Investigando a Confiabilidade e Efetividade de LLMs em Otimização de Desempenho: Um Estudo Exploratório com Dois Servidores Web

Pedro Jardelino Neto[1], **Nabor C. Mendonça**[1]

[1]Programa de pós-graduação em informática aplicada
Universidade de Fortaleza (UNIFOR) – Fortaleza, CE – Brasil

jardelino@tre-ce.jus.br, nabor@unifor.br

Abstract. *This paper evaluates the operational reliability and optimization effectiveness of four state-of-the-art LLMs acting as web server optimizers in a controlled experimental environment. Across* 48 *trials,* ***66.7%*** *of the responses preserved system operability, but only* ***43.8%*** *resulted in effective optimizations, while* ***33.3%*** *caused damage to the target system. Among the models,* ***Gemini 3 Pro*** *achieved the highest overall performance (****75.0%****), whereas* ***DeepSeek-V3.2*** *showed the lowest (****16.7%****). The strongest effect, however, was associated with the target server:* ***Apache*** *achieved a* ***66.7%*** *success rate, compared to* ***20.8%*** *for* ***Nginx*** *(*$p = 0.003$*). Overall, the results indicate that, even under favorable conditions, the use of LLMs for web server optimization still combines limited effectiveness with non-negligible operational risk.*

Resumo. *Este artigo avalia a confiabilidade operacional e a efetividade de otimização de quatro LLMs de ponta atuando como otimizadores de servidores web em ambiente experimental controlado. Em* 48 *ensaios,* ***66,7%*** *das respostas preservaram a operacionalidade do sistema, mas apenas* ***43,8%*** *produziram otimizações efetivas, enquanto* ***33,3%*** *danificaram o servidor-alvo. Entre os modelos, o* ***Gemini 3 Pro*** *obteve o melhor desempenho global (****75,0%****) e o* ***DeepSeek-V3.2****, o pior (****16,7%****). O efeito mais forte, contudo, foi o do servidor-alvo: o* ***Apache*** *atingiu* ***66,7%*** *de sucesso, contra* ***20,8%*** *no* ***Nginx*** *(*$p = 0{,}003$*). Em conjunto, os resultados indicam que, mesmo em condições favoráveis, o uso de LLMs na otimização de servidores web ainda combina efetividade limitada com risco operacional relevante.*

1. Introdução

A automação de tarefas de gerenciamento de infraestrutura por meio de grandes modelos de linguagem (*Large Language Models* — LLMs) tem despertado interesse crescente [Yang et al. 2025, Vitui and Chen 2025]. Entretanto, a confiabilidade e a efetividade desses modelos em cenários não triviais ainda são questões em aberto, especialmente devido à natureza estocástica inerente aos LLMs [Yi et al. 2025, Yang et al. 2023, He et al. 2025].

Neste contexto, a otimização de desempenho de servidores web sob alta carga é um desafio prático e relevante. Sua importância remonta ao clássico problema C10K, focado na busca por servidores capazes de atender eficientemente dez mil conexões simultâneas[Kegel 2003], e estende-se a ambientes modernos e críticos que demandam otimização contínua [Spieker et al. 2025, HighScalability 2013].

Este artigo investiga de forma exploratória a efetividade e confiabilidade de quatro LLMs de última geração, sendo dois de pesos fechados (ChatGPT 5.2 Thinking e Gemini 3 Pro) e dois de pesos abertos (DeepSeek-V3.2 e Qwen3-Max), na otimização de desempenho de dois servidores web amplamente utilizados (Apache e Nginx). Os experimentos foram conduzidos em infraestrutura própria, *in loco* e exclusiva, com o objetivo de minimizar variáveis exógenas.

A metodologia iterativa adotada é inspirada no método OPRO (*Optimization by PROmpting*) [Yang et al. 2023]. Em cada iteração, o modelo recebe um *meta-prompt* contendo o contexto do sistema-base e a trajetória de otimização, gerando uma solução técnica na forma de um *shell script*. Esta solução é aplicada ao sistema-base para originar o sistema-alvo, o qual é submetido a testes de carga rigorosos. A avaliação quantifica a capacidade de atendimento em número de usuários virtuais (*virtual users* — VUs) por meio do valor máximo de carga do sistema-alvo. Para garantir robustez estatística, o valor é certificado por uma bateria de 59 testes independentes e consecutivos para cada solução gerada pelos LLMs.

O estudo foi concebido para responder as seguintes questões de pesquisa:

QP1: Em que medida LLMs produzem soluções funcionais (confiabilidade operacional) ao atuar como otimizadores de servidores web?
QP2: Qual a efetividade de otimização dos LLMs ao atuar como otimizadores de servidores web?
QP3: Entre modelo e servidor, qual fator exerce maior influência sobre a probabilidade de sucesso da otimização gerada?

Os resultados mostram que, mesmo em um cenário experimental favorável, com quatro LLMs de ponta atuando em um ambiente simples, controlado e de baixa complexidade operacional, o desempenho dos modelos permaneceu limitado. No agregado, **66,7%** das respostas preservaram a operacionalidade do sistema, mas apenas **43,8%** produziram otimizações efetivas, ao passo que **33,3%** danificaram o servidor-alvo. Em outras palavras, embora a maioria das saídas tenha sido funcional, menos da metade resultou em cumprir a tarefa solicitada, o que ainda impede caracterizar o uso autônomo desses modelos como confiável.

Quanto à efetividade, houve heterogeneidade entre modelos. O **Gemini 3 Pro** apresentou o melhor desempenho global (**75,0%**), enquanto o **DeepSeek-V3.2** obteve o pior (**16,7%**); **ChatGPT 5.2 Thinking** e **Qwen3-Max** ocuparam posição intermediária, ambos com **41,7%**. Essa variação foi detectável no agregado, mas o fator mais robustamente associado ao sucesso foi o servidor-alvo: o **Apache** atingiu **66,7%** de sucesso, contra **20,8%** no **Nginx**, diferença estatisticamente significativa. Em termos substantivos, uma solução teve probabilidade pouco mais de três vezes maior de ser bem-sucedida quando direcionada ao Apache.

A análise estratificada reforça essa conclusão. Mesmo após o controle por modelo, o efeito do servidor permaneceu estatisticamente significativo e consistente entre os quatro modelos, ao passo que a heterogeneidade entre LLMs deixou de ser detectável quando a comparação foi feita separadamente por servidor. Assim, no contexto deste estudo, as características do sistema-alvo influenciaram mais, e de forma mais consistente, a probabilidade de sucesso do que o modelo utilizado.

Em síntese, as principais contribuições deste trabalho são:

1. uma avaliação experimental controlada do uso de LLMs como otimizadores de servidores web em infraestrutura *in loco*;
2. a quantificação da confiabilidade operacional e da efetividade de otimização de quatro LLMs modernas em dois servidores, com estimativas e testes rigorosos adequados ao tamanho amostral;
3. a produção de evidências empíricas exploratórias de que, nas condições avaliadas, o servidor-alvo exerceu influência mais forte e mais consistente sobre a probabilidade de sucesso do que o modelo empregado.

A estrutura do restante do artigo está organizada da seguinte forma: a Seção 2 compara o presente estudo com diversos trabalhos relacionados; a Seção 3 detalha a metodologia adotada; a Seção 4 apresenta os resultados e responde às três questões de pesquisa; a Seção 5 discute as implicações deste trabalho; a Seção 6 aborda as limitações e ameaças à validade do estudo; finalmente, a Seção 7 conclui o trabalho e aponta direções para pesquisas futuras.

2. Trabalhos Relacionados

A literatura recente sobre LLMs em operações de infraestrutura pode ser organizada em três eixos: **(i)** LLMs como otimizadores de prompts; **(ii)** LLMs como otimizadores de desempenho via modificação de código; e **(iii)** LLMs como apoio à configuração e ao gerenciamento de sistemas. O presente estudo se insere principalmente no terceiro eixo, porém com ênfase explícita em confiabilidade operacional e efetividade de otimização sob um protocolo experimental controlado.

LLMs como otimizadores de prompts. Yang et al. propõem o método OPRO (*Optimization by PROmpting*) [Yang et al. 2023], que formula a melhoria de prompts como um processo iterativo de geração e seleção de candidatos. Embora o OPRO forneça a inspiração metodológica (iterações guiadas por histórico), o seu foco é otimizar **instruções** visando acurácia em tarefas-alvo (por exemplo, problemas de otimização e tarefas de linguagem), e não otimizar configurações de sistemas reais sob restrições operacionais e risco de degradação. Assim, o presente estudo complementa essa linha ao investigar um cenário em que a "solução" produzida pelo LLM é um *shell script* que altera o comportamento de um servidor web, exigindo não apenas qualidade da recomendação, mas também execução correta, reversibilidade e ausência de impacto negativo no serviço.

LLMs para melhoria de desempenho via alterações em código. Yi et al. [Yi et al. 2025] avaliam LLMs na produção de *patches* de desempenho em programas, comparando com soluções humanas. O estudo evidencia ganhos pontuais, porém também limitações para atingir soluções próximas do ótimo. He et al. [He et al. 2025] introduzem o SWE-Perf como *benchmark* para otimização de desempenho em repositórios reais, em que o modelo precisa compreender o contexto do projeto e gerar alterações validadas por testes. Embora esses trabalhos sejam relevantes por discutirem limitações dos LLMs em otimização de desempenho de software, eles diferem do presente estudo em aspectos críticos: (i) atuam no artefato código-fonte (tipicamente com validação por testes), enquanto aqui a intervenção ocorre em configuração e parâmetros de execução de sistemas; (ii) não enfatizam o efeito de falhas operacionais (por exemplo, tornar o serviço indisponível) como variável central; e (iii) usualmente não investigam, de forma explícita, a

variabilidade de soluções sob o mesmo contexto em termos de confiabilidade operacional no ambiente-alvo.

LLMs para configuração e gerenciamento orientados a desempenho. Spieker et al. [Spieker et al. 2025] discutem o uso de prompts para apoiar recomendações de configuração com foco em desempenho, apontando oportunidades e limitações. Essa direção é a mais próxima deste trabalho, pois também trata a otimização como seleção de diretivas e parâmetros sob restrições. A diferença central é metodológica e de escopo experimental: o presente estudo avalia, de forma controlada, a execução de soluções produzidas por LLMs diretamente sobre dois servidores web amplamente utilizados (Apache e Nginx), com métricas operacionais e um protocolo de avaliação que permite quantificar tanto ganho quanto risco (soluções disfuncionais/degradantes). Com isso, busca-se evidenciar não apenas se um modelo pode recomendar configurações, mas com que probabilidade produz soluções utilizáveis e com que consistência produz melhorias.

Em síntese, o presente estudo contribui ao preencher uma lacuna entre trabalhos que tratam LLMs como otimizadores (de prompts ou de código) e o cenário de operação em infraestrutura, onde a otimização é condicionada por limiares de desempenho e pela necessidade de confiabilidade no processo de mudança.

3. Metodologia

A metodologia deste trabalho fundamenta-se na adaptação do desafio clássico C10K [Kegel 2003] para o contexto de gerenciamento autônomo de sistemas via modelos de linguagem. O problema consiste em otimizar as configurações de dois servidores web amplamente utilizados [W3Techs 2025], Apache e Nginx, visando maximizar o número de conexões simultâneas sob limiares de desempenho estritos. A tarefa foi selecionada por exigir competências técnicas não triviais, sendo, contudo, passível de representação e resolução por agentes baseados em linguagem natural.

3.1. Definições Operacionais

Para fins de padronização e análise, adotam-se as seguintes definições:

- **Meta-prompt:** Instrução de entrada fornecida ao LLM que utiliza um ciclo iterativo de meta-prompting que combina instruções de tarefa, o histórico de sucessos e falhas do próprio modelo para guiar o processo de otimização.
- **Ensaio:** Ciclo individual composto pela geração de uma solução pelo LLM, sua aplicação ao sistema e a subsequente avaliação de desempenho.
- **Rodada:** Conjunto de três ensaios independentes realizados para mitigar o impacto da estocasticidade dos modelos.
- **Sistema-base e Sistema-alvo:** O primeiro refere-se ao servidor com configurações padrão; o segundo, ao sistema após a aplicação do *shell script* gerado pelo LLM.
- **Soluções (Inexequível, Disfuncional e Funcional):** Uma solução é **inexequível** se violar o formato ou restrições do prompt; **disfuncional** se prejudica o funcionamento do serviço; e **funcional** se mantiver a operabilidade, independentemente de promover ganho de desempenho.
- **Confiabilidade Operacional:** Taxa de ensaios em que o LLM produz uma solução funcional.

- **Efetividade de Otimização:** Taxa de ensaios em que a solução melhora o desempenho do sistema-base conforme os critérios de sucesso estabelecidos.

3.2. Infraestrutura e Configuração dos Sistemas

Os experimentos foram conduzidos em infraestrutura exclusiva *in loco*, visando mitigar a influência de variáveis exógenas de ambientes compartilhados. O sistema-base compreende os servidores Apache 2.4.58 e Nginx 1.24.0 executando sobre GNU/Linux (Ubuntu Server 24.04 LTS). Utilizaram-se as configurações padrão da distribuição, certificados TLS autoassinados e uma carga de trabalho composta por uma página HTML estática de 21 KiB. A comunicação entre o gerador de carga e o servidor ocorreu em rede IPv4 local, via barramento exclusivo com largura de banda de 1 Gbps.

3.3. Protocolo de Otimização e Iteração

O fluxo de interação com os LLMs é inspirado no método OPRO (*Optimization by PROmpting*) [Yang et al. 2023], adaptado para a geração de artefatos de configuração de sistemas. O processo é estruturado em um ciclo iterativo guiado por um *meta-prompt*, em que a saída esperada do modelo é um *shell script* executável, responsável por aplicar alterações de parâmetros no servidor web.

Considerando a estocasticidade inerente aos modelos de IA generativa [Yi et al. 2025], o protocolo foi organizado em duas rodadas sequenciais, cada uma composta por três ensaios independentes, totalizando **três amostras por servidor em um dado modelo para cada prompt**. Essa abordagem segue práticas consolidadas em *benchmarks* e estudos recentes [Patwardhan et al. 2025, Yi et al. 2025]. Cada ensaio segue três etapas: (i) Geração da solução candidata; (ii) Aplicação da solução gerada ao sistema-base para originar o sistema-alvo; e (iii) Avaliação do sistema-alvo via testes de carga. O *meta-prompt* inicial contém o contexto do sistema e as restrições operacionais; para a segunda rodada, este é atualizado com a trajetória de otimização, que consolida o histórico de soluções anteriores e seus resultados empíricos. Os meta-prompts utilizados em cada rodada, as respostas brutas dos modelos, os scripts gerados, os registros dos testes k6 e os resultados consolidados estão organizados no repositório público do estudo, disponível em `https://gitlab.com/jardelino/mia-2026-s-1`.

3.4. Protocolo de Teste e Critérios de Sucesso

A geração de carga utilizou a ferramenta k6 [Grafana Labs 2025], estabelecendo conexões HTTPS persistentes (*keep-alive*). Cada teste manteve um número fixo de VUs por três minutos: o primeiro minuto destinado ao aquecimento do siteama e o segundo à coleta efetiva de amostras. Os limiares de aceitação seguem requisitos típicos de produção [Beyer et al. 2016]: taxa de falha $< 1\%$; conexão TCP p(95) < 100 ms; tempo total de requisição p(95) < 2 s; e *handshake* TLS p(95) < 300 ms.

Em um **teste binomial unilateral** com $\alpha = 0{,}05$, 59 observações concordantes permitem rejeitar a hipótese $H_0 : p \leq 0{,}95$ e, assim, sustentar que a probabilidade de o sistema apresentar o mesmo comportamento em uma execução qualquer é superior a 95%. Assim, capacidade do sistema é definida a partir de dois valores de referência, cada um validado por **59 testes independentes e consecutivos**.

Valor de sobrecarga: menor quantitativo de VUs para o qual o sistema passa a **não atender** aos requisitos. Para os sistemas-base Apache e Nginx, esses valores são, respectivamente, 500 e 23.800 VUs.

Valor máximo de carga: maior quantitativo de VUs suportado de forma estável, isto é, ainda **atendendo** aos requisitos. Para os sistemas-base Apache e Nginx, esses valores são, respectivamente, 200 e 22.600 VUs.

Ao final dos testes, cuja duração mínima é de aproximadamente **10 horas por solução**, o sistema-alvo é classificado como: (i) *Otimizado*, se o valor máximo de carga for igual ou superior ao valor de sobrecarga do sistema-base; (ii) *Não Otimizado*, se mantiver a estabilidade sem superar o valor de sobrecarga do sistema-base; (iii) *Degradado*, se não suportar o valor máximo de carga do sistema-base; ou (iv) *Danificado*, caso o sistema deixe de prestar o serviço.[1]

3.5. Delineamento Experimental e Questões de Pesquisa

O protocolo experimental descrito nas subseções anteriores foi estruturado para fornecer evidências empíricas que permitam responder às três questões de pesquisa deste estudo, introduzidas na Seção 1:

- **QP1 (Confiabilidade):** Avaliada por meio da métrica de confiabilidade operacional, mensurando a taxa de soluções funcionais entregues pelos modelos.
- **QP2 (Efetividade por Servidor):** Respondida através da análise comparativa da efetividade de otimização em cenários distintos (Apache vs. Nginx).
- **QP3 (Fatores de Influência):** Analisada por meio do impacto relativo que a troca de modelo ou de servidor exerce sobre a probabilidade de sucesso da otimização.

3.6. Modelos Selecionados

A seleção dos modelos de linguagem baseou-se em critérios de representatividade tecnológica, liderança em *benchmarks* de codificação e diversidade de paradigmas de desenvolvimento (pesos abertos vs. pesos fechados). Esses critérios são especialmente relevantes neste estudo porque a tarefa exige interpretar restrições operacionais, raciocinar sobre parâmetros de configuração e gerar scripts *shell* sintaticamente válidos e executáveis. Foram selecionados quatro modelos de última geração, representativos do estado-da-arte entre o final de 2025 e o início de 2026: ChatGPT 5.2 Thinking e Gemini 3 Pro, representando soluções de pesos fechados com arquiteturas otimizadas para raciocínio complexo; e DeepSeek-V3.2 e Qwen3-Max, como expoentes de modelos de pesos abertos que apresentam desempenho competitivo em benchmarks de LLMs [White et al. 2025].

Para refletir o cenário de uso mais comum e mitigar vieses decorrentes de calibração manual de parâmetros, as interações com os quatro modelos ocorreram via suas respectivas interfaces web oficiais, utilizando as configurações padrão de temperatura e amostragem fornecidas pelos provedores. O controle explícito de temperatura não foi adotado porque o objetivo do estudo não era comparar regimes de amostragem, mas avaliar o comportamento dos modelos em seu modo padrão de uso. Além disso, as interfaces oficiais não oferecem, de forma homogênea, o mesmo nível de parametrização entre provedores, o que dificultaria uma comparação justa.

[1]Neste estudo, o termo "Otimizado" é empregado em sentido operacional, indicando que o sistema-alvo passou a sustentar pelo menos uma carga que já caracteriza sobrecarga no sistema-base. Trata-se, portanto, de um critério binário de ultrapassagem de patamar, e não de uma medida contínua da magnitude do ganho.

4. Resultados

Os resultados a seguir devem ser interpretados à luz de um cenário deliberadamente favorável ao bom desempenho dos modelos. Os experimentos foram conduzidos com **quatro LLMs de ponta** e em um **ambiente experimental simples, controlado e de baixa complexidade operacional**, concebido para reduzir variáveis exógenas e facilitar a tarefa de otimização. Por isso, os achados apresentados aqui devem ser entendidos como uma avaliação em condições particularmente favoráveis, e não como um retrato conservador do comportamento dos LLMs em ambientes operacionais mais complexos.

4.1. QP1: Em que medida LLMs produzem soluções funcionais (confiabilidade operacional) ao atuar como otimizadores de servidores web?

A Figura 1 apresenta os resultados dos testes dos sistemas-alvo em três desfechos mutuamente exclusivos, uma vez que não houve ocorrência de sistemas-alvo degradados no estudo. Dado o tamanho amostral reduzido (48 ensaios, 24 por rodada) e a presença de registros esparsos, a análise inferencial foi baseada em métodos exatos: intervalos de confiança binomiais de Clopper–Pearson, teste exato de Fisher–Freeman–Halton para comparar a distribuição dos desfechos entre rodadas e teste exato de Fisher para contrastes binários complementares.

No agregado, as LLMs produziram **soluções funcionais** em **66,7%** dos ensaios ($32/48$; IC95%: $51,6\%$ – $79,6\%$). Contudo, apenas **43,8%** corresponderam a otimizações efetivas ($21/48$; IC95%: $29,5\%$ – $58,8\%$), enquanto **22,9%** mantiveram o sistema operacional sem melhorá-lo ($11/48$; IC95%: $12,0\%$ – $37,3\%$). Mais preocupante, **33,3%** das soluções **danificaram** o sistema-alvo ($16/48$; IC95%: $20,4\%$ – $48,4\%$). Em termos práticos, isso indica que, embora a operacionalidade seja preservada na maioria dos casos, a taxa de saídas disruptivas ainda é alta demais para caracterizar uso autônomo confiável.

A comparação entre rodadas sugere melhora, mas sem evidência estatística conclusiva. Na Rodada 1, a proporção de saídas funcionalmente seguras foi de **58,3%** ($14/24$; IC95%: $36,6\%$ – $77,9\%$), com **33,3%** de otimizações efetivas ($8/24$; IC95%: $15,6\%$ – $55,3\%$) e **41,7%** de soluções disfuncionais ($10/24$; IC95%: $22,1\%$ – $63,4\%$). Na Rodada 2, esses valores passaram para **75,0%** ($18/24$; IC95%: $53,3\%$ – $90,2\%$), **54,2%** ($13/24$; IC95%: $32,8\%$ – $74,4\%$) e **25,0%** ($6/24$; IC95%: $9,8\%$ – $46,7\%$), respectivamente. Ainda assim, a mudança na distribuição tricotômica entre rodadas não foi detectável pelo teste exato de Fisher–Freeman–Halton ($p = 0,329$); o mesmo ocorreu ao se comparar apenas a taxa de otimização ($p = 0,244$) ou a taxa de dano ($p = 0,359$).

Mesmo sem significância formal, a direção do efeito é coerente com a hipótese de que iterações adicionais e maior estruturação do contexto por meio de *meta-prompting* reduzem erros de formulação. Persistem, porém, problemas como duplicação de diretivas, referências inválidas e ausência de verificações de consistência. Em síntese, **as LLMs já geram soluções funcionalmente válidas em parte majoritária dos casos, mas sua confiabilidade operacional ainda é insuficiente**: cerca de dois terços das respostas preservaram o sistema, menos da metade o otimizou, e um terço causou dano operacional mesmo em condições laboratoriais favoráveis.

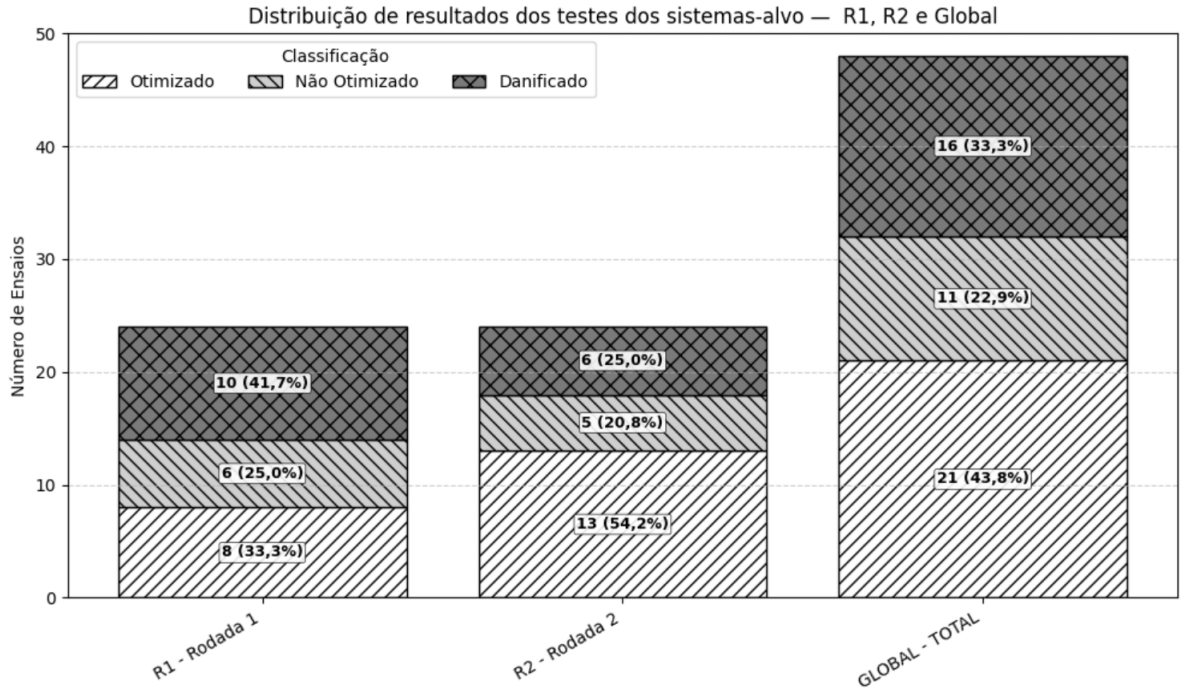

Figura 1. Distribuição de resultados dos testes dos sistemas-alvo para Rodada 1 (R1), Rodada 2 (R2) e Global

4.2. QP2: Qual a efetividade de otimização dos LLMs ao atuar como otimizadores de servidores web?

A Tabela 1 detalha a efetividade de otimização por modelo, servidor e rodada. Cada solução foi tratada como observação binária (*sucesso* ou *fracasso*), e a inferência foi conduzida com métodos exatos devido à baixa frequência por célula ($n = 3$ por combinação modelo$\times$servidor$\times$rodada) e à presença de casos extremos, como $6/6$ para o Gemini 3 Pro no Apache.

No conjunto das 48 soluções, a efetividade global foi de **43,8%** ($21/48$; IC95%: $29,5\% - 58,8\%$). Mesmo em um cenário simplificado e controlado, portanto, menos da metade das respostas produziu otimizações válidas. Essa efetividade foi heterogênea entre modelos: o **Gemini 3 Pro** apresentou o melhor desempenho global, com **75,0%** ($9/12$; IC95%: $42,8\% - 94,5\%$); o **DeepSeek-V3.2**, o pior, com **16,7%** ($2/12$; IC95%: $2,1\% - 48,4\%$); e **ChatGPT 5.2 Thinking** e **Qwen3-Max** ocuparam posição intermediária, ambos com **41,7%** ($5/12$; IC95%: $15,2\% - 72,3\%$). A heterogeneidade global entre LLMs foi detectável pelo teste exato de Fisher–Freeman–Halton ($p = 0,040$).

O efeito mais forte, porém, foi o do servidor-alvo. Agregando-se os quatro modelos, o **Apache** apresentou **66,7%** de sucesso ($16/24$; IC95%: $44,7\% - 84,4\%$), contra apenas **20,8%** no **Nginx** ($5/24$; IC95%: $7,1\% - 42,2\%$). A diferença de **45,8 pontos percentuais** foi estatisticamente significativa ($p = 0,003$), com razão de riscos de **3,20** (IC95%: $1,40 - 7,34$).

Assim, uma solução tem cerca de 3,2 vezes mais chance de ser bem-sucedida no Apache do que no Nginx. Esse resultado sugere que o Nginx constitui um alvo menos favorável à atuação dos modelos, possivelmente por combinar uma configuração padrão

Tabela 1. Efetividade de otimização

Modelo	Servidor	Rodada 1	Rodada 2	Total Final
ChatGPT 5.2 Thinking	Apache	33,3% (1/3)	100% (3/3)	66,7% (4/6)
	Nginx	33,3% (1/3)	0% (0/3)	16,7% (1/6)
	Global	**33,3% (2/6)**	**50,0% (3/6)**	**41,7% (5/12)**
DeepSeek-V3.2	Apache	0% (0/3)	66,7% (2/3)	33,3% (2/6)
	Nginx	0% (0/3)	0% (0/3)	0% (0/6)
	Global	**0% (0/6)**	**33,3% (2/6)**	**16,7% (2/12)**
Gemini 3 Pro	Apache	100% (3/3)	100% (3/3)	100% (6/6)
	Nginx	33,3% (1/3)	66,7% (2/3)	50,0% (3/6)
	Global	**66,7% (4/6)**	**83,3% (5/6)**	**75,0% (9/12)**
Qwen3-Max	Apache	33,3% (1/3)	100% (3/3)	66,7% (4/6)
	Nginx	33,3% (1/3)	0% (0/3)	16,7% (1/6)
	Global	**33,3% (2/6)**	**50,0% (3/6)**	**41,7% (5/12)**
Global Geral	–	**33,3% (8/24)**	**54,2% (13/24)**	**43,8% (21/48)**

já otimizada para alta concorrência com menor disponibilidade de padrões de otimização triviais aprendidos durante o treinamento, o que reduz a margem para ganhos facilmente exploráveis pelas LLMs.

4.3. QP3 – Entre modelo e servidor, qual fator exerce maior influência sobre a probabilidade de sucesso da otimização gerada?

Mantendo o mesmo critério inferencial da QP2, esta questão foi examinada com métodos exatos e análise estratificada. No agregado, a diferença entre servidores foi pronunciada e estatisticamente robusta: **66,7%** de sucesso no Apache ($16/24$; IC95%: $44,7\% – 84,4\%$) contra **20,8%** no Nginx ($5/24$; IC95%: $7,1\% – 42,2\%$), com $p = 0,003$ e razão de riscos bruta de **3,20** (IC95%: $1,40 – 7,34$). Esse efeito permaneceu após controle por modelo: na análise de Cochran–Mantel–Haenszel, a associação entre *servidor* e *sucesso* manteve-se significativa, com *odds ratio* comum de **17,5** (IC95%: $2,73 – 112,20$; $p < 0,001$). Além disso, o teste de Breslow–Day não indicou heterogeneidade relevante entre estratos ($p = 0,842$), sugerindo que a penalização do Nginx foi consistente entre os quatro LLMs.

O fator **modelo** também influenciou o sucesso, mas de forma menos robusta. No consolidado por LLM, as taxas variaram de **16,7%** no DeepSeek-V3.2 a **75,0%** no Gemini 3 Pro, com heterogeneidade global detectável ($p = 0,040$). Contudo, essa diferença desaparece quando a comparação é feita separadamente por servidor: no **Apache**, $p = 0,145$; no **Nginx**, $p = 0,317$.

Portanto, no contexto deste experimento, a evidência empírica favorece a interpretação de que **o servidor web a ser otimizado influenciou mais, e de forma mais consistente, a probabilidade de sucesso do que o modelo utilizado**. Isso não implica que o modelo seja irrelevante: o Gemini 3 Pro, por exemplo, manteve vantagem descritiva sobre os demais. Entretanto, o efeito do servidor foi mais estável entre estratos e mais facilmente detectável estatisticamente. Mesmo em um cenário experimental bastante favorável, portanto, as características do sistema-alvo emergiram como determinante central do sucesso da otimização baseada em LLMs.

5. Discussão

Os resultados indicam que, mesmo sob condições particularmente favoráveis, com **quatro LLMs de ponta** atuando em um **ambiente simples, controlado e de baixa complexidade operacional**, o uso desses modelos como otimizadores de servidores web permanece promissor, mas limitado. Em princípio, esse era um contexto no qual se esperaria desempenho elevado; por isso, a performance observada reforça a relevância dos achados.

Do ponto de vista da confiabilidade operacional, o quadro geral é frustrante. Embora **66,7%** das respostas tenham preservado o funcionamento do serviço, apenas **43,8%** produziram otimizações efetivas, ao passo que **33,3%** danificaram o sistema. Assim, a capacidade de gerar saídas funcionalmente válidas já é majoritária, mas ainda insuficiente para sustentar uso autônomo confiável. A melhora observada da Rodada 1 para a Rodada 2, com aumento da taxa de otimização e redução da taxa de dano, aponta na direção de benefício do refinamento iterativo, mas a ausência de significância estatística recomenda cautela: os dados sugerem avanço, sem ainda demonstrar mudança conclusiva.

Quanto à efetividade, o desempenho agregado foi apenas moderado, com menos da metade das 48 soluções resultando em sucesso. Houve heterogeneidade entre modelos, com destaque para o **Gemini 3 Pro**, mas o padrão mais consistente foi a forte dependência do servidor-alvo. No consolidado, o **Apache** apresentou taxa de sucesso muito superior à do **Nginx**, diferença que se mostrou estatisticamente significativa. Em termos substantivos, isso sugere que a utilidade prática dos LLMs, nesse tipo de tarefa, depende menos de uma capacidade genérica de "otimizar" e mais do acoplamento entre o modelo e o sistema específico sobre o qual ele opera.

Essa interpretação é reforçada pela análise comparativa entre os fatores *modelo* e *servidor*. Embora o tipo de LLM tenha influenciado o sucesso, o efeito do servidor foi mais forte e mais estável: a penalização associada ao Nginx permaneceu mesmo após o controle por modelo e não apresentou heterogeneidade relevante entre estratos. Em outras palavras, a variação entre modelos existe, mas foi a natureza do sistema-alvo que melhor explicou a probabilidade de êxito. No contexto deste estudo, portanto, as características do servidor a ser otimizado emergem como determinante central do sucesso.

Em conjunto, esses achados sustentam uma leitura equilibrada. Os LLMs já demonstram capacidade real de gerar configurações válidas e, em parte dos casos, efetivamente superiores ao sistema-base. No entanto, essa capacidade ainda é inconsistente, sensível ao contexto e acompanhada de risco operacional não desprezível. Em um cenário simples e favorável, seria razoável esperar desempenho robusto; o fato de isso não ter ocorrido de forma uniforme sugere que, em ambientes mais complexos, a dependência de validação externa, supervisão humana e mecanismos adicionais de verificação tende a ser ainda maior. Para a comunidade de análise de desempenho, o estudo também sugere que avaliações de LLMs como otimizadores devem considerar não apenas ganho obtido, mas também risco operacional, sensibilidade ao sistema-alvo e variabilidade entre execuções.

6. Limitações e Ameaças à Validade

Validade externa e generalização. Os resultados restringem-se às condições de contorno estabelecidas: carga sintética baseada em página estática (21 KiB) e rede de 1 Gbps. A ausência de tráfego dinâmico ou roteamento complexo implica cautela na generalização

para cenários heterogêneos. Além disso, as conclusões limitam-se às versões específicas do Apache (2.4.58) e Nginx (1.24.0) sob Ubuntu Server 24.04 LTS.

Validade interna e estocasticidade. Embora o ambiente *in loco* controle variáveis exógenas de rede e hardware, a estocasticidade intrínseca aos LLMs representa a principal ameaça à validade interna. A variabilidade nas respostas dos modelos pode gerar soluções disfuncionais mesmo sob prompts idênticos. O protocolo iterativo adotado mitiga parcialmente essa volatilidade, mas não exaure o espaço amostral de respostas possíveis. Adicionalmente, aponta-se uma limitação no isolamento do sistema de carga: enquanto o sistema-alvo foi reiniciado após cada teste, o motor de testes k6 permaneceu ativo durante a bateria de cada solução por razões de viabilidade técnica, o que poderia introduzir efeitos de borda residuais. Por fim, como parte das interações com os modelos ocorreu via interfaces web oficiais, não se pode descartar totalmente a ocorrência de atualizações silenciosas dos modelos ao longo da campanha experimental, o que constitui uma limitação adicional de reprodutibilidade.

Validade de construção e conclusão. A análise de dados envolve escolhas metodológicas que influenciam os resultados finais: (i) a definição de "otimização" é binária e baseada em limiares de superação da sobrecarga do sistema-base e (ii) ao reportar a melhor solução entre as repetições, o estudo foca no potencial máximo de otimização do modelo, o que pode superestimar a confiabilidade operacional típica em comparação com critérios de seleção mais conservadores.

Limitações operacionais. Devido ao alto custo temporal de execução em hardware físico e aos testes de longa duração de cada solução necessários para garantir estabilidade estatística, a escala do experimento foi limitada a quatro modelos e dois tipos de servidores. O planejamento experimental priorizou a fidelidade operacional, isolamento e mitigação de variáveis ambientais em detrimento de uma amostragem massiva, tornando a expansão do número de rodadas e sistemas-alvo um desafio de escalabilidade para pesquisas futuras.

7. Conclusão e Trabalhos Futuros

Este trabalho avaliou, em um cenário experimental deliberadamente favorável, o desempenho de quatro LLMs de ponta na otimização de servidores web. Mesmo nesse contexto simples, controlado e de baixa complexidade operacional, os resultados mostraram limitações importantes: **66,7%** das respostas preservaram a operacionalidade do serviço, mas apenas **43,8%** produziram otimizações efetivas, enquanto **33,3%** danificaram o sistema-alvo. Entre os modelos, houve heterogeneidade de desempenho, com melhor resultado global do Gemini 3 Pro e pior do DeepSeek-V3.2. Ainda assim, o achado mais robusto foi a influência do servidor-alvo: o **Apache** apresentou **66,7%** de sucesso, contra **20,8%** no **Nginx**, diferença estatisticamente significativa que permaneceu após o controle por modelo. Em conjunto, os resultados indicam que, embora LLMs possam contribuir para tarefas de otimização de servidores web, sua atuação ainda é inconsistente e envolve risco operacional relevante, o que torna necessária a supervisão humana e mecanismos adicionais de verificação antes de sua aplicação em ambientes reais.

Como trabalhos futuros, pretende-se realizar um estudo controlado com participantes humanos para investigar em que medida a assistência por LLMs melhora o desempenho de profissionais na otimização de servidores web e como esse desempenho se

compara ao dos próprios modelos. Também se pretende ampliar a escala amostral, de modo a avaliar com maior poder estatístico o efeito de iterações adicionais e aprofundar a investigação sobre por que o servidor-alvo exerceu influência mais forte e mais consistente do que o modelo sobre a probabilidade de sucesso.

8. Agradecimentos

Nabor C. Mendonça é parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), sob processo no. 313558/2023-0. Os assistentes de inteligência artificial gerativa Microsoft 365 Copilot, Google NotebookLM, ChatGPT e Gemini foram utilizados no apoio à concepção e à revisão textual deste artigo. Artefatos, ferramental e registros experimentais deste estudo encontram-se disponíveis publicamente em: `https://gitlab.com/jardelino/mia-2026-s-1`.

Referências

Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Sebastopol, CA.

Grafana Labs (2025). k6. Acessado em: 24 de outubro 2025.

He, X., Liu, Q., Du, M., Yan, L., Fan, Z., Huang, Y., Yuan, Z., and Ma, Z. (2025). Sweperf: Can language models optimize code performance on real-world repositories? *arXiv preprint arXiv:2507.12415*.

HighScalability (2013). The secret to 10 million concurrent connections: The kernel is the problem, not the solution. Acessado em: 24 de outubro 2025.

Kegel, D. (2003). The c10k problem. Acessado em: 24 de outubro 2025.

Patwardhan, T. et al. (2025). Gdpval: Evaluating ai model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*.

Spieker, H. et al. (2025). Prompting for performance: Exploring llms for configuring software. In *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 114–121. IEEE.

Vitui, A. and Chen, T.-H. (2025). Empowering aiops: Leveraging large language models for it operations management. *arXiv preprint arXiv:2501.12461*.

W3Techs (2025). Usage statistics and market shares of web servers. Acessado em: 24 de outubro 2025.

White, C. et al. (2025). Livebench: A challenging, contamination-free LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. (2023). Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Yang, Z., Bhatnagar, A., Qiu, Y., Miao, T., Tser Jern Kon, P., Xiao, Y., Huang, Y., Casado, M., and Chen, A. (2025). Cloud infrastructure management in the age of ai agents. *ACM SIGOPS Operating Systems Review*, 59(1):1–8.

Yi, L., Gay, G., and Leitner, P. (2025). An experimental study of real-life llm-proposed performance improvements. *arXiv preprint arXiv:2510.15494*.