2316.

Hall, M., Frank, E., Holmes, G., et al. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Kim, S., Sim, A., Wu, K., Byna, S., Son, Y., and Eom, H. (2020). Towards HPC I/O Performance Prediction through Large-scale Log Analysis. In *29th International Symposium on High-Performance Parallel and Distributed Computing*, pages 77–88. ACM.

Kuchnik, M., Park, J. W., Cranor, C., Moore, E., DeBardeleben, N., and Amvrosiadis, G. (2019). This is why ML-driven cluster scheduling remains widely impractical. Technical report, Carnegie Mellon University.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*, volume 26. Springer.

Lifka, D. A. (1998). *An extensible job scheduling system for massively parallel processor architectures*. Illinois Institute of Technology.

Lopes, R. V. and Menascé, D. (2016). A Taxonomy of Job Scheduling on Distributed Computing Systems. *IEEE Trans. on Parallel and Distrib. Systems*, 27(12):3412–3428.

Naghshnejad, M. and Singhal, M. (2018). Adaptive Online Runtime Prediction to Improve HPC Applications Latency in Cloud. In *11th International Conference on Cloud Computing*, pages 762–769. IEEE.

Nichols, D., Marathe, A., Shoga, K., Gamblin, T., and Bhatele, A. (2022). Resource Utilization Aware Job Scheduling to Mitigate Performance Variability. In *IEEE International Parallel and Distributed Processing Symposium*, pages 335–345.

Nunes, A. L., Portella, F., Estrela, P., Malini, R., Lopes, B., Bittencourt, A., Leite, G., Coutinho, G., and Drummond, L. (2023). Prediction of Reservoir Simulation Jobs Times Using a Real-World SLURM Log. In *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 49–60. SBC.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer.

Portella, F., Buchaca, D., Rodrigues, J. R., and Berral, J. L. (2022). TunaOil: A tuning algorithm strategy for reservoir simulation workloads. *Journal of Comput. Science*, 63.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Reuther, A., Byun, C., Arcand, W., Bestor, D., Bergeron, B., Hubbell, M., Jones, M., Michaleas, P., Prout, A., Rosa, A., and Kepner, J. (2018). Scalable system scheduling for HPC and big data. *Journal of Parallel and Distributed Computing*, 111:76–92.

Simakov, N. A., Innus, M. D., Jones, M. D., DeLeon, R. L., White, J. P., Gallo, S. M., Patra, A. K., and Furlani, T. R. (2018). A Slurm Simulator: Implementation and Parametric Analysis. In *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, pages 197–217. Springer.

Tanash, M., Dunn, B., Andresen, D., Hsu, W., Yang, H., and Okanlawon, A. (2019). Improving HPC System Performance by Predicting Job Resources via Supervised Machine Learning. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines*, pages 1–8. ACM.

Tsafrir, D., Etsion, Y., and Feitelson, D. G. (2007). Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):789–803.

Wang, H., Dai, Y.-Q., Yu, J., and Dong, Y. (2021). Predicting running time of aerodynamic jobs in HPC system by combining supervised and unsupervised learning method. *Advances in Aerodynamics*, 3(1).

Witt, C., Bux, M., Gusew, W., and Leser, U. (2019). Predictive performance modeling for distributed batch processing using black box monitoring and machine learning. *Information Systems*, 82:33–52.

Yang, W., Liao, X., Dong, D., and Yu, J. (2023). Exploring job running path to predict runtime on multiple production supercomputers. *Journal of Parallel and Distributed Computing*, 175(C):109—-120.

Yoo, A. B., Jette, M. A., and Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer.