Melhorando a Eficiência Energética na Execução de LLMs via Controle de Potência de GPUs Orientado por SLAs

Alex F. R. Trajano[1], Crislane Costa[1], Francisco V. J. Nobre[2], Rafael L. Gomes[2]

[1]Instituto Atlântico – Fortaleza – Ceará – Brasil

alex_ferreira@atlantico.com.br, crislane_costa@atlantico.com.br

[2]Universidade Estadual do Ceará (UECE) – Fortaleza – Ceará – Brasil

valderlan.nobre@aluno.uece.br, rafa.lopes@uece.br

Abstract. *LLM inference services incur high energy consumption on GPUs while operating under stringent SLAs. This paper proposes an adaptive and reactive GPU power-control algorithm that directly leverages hardware* power capping *mechanisms, dynamically adjusting power limits based on SLA compliance. The approach is lightweight, operates at runtime, and does not rely on offline profiling or predictive models. An experimental evaluation in a real environment with multiple LLM instances deployed across multiple GPUs shows that fixed power caps can reduce energy consumption but cause severe performance degradation under higher load, whereas the dynamic controller consistently lowers energy usage relative to the baseline and mitigates abrupt performance degradation.*

Resumo. *Serviços de inferência de LLMs apresentam elevado consumo energético em GPUs e operam sob SLAs exigentes. Este trabalho propõe um algoritmo adaptativo e reativo de controle de potência para GPUs, que atua diretamente sobre mecanismos de* power capping*, ajustando dinamicamente o limite de potência com base no cumprimento de SLAs. A proposta é de baixo overhead, opera em tempo de execução e não depende de perfis offline ou modelos preditivos. A avaliação experimental em um ambiente real com múltiplas instâncias de LLM em múltiplas GPUs demonstra que limites de potência fixos podem reduzir o consumo energético, porém causam degradações severas de desempenho sob maior carga, enquanto o controle dinâmico reduz consistentemente o consumo em relação ao* baseline *e mitiga degradações abruptas de desempenho.*

1. Introdução

Os *Large Language Models (LLMs)* consolidaram-se como peças fundamentais para uma ampla variedade de aplicações modernas, incluindo assistentes conversacionais, motores de busca, geração de código, sistemas de recomendação, análise de textos em larga escala, dentre outros exemplos. O crescimento contínuo dessas aplicações tem impulsionado um corpo significativo de pesquisas voltadas à otimização de desempenho, escalabilidade e confiabilidade de LLMs, abrangendo desde treinamento e fine-tuning até inferência em produção [Rostam et al. 2024, Miao et al. 2025]. Nesse contexto, a eficiência da execução de LLMs tornou-se um fator crítico para a operação sustentável de serviços baseados em modelos generativos.

Em ambientes de produção, os serviços de LLMs são expostos a um alto volume de requisições que impõem cargas de trabalho dinâmicas e concorrentes, com requisitos estritos de latência [Liu et al. 2024]. Evidências recentes indicam que, quando considerados em regime contínuo, os custos energéticos associados à inferência representam uma parcela expressiva do custo operacional total de sistemas baseados em LLMs [Niu et al. 2025]. Em particular, medições detalhadas em servidores modernos mostram que as GPUs concentram a maior parte do consumo energético durante a inferência, superando CPU, memória e demais componentes do sistema [Samsi et al. 2023, Stojkovic et al. 2025a].

Serviços de LLM têm sido tradicionalmente projetados com foco em maximizar métricas de desempenho, como *throughput* (*tokens* por segundo), *time-to-first-token* (TTFT), *time-per-output-token* (TPOT) e latência fim a fim. Revisões recentes da literatura indicam que a maior parte das otimizações propostas para LLMs enfatiza eficiência computacional e escalabilidade do pipeline, com pouca exploração de mecanismos de controle energético no nível do hardware [Miao et al. 2025, Rostam et al. 2024]. Mesmo estudos que mensuram o consumo de energia durante a inferência tendem a analisá-lo sob configurações estáticas, comparando *engines* ou fases do pipeline, sem considerar a interação entre o comportamento do *workload* e os mecanismos de gerenciamento de potência expostos pelo hardware, em particular pelas GPUs modernas [Niu et al. 2025].

Paralelamente, estudos experimentais em sistemas de LLMs demonstram que o desempenho não escala de forma linear com a potência elétrica aplicada à GPU. Resultados recentes indicam que a redução do limite de potência (*power cap*) de GPUs de alto desempenho (por exemplo, de 450W para 300W) pode resultar em quedas modestas de *throughput*, tipicamente em torno de 5%, enquanto o consumo energético total é reduzido em mais de 30% [Gogineni et al. 2025]. Esses resultados reforçam a observação de que reduções controladas de potência podem ser aplicadas sem perdas proporcionais de desempenho em cenários típicos de inferência de LLMs.

Apesar da disponibilidade de mecanismos de *power capping* em hardware comercial atual, sistemas que fornecem LLMs ainda operam majoritariamente com configurações estáticas de potência, frequentemente provisionadas para o pior caso, ou sequer exploram esses mecanismos de forma sistemática. Essa prática ignora a variabilidade temporal da carga, bem como a possibilidade de explorar, de forma controlada, as não proporcionalidades entre consumo energético e desempenho, especialmente em cenários nos quais Acordos de Nível de Serviço (SLAs) podem ser atendidos mesmo sob restrições explícitas de potência impostas às GPUs.

Diante desse contexto, este trabalho apresenta um algoritmo adaptativo de controle de potência para GPUs em ambientes de serviços de LLMs, com o objetivo de reduzir o consumo energético sem violar Acordos de Nível de Serviço (SLAs). O algoritmo ajusta dinamicamente o limite de potência das GPUs em tempo de execução, considerando o estado do sistema e as exigências do *workload*, em um cenário no qual múltiplas instâncias de um LLM são executadas em GPUs distintas e atendem conjuntamente a um SLA global. A proposta explora de forma controlada a relação não linear entre potência e desempenho observada em GPUs modernas.

As principais contribuições deste trabalho incluem: (i) um controlador adaptativo

e reativo de *power cap* por GPU, não intrusivo ao sistema de inferência, que ajusta dinamicamente o limite de potência em tempo de execução a partir de telemetria local da GPU e do cumprimento do SLA, combinando uma estratégia *Multiplicative Increase / Additive Decrease* (MIAD) com um mecanismo de *snap* para acelerar a adaptação; e (ii) uma avaliação experimental controlada em infraestrutura real multi-GPU (8× NVIDIA H200), comparando limites de potência fixos e o controlador proposto sob diferentes regimes de concorrência e com balanceamento de carga fixo, quantificando o trade-off entre consumo energético e latência de cauda (TTFT p95) sob um SLA global compartilhado.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados e o estado da arte. A Seção 3 descreve o modelo de sistema e o algoritmo de controle de limite potência proposto. A Seção 4 apresenta a metodologia experimental. A Seção 5 apresentamos e discutimos os resultados obtidos. Por fim, a Seção 6 conclui o trabalho e aponta direções para pesquisas futuras.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados que tratam de desempenho e eficiência energética em serviços de inferência de LLMs, organizados segundo o nível de atuação no sistema, desde balanceamento de carga e gerenciamento em nível de datacenter até mecanismos de controle energético no hardware.

Jain et al. [Jain et al. 2025] propõem um balanceador de carga sensível ao desempenho para serviços de inferência de LLMs executados em múltiplas instâncias homogêneas, com o objetivo de reduzir a latência fim a fim em cenários de cargas mistas. O trabalho parte da observação de que as fases de *prefill* e *decode* apresentam características computacionais distintas e que a coexecução de requisições heterogêneas pode introduzir interferências significativas, degradando métricas como latência média e TTFT. Para mitigar esse efeito, os autores propõem um roteador baseado em *reinforcement learning*, que combina a predição do comprimento da saída gerada pela LLM com um modelo analítico do impacto da execução conjunta de diferentes *workloads*, alcançando reduções de até 11% na latência média em comparação com estratégias tradicionais de roteamento.

Stojković et al. [Stojkovic et al. 2025b] investigam os desafios de gerenciamento térmico e energético de clusters de GPUs dedicados à inferência de LLMs em *datacenters* de larga escala e propõem o *TAPAS*. O trabalho introduz um *framework* sistêmico que integra alocação de máquinas virtuais com consciência térmica e energética, roteamento adaptativo de requisições e reconfiguração dinâmica de parâmetros operacionais, visando aumentar a capacidade de *oversubscription* de potência e refrigeração e, consequentemente, reduzir o *Total Cost of Ownership* (TCO) do datacenter. A proposta explora explicitamente a não linearidade entre potência aplicada às GPUs e desempenho de inferência, utilizando múltiplos mecanismos de ajuste, como paralelismo de modelo, tamanho de *batch*, quantização e seleção dinâmica do tamanho do modelo.

Stojković et al. [Stojkovic et al. 2025a] apresentam o *DynamoLLM*, um *framework* de gerenciamento energético para clusters de inferência de LLMs que explora a heterogeneidade das requisições e a variabilidade temporal da carga sob restrições de Objetivos de Nível de Serviço (SLOs) de latência. O sistema opera no nível do cluster e baseia-se na construção offline de perfis detalhados de energia e desempenho, modelando a relação entre carga, configuração do sistema e consumo energético. A partir desses

perfis, o *DynamoLLM* seleciona dinamicamente entre múltiplos mecanismos de controle, incluindo escalonamento do número de instâncias ativas, ajustes de paralelismo de modelo e *Dynamic Voltage and Frequency Scaling* (DVFS), com o objetivo de minimizar o consumo energético mantendo o atendimento aos SLOs.

Kakolyris et al. [Kakolyris et al. 2024] propõem um mecanismo de otimização energética em nível de hardware que ajusta dinamicamente a frequência de operação das GPUs durante a inferência de LLMs com base no cumprimento de SLOs, particularmente percentis de latência fim a fim. A abordagem explora a natureza autorregressiva da inferência para intervir no ritmo de geração de *tokens*, utilizando um laço de controle de granularidade fina orientado por um modelo preditivo treinado offline, capaz de estimar latências de cauda a partir da taxa de chegada de requisições e da configuração da GPU. Os autores demonstram reduções de consumo energético entre 22,8% e 45,5%, dependendo da GPU avaliada e da rigidez do SLO imposto.

Apesar dos avanços apresentados, esses trabalhos compartilham limitações importantes. Abordagens focadas em balanceamento de carga concentram-se exclusivamente no roteamento e escalonamento entre instâncias, sem mecanismos explícitos de controle energético, tratando o consumo como consequência indireta da organização do pipeline de inferência [Jain et al. 2025]. Soluções em nível de *datacenter* e *cluster* assumem conhecimento detalhado da infraestrutura física, dependem de perfis *offline* e previsões de carga, além de reconfigurações frequentes do sistema, o que introduz maior complexidade operacional e sobrecusto [Stojkovic et al. 2025b, Stojkovic et al. 2025a]. Por sua vez, técnicas baseadas em DVFS atuam indiretamente sobre o consumo energético e requerem modelos preditivos treinados *offline* para estimar latências de cauda, o que pode limitar sua robustez frente a mudanças dinâmicas no comportamento das cargas [Kakolyris et al. 2024].

A proposta apresentada neste trabalho foca em um mecanismo de controle adaptativo de potência leve, reativo e de baixo *overhead*, operando diretamente sobre os limites de potência das GPUs durante a inferência de LLMs. Ao explorar explicitamente a relação não linear entre potência e desempenho no hardware, o método permite reduzir o consumo energético sem violar SLAs globais compartilhados entre múltiplas instâncias, dispensando reconfigurações estruturais do pipeline de inferência, modelos preditivos complexos ou conhecimento detalhado da topologia do datacenter.

3. Modelo do Sistema e Proposta

Esta seção formaliza o modelo do sistema e descreve a proposta deste trabalho. Inicialmente, são definidas as métricas de qualidade de serviço e o SLA que orientam o controle, seguidos da modelagem do ambiente de execução e das restrições operacionais assumidas. Em seguida, é apresentado o algoritmo adaptativo de controle de potência proposto, que ajusta dinamicamente os limites de potência das GPUs com base no cumprimento das restrições de SLO e na telemetria de execução, explorando a não linearidade entre consumo energético e desempenho durante a inferência de LLMs. A Figura 1 apresenta uma visão geral do sistema, incluindo elementos externos à proposta.

3.1. Definição de SLA

Neste trabalho, distinguimos explicitamente os conceitos de Objetivo de Nível de Serviço (SLO) e Acordo de Nível de Serviço (SLA). Um SLO define um objetivo quantitativo

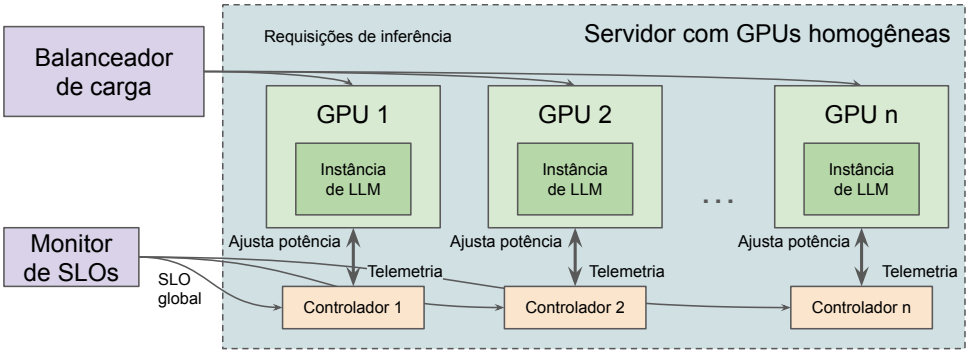


Figura 1. Visão geral do sistema.

associado a uma métrica específica de desempenho (por exemplo, um limite máximo aceitável para a latência p95), enquanto o SLA corresponde ao contrato de qualidade de serviço do sistema, que pode ser composto por um ou mais SLOs simultaneamente. Assim, as métricas e seus respectivos limites especificam SLOs individuais, e o SLA é considerado satisfeito quando todos os SLOs relevantes são atendidos.

O sistema é avaliado sob um SLA definido por métricas de desempenho observadas em janelas temporais discretas k (com duração fixa). De forma abstrata, considera-se um conjunto de métricas $\mathcal{M}$ e limites associados θ_m, que é definida pela Equação 1:

$$\text{SLO}_m(k) \leq \theta_m, \quad \forall m \in \mathcal{M}. \tag{1}$$

O SLA do sistema é considerado satisfeito em uma janela k quando todas as restrições acima são atendidas.

Denotaremos por $s(k)$ a observação da métrica associada a um SLO na janela k, e por θ o limite correspondente definido para esse SLO. A escolha concreta da métrica e do respectivo limite é apresentada na Seção 4, para fins de avaliação da proposta.

3.2. Modelo do Sistema

Considera-se um servidor físico com um conjunto de GPUs homogêneas $G = \{g_1, g_2, \ldots, g_K\}$. Um único modelo de LLM é servido. Para cada GPU $g \in G$, executa-se uma instância dedicada do servidor de inferência associada exclusivamente àquela GPU. Assim, cada GPU possui uma fila lógica independente e não há particionamento de um mesmo modelo entre múltiplas GPUs.

Cada GPU g possui um limite de potência em Watts que é diretamente configurável ($c_g(t) \in [c_g^{min}, c_g^{max}]$), sendo $c_g(t)$ aplicável por meio da interface do *driver* da GPU[1]. Além disso, estabelece-se um limite mínimo operacional adicional $C_{\min}$ (por exemplo, 200W). O limite mínimo efetivo considerado pelo controlador é então definido pela Equação 2. Denotamos por c_g^{def} o limite de potência padrão da GPU g, que em hardware comercial corresponde tipicamente ao limite máximo de potência suportado pela GPU, isto é, $c_g^{def} = c_g^{max}$. Assume-se um regime de operação viável no qual, sem limitação de potência explícita (isto é, com $c_g(t) = c_g^{def}$), o SLA é satisfeito para todas

[1]Em GPUs NVIDIA, o utilitário *nvidia-smi* é utilizado diretamente.

as janelas k. Quando conveniente, utilizamos $c_g(k)$ para denotar o valor do limite de potência amostrado ao final da janela de controle k.

$$\underline{c}_g = \max(c_g^{min}, C_{\min}) \qquad (2) \qquad\qquad E = \sum_{g \in G} \int_0^T P_g(t)\, dt \qquad (3)$$

A definição de energia total consumida ao longo de um horizonte T é dada pela Equação 3, sendo que o problema consiste em minimizar E sujeito ao cumprimento das restrições de SLO em todas as janelas k.

O sistema recebe uma sequência de requisições $\mathcal{R} = \{r_1, r_2, \dots\}$; cada requisição r tem instante de chegada $a(r)$, número de *tokens* de entrada $x(r)$ e um limite superior de *tokens* de saída $y(r)$. Os mecanismos internos de operação do servidor de inferência são mantidos fixos e inalterados.

3.3. Algoritmo Adaptativo de Controle de Potência Máxima de GPU

Este trabalho propõe um controlador adaptativo de potência que opera de forma independente em cada GPU e é executado periodicamente em janelas de controle de duração fixa T_c. Em cada janela, o controlador toma decisões com base tanto no cumprimento do objetivo de desempenho estabelecido quanto no estado corrente de execução da GPU.

De forma mais específica, o cada controlador observa o valor do SLO do sistema na janela k, denotado por $s(k)$, e o compara ao limite alvo θ. Essa observação é **global**, isto é, é agregada sobre todas as requisições atendidas no sistema na janela k. A observação é combinada com a telemetria **local** da GPU g, incluindo o consumo instantâneo (*power draw*) $d_g(k)$, o nível de utilização $u_g(k)$ e o limite de potência atualmente configurado $c_g(k)$. A partir dessas observações, cada controlador ajusta o limite de potência de cada GPU dinamicamente.

O controlador segue um padrão de controle do tipo *Multiplicative Increase / Additive Decrease* (MIAD). Em janelas nas quais o SLO é violado, o limite de potência é aumentado de forma multiplicativa, permitindo uma reação rápida a degradações de desempenho. Quando o SLO é satisfeito com folga sustentada, o limite de potência é reduzido gradualmente por meio de decrementos aditivos, promovendo estabilidade. Adicionalmente, quando o limite configurado encontra-se significativamente acima do consumo instantâneo observado, o algoritmo aplica um ajuste direto (*snap*) que aproxima o limite de potência do nível efetivamente demandado, acelerando a convergência para regimes energeticamente eficientes.

Para tornar o controle robusto a diferentes limites de potência, o algoritmo opera sobre uma variável normalizada, que é definida pela Equação 4:

$$p_g(k) = \frac{c_g(k) - \underline{c}_g}{c_g^{max} - \underline{c}_g}, \qquad (4)$$

que representa a posição relativa da potência vigente entre o mínimo e o máximo efetivos. O controlador utiliza os seguintes parâmetros (constantes): passo aditivo α para redução gradual de potência, fator multiplicativo $\beta \in (0, 1)$ para aumento agressivo de potência, margem de folga ε para caracterizar o SLO como sustentado, limiar de utilização $U_{\min}$ para evitar reduzir a potência quando a GPU está ociosa, além do limiar G_{snap} e uma margem M_{snap} para identificar a necessidade de aplicar *snap*.

O mecanismo de *snap* é introduzido para acelerar a convergência do controle em situações nas quais o limite de potência configurado encontra-se significativamente acima do consumo efetivo da GPU. Formalmente, define-se o *gap* de potência na janela k pela Equação 5onde $c_g(k)$ é o limite configurado e $d_g(k)$ o consumo instantâneo observado. Quando o SLO é satisfeito com folga sustentada e $g_g(k)$ excede um limiar predefinido G_{snap}, o algoritmo interpreta essa diferença como indício de superprovisionamento energético e aplica um ajuste direto no limite de potência.

$$g_g(k) = c_g(k) - d_g(k) \qquad (5) \qquad\qquad c'_g(k) = d_g(k) + M_{\text{snap}} \qquad (6)$$

O novo limite de potência aplicado pelo mecanismo de *snap* é definido pela Equação 6, onde M_{snap} é uma margem de segurança que absorve variações de curto prazo no consumo. Diferentemente do decréscimo aditivo, que reduz o limite de potência de forma incremental, o *snap* permite eliminar rapidamente folgas excessivas, evitando que o sistema opere por múltiplas janelas com um limite de potência desnecessariamente elevado. A aplicação do *snap* é condicionada à observação simultânea de folga de SLO e utilização significativa da GPU, de modo a preservar a estabilidade do controle e evitar reduções agressivas em regimes de baixa carga.

O Algoritmo 1 apresenta o pseudocódigo do controlador proposto, destacando os regimes de decisão e os mecanismos de ajuste do limite de potência empregados para reduzir o consumo energético sem violar o SLO. De modo simplificado, dada uma janela k o algoritmo atua da seguinte forma:

- **Violação de SLO:** se $s(k) > \theta$, aumenta-se o limite da potência da GPU agressivamente via aumento multiplicativo na variável normalizada.
- **Folga sustentável:** se $s(k) < (1 - \varepsilon)\theta$ e $u_g(k) \geq U_{\min}$, tenta-se reduzir o limite da potência da GPU. Se houver grande diferença entre o limite da potência configurado e o consumo instantâneo, aplica-se o *snap*; caso contrário, reduz-se gradualmente por passo aditivo.
- **Região neutra:** caso contrário, não se altera o limite da potência da GPU a fim de evitar oscilações.

Após a decisão, aplica-se um *clamp* ao limite de potência para o intervalo $[\underline{c}_g, c_g^{max}]$ e arredondamento para múltiplos de 5 W, a fim de evitar micro gestão da potência.

4. Avaliação Experimental

Esta seção descreve o ambiente experimental, a arquitetura do sistema avaliado, a política de balanceamento de carga empregada, o *workload* utilizado, os regimes experimentais considerados, as métricas coletadas e o protocolo seguido nos experimentos. O objetivo é avaliar, de forma controlada e reprodutível, o compromisso entre eficiência energética e manutenção de SLA ao comparar limites de potência fixos com o controlador dinâmico de *power cap* proposto. Assim, a seguinte pergunta de pesquisa será avaliada:

> **RQ:** Entre a utilização de um *power cap* fixo e o uso de um algoritmo de controle dinâmico de limite potência, qual abordagem oferece o melhor compromisso entre redução do consumo energético e manutenção do SLA de latência (TTFT p95) em um serviço de inferência de LLM com múltiplas instâncias sob uma política de balanceamento de carga?

Algoritmo 1 Controle adaptativo de *power cap* por GPU

Require: Observação de SLO $s(k)$, alvo θ, telemetria $\{c, d, u\}$ da GPU
Require: Parâmetros α (passo), β (fator), ε (folga), $U_{\min}$, G_{snap}, M_{snap}, passo W_{step}
  $\underline{c} \leftarrow \max(c^{min}, C_{\min})$; $\overline{c} \leftarrow c^{max}$
  $c \leftarrow \text{clamp}(c, \underline{c}, \overline{c})$
  $p \leftarrow \dfrac{c - \underline{c}}{\overline{c} - \underline{c}}$ ▷ limite de potência normalizado em $[0, 1]$
  if $s(k) > \theta$ **then** ▷ violação de SLO
    $p \leftarrow \min\left(1,\ \dfrac{p}{\beta}\right)$ ▷ aumento multiplicativo
  else if $s(k) < (1 - \varepsilon)\theta$ **and** $u \geq U_{\min}$ **then** ▷ folga sustentável
    $gap \leftarrow c - d$
    if $gap \geq G_{\text{snap}}$ **then** ▷ limite muito acima do consumo, aplica *snap*
      $c' \leftarrow \text{clamp}(d + M_{\text{snap}},\ \underline{c},\ \overline{c})$
      $c' \leftarrow \text{round\_step}(c', W_{\text{step}})$
      return aplicar c'
    else
      $p \leftarrow \max(0,\ p - \alpha)$ ▷ decréscimo aditivo
    end if
  else
    return ▷ região neutra: não altera limite de potência
  end if
  $p \leftarrow \text{clamp}(p, 0, 1)$
  $c' \leftarrow \underline{c} + p(\overline{c} - \underline{c})$
  $c' \leftarrow \text{round\_step}(c', W_{\text{step}})$
  aplicar c'

A análise concentra-se simultaneamente no consumo energético das GPUs e no cumprimento de um SLA composto por um único SLO, definido sobre o TTFT no percentil 95. Essa métrica é particularmente relevante em serviços interativos baseados em LLMs, pois reflete a responsividade percebida pelo usuário e é sensível a efeitos de congestionamento e contenção de recursos, sendo amplamente adotada como métrica de cauda em sistemas de inferência [Liu et al. 2024].

4.1. Ambiente Experimental

Os experimentos foram conduzidos em um servidor HPE Cray XD670, equipado com 2 processadores Intel Xeon Platinum 8570, 2 TB de memória RAM e 8 GPUs NVIDIA H200 SXM. Cada GPU opera de forma dedicada, sem compartilhamento com outras instâncias de inferência.

O serviço de inferência utilizado é o Ollama versão 0.15.4, executando o modelo `qwen3:32b`. Cada GPU hospeda exatamente uma instância independente do servidor de inferência, consumindo 20GB dos 141GB de VRAM disponíveis. O *driver* NVIDIA utilizado é a versão 550.163.01, com CUDA 12.4. O balanceador de carga é executado no mesmo nó físico que as instâncias de inferência.

4.2. Política de Balanceamento de Carga

Como o sistema assume um conjunto de GPUs G e uma instância de inferência por GPU, é necessário um mecanismo reprodutível para distribuir as requisições dos clientes entre essas instâncias. Neste trabalho, adotamos uma política fixa de *Shortest Expected Delay*

(SED), em que o atraso esperado é aproximado pelo *backlog* de *tokens* pendentes de geração em cada GPU.

O balanceador mantém, para cada GPU g, um contador de *tokens* pendentes que aproxima a carga enfileirada para g. Para cada requisição em execução, considera-se uma estimativa fixa de 1000 *tokens* de saída; à medida que *tokens* são produzidos e recebidos pelo balanceador, esse contador é decrementado. Assim, o número total de *tokens* pendentes de uma GPU corresponde à soma, sobre suas requisições pendentes, do numero de *tokens* estimado, menos o número de *tokens* já emitidos em cada requisição. A estimativa de 1000 *tokens* é conservadora e foi escolhida para cobrir os comprimentos de saída típicos do *workload* utilizado.

Quando uma nova requisição chega, ela é roteada para a GPU com o menor número de *tokens* pendentes. Essa mesma política de balanceamento é utilizada em todos os regimes experimentais, de modo que o roteamento é mantido constante entre os cenários, permitindo comparar as políticas de gerenciamento de potência sob o mesmo mecanismo de distribuição de carga.

4.3. Workload e Geração de Carga

O *workload* experimental consiste num conjunto de entradas amostradas aleatoriamente a partir de três conjuntos de dados distintos: IMDB[2], ELI5[3] e SQuAD10[4]. A amostragem segue uma distribuição fixa de probabilidades ($\Pr[\text{IMDB}] = 0.5, \Pr[\text{ELI5}] = 0.3, \Pr[\text{SQuAD10}] = 0.2$). É construído um *prompt* para cada amostra seguindo a mesma metodologia descrita no trabalho de [Jain et al. 2025].

A carga é gerada ao balanceador de carga por múltiplos clientes independentes. Cada cliente envia 100 requisições de forma estritamente sequencial, sem paralelismo interno. O grau de concorrência do sistema é controlado pelo número de clientes simultâneos C, sendo considerados três regimes distintos ($C \in \{6, 8, 10\}$).

4.4. Regimes Experimentais

Os experimentos comparam dois tipos de gerenciamento de potência das GPUs:

Limite de potência fixo: O limite de potência da GPU é configurado estaticamente em valores $P \in \{700, 650, 600, \ldots, 250, 200\}$ W. Para cada valor de P, os experimentos são executados nos três regimes de concorrência definidos. A potência padrão (c_g^{def}) da GPU NVIDIA H200 é de 700W, valor utilizado como *baseline*.

Controle dinâmico de potência: O controlador proposto ajusta dinamicamente o *power cap* de cada GPU em tempo de execução, observando exclusivamente o cumprimento do SLO definido sobre o TTFT p95. As execuções com controle dinâmico também são realizadas para cada valor de C.

4.5. Métricas Coletadas

As métricas de desempenho e energia são coletadas por meio do Prometheus. Para cada inferência, é registrado um histograma para a métrica TTFT. Como dito anteriormente,

[2]Endereço: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
[3]Endereço: https://huggingface.co/datasets/sentence-transformers/eli5
[4]Endereço: https://rajpurkar.github.io/SQuAD-explorer/

a avaliação de SLA considera exclusivamente o TTFT no percentil 95. Os percentis são estimados a partir dos histogramas utilizando os mecanismos padrão de quantil do Prometheus, agregados globalmente independente da GPU que serviu uma requisição.

Além das métricas de latência, o sistema coleta continuamente o consumo de energia instantâneo de cada GPU, diretamente via telemetria de hardware e registrada no Prometheus. O consumo energético é avaliado pela integração do consumo ao longo do tempo para cada GPU, sendo a energia total do sistema dada pela soma da energia consumida por todas as GPUs estritamente dentro do período do experimento.

4.6. Protocolo Experimental

Cada combinação de parâmetros (P, C) é executada de forma isolada em 5 rodadas independentes, após uma fase de aquecimento para evitar efeitos de *cold start*. O mesmo protocolo é aplicado às execuções com controle dinâmico de potência. Este protocolo permite comparar diretamente limites de potência fixos e o controlador dinâmico sob condições idênticas de carga e roteamento, isolando o impacto do gerenciamento de potência no cumprimento do SLA e no consumo energético.

Para as execuções com controle dinâmico de potência, os parâmetros apresentados no Algoritmo 1 foram mantidos fixos ao longo de todos os experimentos. O passo aditivo de redução de potência foi definido como $\alpha = 0.03$, enquanto o fator de aumento multiplicativo foi $\beta = 0.4$. A margem de folga utilizada foi $\varepsilon = 0.1$. O limite mínimo de potência considerado pelo controlador foi $C_{\min} = 250$W, sendo aplicado na definição do limite mínimo efetivo $\underline{c}_g$. O mecanismo de *snap* foi configurado com limiar $G_{\text{snap}} = 30$W e margem $M_{\text{snap}} = 20$W. Os limites de potência aplicados pelo controlador são arredondados para múltiplos de $W_{\text{step}} = 5$W. O intervalo de atuação de cada controlador foi definido em $T_c = 5$ segundos, e a utilização mínima da GPU considerada pra redução de limite é $U_{\min} = 0.5$. Por fim, foi estabelecida uma janela $k = 1$ minuto para avaliação do SLO.

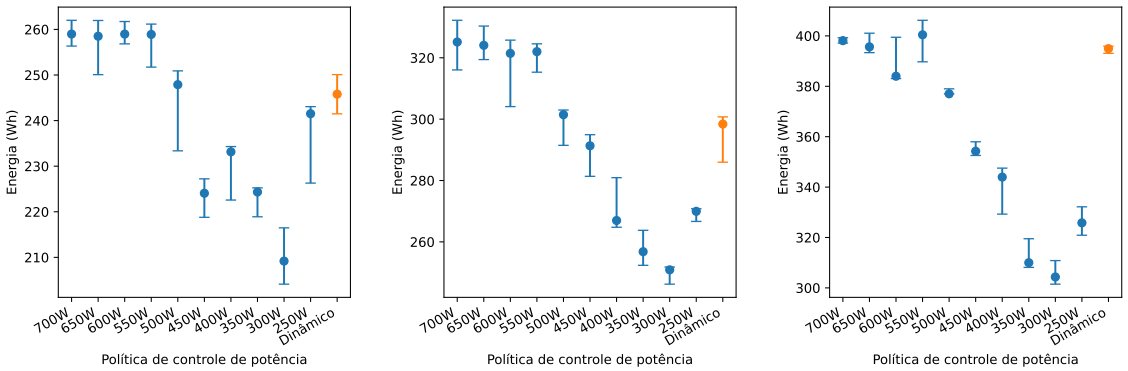
5. Resultados

Esta seção apresenta os resultados experimentais obtidos, comparando limites de potência fixos e o controlador dinâmico proposto sob diferentes regimes de concorrência. A análise foca no compromisso entre consumo energético das GPUs e cumprimento do SLA definido sobre o TTFT no percentil 95, conforme a pergunta de pesquisa estabelecida na Seção 4.

5.1. Consumo Energético sob Diferentes Regimes de Concorrência

A Figura 2 apresenta o consumo energético total do sistema para os três regimes de concorrência avaliados ($C \in \{6, 8, 10\}$), sob diferentes políticas de controle de potência.

Para limites de potência fixos, observa-se uma redução quase monotônica do consumo energético à medida que o *power cap* é reduzido, em todos os regimes de concorrência, até aproximadamente 300W. Abaixo desse valor, o consumo energético total deixa de diminuir de forma proporcional. Embora o limite de potência em 250W seja menor, a execução da inferência torna-se significativamente mais lenta, aumentando o tempo total de processamento e resultando em maior energia consumida ao longo do experimento quando comparado ao limite de 300W. Esse comportamento evidencia que



(a) $C = 6$ **(b)** $C = 8$ **(c)** $C = 10$

Figura 2. Consumo energético total sob diferentes políticas de controle de potência máxima, para distintos regimes de concorrência.

a minimização de potência instantânea não implica necessariamente na minimização da energia total.

O controle dinâmico apresenta, em todos os valores de C, um consumo energético consistentemente inferior ao *baseline* definido pelo limite padrão de 700W. Ainda que não busque minimizar o consumo energético absoluto, o consumo observado com o controlador dinâmico situa-se entre os valores obtidos com limites fixos moderados e agressivos. Esse comportamento reflete a natureza conservadora do algoritmo, que ajusta o limite de potência em função do cumprimento do SLO, mantendo margens de segurança para absorver variações de carga e evitar violações de SLA.

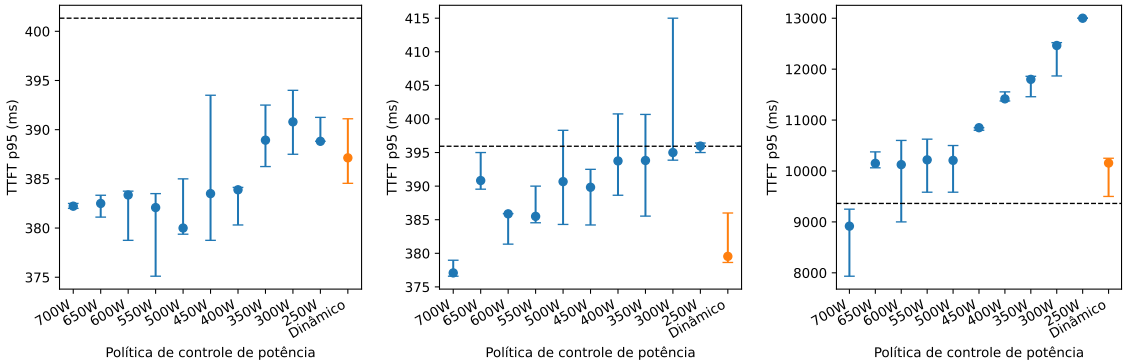
À medida que a concorrência aumenta, a diferença de consumo energético entre políticas fixas e dinâmicas se amplia, particularmente em $C = 10$. Nesse regime, embora limites fixos agressivos apresentem menor consumo energético, como mostrado a seguir, eles não conseguem manter o SLA de latência, tornando-se inviáveis do ponto de vista de qualidade de serviço.

5.2. Cumprimento do SLA de TTFT p95

A Figura 3 apresenta o TTFT no percentil 95 para os diferentes regimes de concorrência, juntamente com o limiar do SLA (θ), definido a partir do valor mediano do TTFT p95 observado no *baseline* (700W) ao longo de múltiplas execuções independentes, acrescido de uma margem fixa de 5%. Essa abordagem reflete a capacidade operacional observada do sistema sob configuração nominal, prática comum em ambientes de produção.

Para $C = 6$, todos os limites de potência fixos e o controle dinâmico satisfazem o SLA. Esse comportamento indica que, em regimes de baixa concorrência, o sistema possui folga estrutural suficiente para absorver reduções agressivas de potência sem impacto significativo na latência de cauda.

Em $C = 8$, começa a emergir uma separação clara entre as políticas. Limites fixos mais agressivos (400W e abaixo) passam a operar próximos ao limiar do SLA, com alguns casos ultrapassando θ. Em contraste, o controle dinâmico mantém o TTFT p95 consistentemente abaixo do SLA, apresentando menor variabilidade e maior robustez frente ao aumento de carga.



(a) $C = 6$ **(b)** $C = 8$ **(c)** $C = 10$

Figura 3. TTFT no percentil 95 sob diferentes políticas de controle de potência máxima, para distintos regimes de concorrência. A linha tracejada indica o limiar do SLA (θ).

O regime mais crítico é observado em $C = 10$. Nesse cenário, o *baseline* é a única configuração que permanece abaixo do limiar do SLA (θ). Para limites de potência fixos abaixo de 700W, observa-se uma degradação progressiva do TTFT p95 à medida que o limite de potência é reduzido, com aumentos particularmente acentuados para limites mais agressivos (300W e 250W). O controle dinâmico, embora reduza consumo energético em relação ao *baseline*, não é suficiente para recuperar o TTFT p95 a ponto de satisfazer o SLA em $C = 10$, mantendo-se acima de θ.

5.3. Trade-off entre Eficiência Energética e SLA

A análise conjunta do consumo energético da Figura 2 e do TTFT p95 na Figura 3 evidencia um trade-off fundamental entre eficiência energética e cumprimento de SLA em serviços de inferência de LLMs. A simples redução do limite de potência da GPU permite diminuir o consumo energético total, mas introduz degradações progressivas na latência de cauda, que se tornam críticas à medida que a concorrência aumenta.

Nos regimes de menor concorrência ($C = 6$), tanto limites de potência fixos quanto o controle dinâmico conseguem reduzir o consumo energético sem comprometer o SLA, indicando a existência de folga estrutural no sistema. Em $C = 8$, esse trade-off torna-se mais evidente: limites de potência fixos mais agressivos passam a operar próximos ou acima do limiar do SLA, enquanto o controle dinâmico apresenta uma degradação de latência mais controlada, ainda que nem sempre suficiente para satisfazer θ.

No regime mais crítico ($C = 10$), observa-se que apenas o *baseline* satisfaz o SLA definido. Todas as demais políticas, incluindo o controle dinâmico, violam o limiar de TTFT p95. No entanto, há diferenças qualitativas importantes entre as abordagens. Limites de potência fixos agressivos resultam em aumentos abruptos do TTFT p95, com degradações severas de latência à medida que o *power cap* é reduzido. Em contraste, o controle dinâmico mantém o TTFT p95 sistematicamente mais próximo do limiar do SLA, com menor variabilidade e sem os colapsos observados em limites fixos equivalentes ou mais econômicos em termos de energia.

Esses resultados indicam que, embora o controle dinâmico não seja suficiente

para garantir o cumprimento do SLA em regimes de concorrência extrema sob o limiar definido, ele oferece um compromisso mais equilibrado entre consumo energético e degradação de desempenho quando comparado a limites de potência fixos. Em particular, o controlador permite reduzir substancialmente o consumo energético em relação ao *baseline*, ao mesmo tempo em que evita as degradações abruptas de latência associadas a *power caps* fixos agressivos, caracterizando-se como uma abordagem mais robusta para operar sob restrições energéticas em cenários realistas de carga variável.

Em síntese, em resposta direta à **RQ** descrita na Seção 4, os resultados mostram que *power caps* fixos podem reduzir energia de forma agressiva, porém o fazem à custa de degradações pronunciadas de latência de cauda à medida que a concorrência cresce, tornando-se inviáveis sob SLAs mais rígidos. O controlador dinâmico, por sua vez, oferece o melhor compromisso prático entre eficiência e qualidade: ele reduz consistentemente o consumo em relação ao *baseline* e evita colapsos abruptos de TTFT p95 típicos de limites fixos agressivos, ajustando-se às variações de carga. Mesmo quando o SLA não é atendido no regime mais crítico ($C = 10$), o controlador atua como um mecanismo de mitigação, mantendo a degradação de latência mais controlada e próxima ao limiar, ao mesmo tempo em que preserva gastos energéticos em relação ao *baseline*. Esses achados reforçam a utilidade de controle dinâmico de potência como uma alternativa mais robusta do que políticas estáticas em cenários realistas de carga variável.

6. Conclusão

Este trabalho investigou o uso de controle de potência em nível de GPU como mecanismo para melhorar a eficiência energética de serviços de inferência de LLMs sob restrições de latência. Foi proposto um algoritmo adaptativo de controle de *power cap*, orientado por SLA e baseado em telemetria de execução, que ajusta dinamicamente o limite máximo de potência das GPUs explorando a relação não linear entre consumo energético e desempenho durante a inferência. A proposta foi avaliada em um ambiente real, com múltiplas instâncias de inferência balanceadas por uma política fixa de *Shortest Expected Delay*, permitindo comparar de forma controlada limites de potência fixos e controle dinâmico.

Os resultados experimentais demonstram que a simples redução estática do limite de potência é eficaz para reduzir o consumo energético, porém introduz degradações progressivas na latência de cauda à medida que a concorrência aumenta. Em regimes de baixa concorrência, limites fixos agressivos conseguem reduzir significativamente o consumo energético sem violar o SLA. No entanto, à medida que a carga se intensifica, esses limites passam a produzir aumentos abruptos do TTFT p95, tornando-se inviáveis do ponto de vista de qualidade de serviço. Em regimes de alta concorrência, apenas o *baseline* com potência máxima é capaz de satisfazer o SLA definido.

O controle dinâmico proposto por este trabalho, por sua vez, apresenta um comportamento intermediário. Embora não seja capaz de garantir o cumprimento do SLA em regimes de concorrência extrema sob o limiar adotado, o controlador reduz consistentemente o consumo energético em relação ao *baseline* e evita as degradações severas de latência observadas em limites de potência fixos agressivos. Esse resultado evidencia que o ajuste dinâmico do *power cap* constitui uma alternativa mais robusta do que políticas estáticas quando o sistema opera sob cargas variáveis, oferecendo um compromisso mais equilibrado entre eficiência energética e degradação de desempenho.

Como trabalho futuro, pretende-se explorar algoritmos mais sofisticados para o controle dinâmico de potência, incluindo abordagens preditivas e adaptativas que incorporem informações adicionais sobre o estado do sistema e do *workload*. Os resultados apresentados neste artigo demonstram a viabilidade prática do controle dinâmico de *power cap* como um mecanismo efetivo para redução do consumo energético em serviços de inferência de LLMs, abrindo caminho para investigações nesse eixo de otimização.

Referências

Gogineni, K., Suvizi, A., and Venkataramani, G. (2025). Llms on a budget: System-level approaches to power-efficient and scalable fine-tuning. *IEEE Open Journal of the Computer Society*, 6:987–1000.

Jain, K., Parayil, A., Mallick, A., Choukse, E., Qin, X., Zhang, J., Goiri, I. n., Wang, R., Bansal, C., Rühle, V., Kulkarni, A., Kofsky, S., and Rajmohan, S. (2025). Performance aware llm load balancer for mixed workloads. In *Proceedings of the 5th Workshop on Machine Learning and Systems*, EuroMLSys '25, page 19–30, New York, NY, USA. Association for Computing Machinery.

Kakolyris, A. K., Masouros, D., Xydis, S., and Soudris, D. (2024). Slo-aware gpu dvfs for energy-efficient llm inference serving. *IEEE Computer Architecture Letters*, 23(2):150–153.

Liu, J., Chung, J.-W., Wu, Z., Lai, F., Lee, M., and Chowdhury, M. (2024). Andes: Defining and enhancing quality-of-experience in llm-based text streaming services.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Jin, H., Chen, T., and Jia, Z. (2025). Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Comput. Surv.*, 58(1).

Niu, C., Zhang, W., Zhao, Y., and Chen, Y. (2025). Energy efficient or exhaustive? benchmarking power consumption of llm inference engines. *SIGENERGY Energy Inform. Rev.*, 5(2):56–62.

Rostam, Z. R. K., Szénási, S., and Kertész, G. (2024). Achieving peak performance for large language models: A systematic review. *IEEE Access*, 12:96017–96050.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. (2023). From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9.

Stojkovic, J., Zhang, C., Goiri, I., Torrellas, J., and Choukse, E. (2025a). Dynamollm: Designing llm inference clusters for performance and energy efficiency. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1348–1362.

Stojkovic, J., Zhang, C., Goiri, I. n., Choukse, E., Qiu, H., Fonseca, R., Torrellas, J., and Bianchini, R. (2025b). *TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms*, page 1266–1281. Association for Computing Machinery, New York, NY, USA.