Efeito das Políticas de Escalonamento do Linux na Execução Sustentável de um Modelo de RNN LSTM*

Davi Lopes Lemos[1], **Naylor Bastiani Perez**[1], **Leonardo Bidese de Pinho**[1]

[1]Núcleo de Pesquisa em Pecuária de Precisão (NP3) - UNIPAMPA Campus Bagé

{davilemos.aluno, leonardopinho}@unipampa.edu.br, naylor.perez@embrapa.br

Resumo. *Este trabalho avalia o impacto das políticas de escalonamento do Linux na execução de um modelo de RNN LSTM para estimar massa de forragem no Bioma Pampa, comparando FIFO, RR, Batch, Idle e Other. A análise com Perf indicou melhor desempenho da RR e, em diferentes cenários, menor consumo energético, sugerindo que ajustes no SO tornam a computação mais eficiente e sustentável, reduzindo a pegada de carbono.*

1. Introdução

A Inteligência Artificial (IA) expandiu-se rapidamente e consolidou-se como forma de automatizar tarefas nas quais a atuação humana era predominante, promovendo avanços em setores como saúde, indústria e, especialmente, agricultura. Contudo, está associada a altos custos computacionais e elevado consumo energético [Strubell et al. 2019], com impactos negativos ao meio ambiente e social, em desacordo com as boas práticas Ambientais, Sociais e de Governança (ESG), como a significativa emissão de Gases de Efeito Estufa. Assim, para ampliar a adoção da computação verde, torna-se central investigar estratégias que tornem a execução desses algoritmos mais eficiente, incorporando a sustentabilidade da solução à sua análise de desempenho [Schwartz et al. 2020].

Entre os contextos de uso da IA, destaca-se a agricultura digital, em especial a pecuária de precisão. Nessa perspectiva, este estudo analisa uma aplicação para predição da massa de forragem em unidades de manejo, por meio de uma Rede Neural Recorrente (RNN) com arquitetura *Long Short-Term Memory* (LSTM). Como o desempenho dessa aplicação também depende do gerenciamento dos recursos computacionais, torna-se relevante observar o papel do Sistema Operacional (SO), responsável por gerenciar o *hardware* do computador. Assim, este trabalho compara o efeito de diferentes políticas de escalonamento do Linux, com uma ou mais instâncias concorrentes do modelo, tendo como métricas principais o tempo de execução e eventos de hardware para identificar a configuração que favoreça menor pegada de carbono, considerando o consumo energético.

2. Fundamentação Teórica

Sob uma perspectiva de multiprogramação, o SO compartilha o tempo da CPU por meio do escalonador, que define qual processo da fila de prontos executa e pode interrompê-lo por preempção. No Linux, as tarefas são organizadas em classes (modelo de *Multiple Feedback Queues*) e seguem a hierarquia `SCHED_DEADLINE > SCHED_FIFO > SCHED_RR > SCHED_OTHER`, `SCHED_BATCH` e `SCHED_IDLE`. Assim, classes de

*Apoio da CAPES - Código de Financiamento 001

tempo real (SCHED_DEADLINE, SCHED_FIFO e SCHED_RR) têm precedência sobre tarefas das demais classes [Maziero 2019].

As políticas privilegiam critérios distintos: SCHED_DEADLINE atende tarefas periódicas com *Earliest Deadline First*, SCHED_FIFO mantém prioridade fixa sem *quantum*, SCHED_RR usa preempção por tempo com *quantum* entre 10ms e 200ms, SCHED_OTHER é a política interativa padrão com prioridades dinâmicas, SCHED_BATCH atende tarefas *CPU-bound*, e SCHED_IDLE executa apenas quando o sistema está ocioso [Maziero 2019]. Quanto às prioridades, políticas de tempo real (como FIFO e RR) usam escala de 1 a 99 (valores maiores indicam maior precedência, por exemplo prioridade 50 sobre 1), enquanto as demais usam a escala *nice* de -20 a +19 (valores maiores indicam menor prioridade) [Maziero 2019]. Modelos em Python, em geral, não operam em SCHED_DEADLINE devido ao uso dinâmico de *threads*, que pode conflitar com escalonamento determinístico e causar esgotamento de recursos, bloqueios e falhas.

3. Material e Métodos

Realizou-se uma pesquisa exploratória experimental destinada a capturar o impacto das diferentes políticas de escalonamento do Linux, bem como da prioridade (quando aplicável), no contexto de um modelo voltado à predição da massa de forragem em áreas de pastejo. A estimativa, expressa em kg/ha, é obtida a partir de séries de dados meteorológicos e ambientais, com a qualidade da predição avaliada pelo erro médio quadrático. O modelo foi implementado em Python utilizando as bibliotecas TensorFlow e Keras, com dados previamente normalizados e processados sequencialmente, com a arquitetura da rede definida empiricamente, composta por três camadas LSTM de 30, 15 e 1 neurônios, função de ativação tanh e treinamento com 5000 épocas [Lemos et al. 2025].

A seleção das políticas foi feita com a ferramenta chrt, capaz de manipular atributos de escalonamento do programa (aqui, a máquina Python). Utilizou-se o comando padrão "sudo chrt [opção de política] [prioridade] [programa]", em que a prioridade varia de um a 99 para políticas de tempo real e nos demais casos é zero, mantendo-se a prioridade nice do processo. Por sua vez a política é definida pelos parâmetros -f (FIFO), -r (RR), -b (Batch), -i (Idle) e -o (Other). Assim, ao disparar o programa com chrt, o processo inicia no regime da política escolhida.

Os diferentes testes foram organizados em *scripts shell* para evidenciar o impacto das políticas, de forma que nos cenários com mais de uma instância disparou-se as execuções em concorrência. O primeiro cenário avaliou a influência da prioridade em FIFO e RR, com três níveis: 1, 50 e 99. Na sequência, realizou-se um teste com cinco instâncias concorrentes, cada uma com uma política distinta (FIFO com prioridade 99, RR com prioridade 99, Other, Batch e Idle), visando avaliar o efeito da hierarquia de escalonamento quando políticas diferentes compartilham os mesmos recursos computacionais. Por fim, analisou-se a execução de uma, duas e quatro instâncias de uma mesma política concorrendo por recursos, para indicar o potencial de escalabilidade em cenários de pecuária de precisão, nos quais diferentes unidades de manejo demandam previsões individualizadas de massa de forragem e, portanto, múltiplas instâncias em concorrência. Para a instrumentação, utilizou-se o Perf, ferramenta do Linux, com dez repetições (-r 10) em todos os cenários, observando-se, para o tempo de execução, sobretudo a média e, de modo complementar, o desvio padrão (σ). Além do tempo, o campo de

análise inclui eventos de *hardware* como trocas de contexto, ciclos, instruções, falhas de página, *branches* e *branch-misses*. Quanto ao consumo energético, adotou-se procedimento também baseado no `perf` para acesso aos contadores RAPL, via "`perf stat -e power/energy-pkg/`", monitorando o consumo global da CPU e subtraindo o consumo em ociosidade (*baseline*) para melhor estimativa. Para simular essa condição, empregou-se o comando `sleep`, com duração equivalente ao tempo médio de execução do experimento. Os testes foram executados em microprocessador *multicore* AMD Ryzen 7 5700U (até 4,3 GHz), 8 núcleos e 16 *threads*, GPU integrada AMD Radeon Graphics 8, 12 GB de memória DDR4 e sistema operacional Ubuntu 22.04.4 LTS.

4. Resultados e Discussões

A experimentação voltada à comparação da execução de três instâncias com mesma política de escalonamento, variando entre FIFO e RR, com diferentes níveis de prioridade, representada na Tabela 1, evidencia que o tempo de execução, para FIFO, sofre redução de aproximadamente 3% entre a maior e menor prioridade, enquanto para RR esta diferença aumenta, sendo cerca de 11%. Além disso, comparando-se da menor para a maior prioridade, é possível verificar considerável redução na quantidade de trocas de contexto e migrações de CPU para ambas políticas, assim como no número de instruções e ciclos para RR. De forma complementar, a tabela ainda apresenta o σ do tempo de execução para cada prioridade. Observou-se que, em ambos os casos, há tendência de redução no consumo à medida que a prioridade aumenta, com RR sendo a mais econômica.

Tabela 1. Impacto da prioridade para FIFO e RR no desempenho

Política	Prioridade	Trocas de Contexto ($\times 10^6$)	Migrações de CPU ($\times 10^6$)	Ciclos ($\times 10^9$)	Instruções ($\times 10^9$)	Média de CPU	Tempo de execução (s)
FIFO	1	1,249	364,479	616,377	480,719	1,29	143,21 (σ=0,521)
	50	1,196	275,087	621,320	484,078	1,322	140,06 (σ=0,147)
	99	1,152	161,825	625,356	485,282	1,342	139,66 (σ=0,0668)
RR	1	1,245	368,902	600,649	479,142	1,289	137,35 (σ=0,649)
	50	1,184	266,148	606,759	481,999	1,319	134,75 (σ=0,195)
	99	1,033	144,857	555,011	437,547	1,34	121,68 (σ=0,11)

Analisou-se o tempo de execução com cinco instâncias concorrentes, cada uma sob uma política. Os tempos médios obtidos foram de 149,60 s para RR, 149,64 s para FIFO, 163,05 s para Other, 163,11 s para Batch e 164,76 s para Idle, com σ de 0,715, 0,713, 3,9, 3,93 e 4,16, respectivamente. Assim, as políticas de tempo real (FIFO e RR) destacaram-se frente às demais (Other, Batch e Idle), com redução aproximada de 8,5%. Do ponto de vista energético, embora a potência média varie pouco entre as políticas, a diferença no tempo de execução impacta a energia total, e Idle, Other e Batch operaram com demanda aproximadamente 10% maior em relação às políticas de tempo real.

Para a escalabilidade, a Figura 1 mostra aumento claro do tempo ao passar de uma para quatro instâncias, com acréscimo médio acima de 20% e variações relevantes entre políticas. De forma complementar, o σ indica baixa dispersão em termos de tempo de execução no cenário de 1 instância e maior variabilidade em parte dos cenários com 2 e 4 instâncias. As variações da RR indicam crescimento mais moderado em prioridade 1, enquanto Batch e RR com prioridade 50 apresentam aumentos mais acentuados, sugerindo maior sensibilidade à carga. A Figura 2 aponta comportamento semelhante para o crescimento energético, também próximo de 20% em média, com RR prioridade 50 entre os

maiores aumentos e FIFO prioridade 50 e Other mais estáveis. Esses resultados reforçam que deve-se analisar desempenho e energia em conjunto na escolha da política.

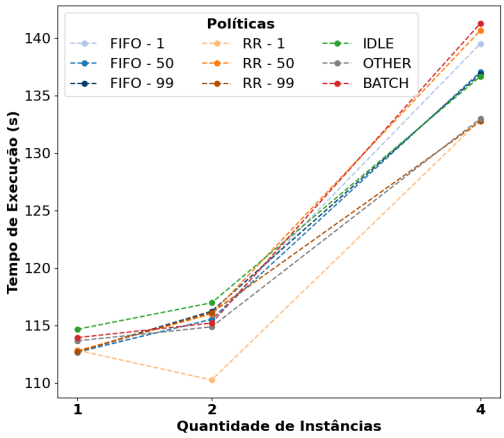

Figura 1. Tempo de execução

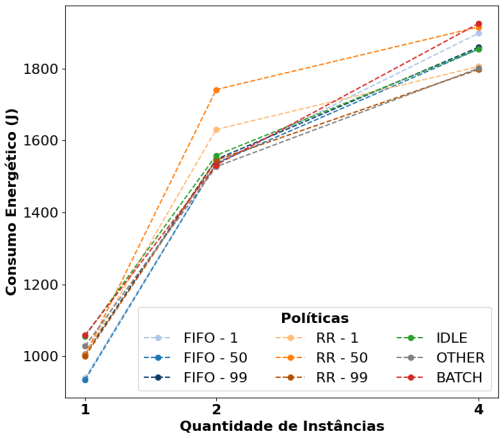

Figura 2. Consumo energético

5. Considerações Finais

Este estudo investigou o impacto das políticas de escalonamento do Linux na execução de um modelo de RNN LSTM para predição de massa de forragem no Bioma Pampa, evidenciando influência direta no desempenho em cenários com concorrência. A política RR apresentou o melhor resultado, com menor tempo de execução e menor incidência de eventos de *hardware*. O FIFO apareceu como segundo melhor, enquanto Other, Batch e Idle tiveram desempenho inferior e maior gasto energético total. De modo geral, a redução do tempo de execução esteve associada à redução do consumo, indicando que a escolha da política deve considerar conjuntamente eficiência computacional e energética.

Como limitação e oportunidade de melhoria, reconhece-se a relevância de ampliar o número de repetições, visando refinar o rigor estatístico dos achados. Como continuidade, também é pertinente exercitar esse método em diferentes modelos e aplicações, para avaliar sua aplicabilidade em contextos diversos, visto que o uso sustentável de recursos computacionais é necessário em todas as áreas da computação. Ainda assim, os resultados demonstram que ajustes em nível de SO são viáveis para melhorar desempenho e eficiência energética, reforçando seu potencial para fomentar uma computação mais sustentável, para o modelo analisado e outros com características similares.

Referências

Lemos, D., Durgante, B., Correia, M., Perez, N., and Pinho, L. (2025). Instrumentação das Técnicas Árvore de Decisão e KerasTuner para Autoajuste de Redes Neurais Recorrentes GRU na Perspectiva de Computação Sustentável. In *Anais da XXV ERAD/RS*.

Maziero, C. A. (2019). *Sistemas Operacionais: Conceitos e Mecanismos*. Universidade Federal do Paraná, Departamento de Informática (UFPR), Curitiba, Brasil.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green AI. *Communications of the ACM*.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th ACL*.