UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

EVERTON SOUZA WIENCE

UMA ANÁLISE COMPARATIVA DO DESEMPENHO DO SERVIDOR NGINX COMO PROXY REVERSO COM E SEM CACHE EM APLICAÇÕES WEB

GUARAPUAVA

2025

EVERTON SOUZA WIENCE

UMA ANÁLISE COMPARATIVA DO DESEMPENHO DO SERVIDOR NGINX COMO PROXY REVERSO COM E SEM CACHE EM APLICAÇÕES WEB

A Comparative Evaluation of NGINX Performance as a Reverse Proxy with and without Caching in Web Application

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Sistemas para Internet do Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Profª. Drª. Sediane Carmem Lunardi Hernandes

Coorientador: Prof. Dr. Hermano Pereira

GUARAPUAVA

2025

RESUMO

O crescente uso de aplicações *web* tem ampliado a necessidade de soluções capazes de melhorar o desempenho, reduzir a latência e otimizar o consumo de recursos em servidores. Nesse contexto, o *Nginx* destaca-se por sua arquitetura eficiente e por sua ampla utilização como servidor *web* e *proxy reverso*, especialmente quando combinado com mecanismos de *cache*. Este projeto investiga o impacto do uso do *Nginx* como *proxy reverso* com e sem *cache* no desempenho de páginas estáticas, dinâmicas e ativas, por meio de experimentos controlados. A metodologia envolve a construção de ambiente de testes, execução de cenários com diferentes clientes e análise das métricas *Speed Index*, consumo de *Central Processing Unit (CPU)* e uso de memória. Os resultados esperados incluem a comprovação de que o uso do *Nginx* como *proxy* reverso com *cache* pode diminuir significativamente a latência e melhorar o aproveitamento dos recursos computacionais, promovendo ambientes mais eficientes. O impacto dessa melhoria pode se refletir em uma experiência de usuário mais fluida e em maior competitividade para empresas que adotam tais estratégias.

Palavras-chave: proxy reverso; cache; nginx; desempenho; aplicações web.

ABSTRACT

The increasing reliance on web applications has intensified the need for solutions that improve performance, reduce latency, and optimize resource usage on servers. In this context, Nginx stands out due to its efficient architecture and its wide adoption as both a web server and a reverse proxy, especially when combined with caching mechanisms. This project investigates the impact of using Nginx as a reverse proxy with and without cache on the performance of static, dynamic, and active web pages through controlled experiments. The methodology involves building a test environment, running scenarios with different clients, and analyzing metrics such as CPU consumption and memory usage. Expected results include demonstrating that using NGINX as a reverse proxy with caching can significantly reduce latency and improve the utilization of computing resources, promoting more efficient environments. The impact of this improvement can translate into a smoother user experience and greater competitiveness for companies that adopt such strategies.

Keywords: reverse proxy; cache; nginx; performance; web applications.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Abreviaturas

art.	Artigo
cap.	Capítulo
sec.	Seção

Siglas

ABNT	Associação Brasileira de Normas Técnicas
AVA	Ambiente Virtual de Aprendizagem
CC	Creative Commons
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPU	Central Processing Unit
CSS	Cascading Style Sheets
EPS	*Encapsulated PostScript*
HTML	Hypertext Markup Language
HTTP	Hyper Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
LAN	Local Area Network
NGINX	Servidor web (open source de alto desempenho)
PDF	Formato de Documento Portátil, do inglês *Portable Document Format*
PS	*PostScript*
RAM	Random Access Memory
SSD	Solid State Drive
TCC	Trabalho de Conclusão de Curso
UTFPR	Universidade Tecnológica Federal do Paraná

WAN Wide Area Network

SUMÁRIO

1 INTRODUÇÃO

As aplicações *web* têm-se tornado parte essencial da vida contemporânea, sendo utilizadas em diversos setores, como comunicação, entretenimento, comércio eletrônico e sistemas corporativos. Com a crescente digitalização da sociedade, tais aplicações passaram a desempenhar um papel crítico não apenas no cotidiano dos usuários, mas também no funcionamento de empresas e instituições. Nesse cenário, aspectos como desempenho e disponibilidade são fundamentais para garantir experiências satisfatórias e confiáveis por parte dos usuários (Sá; SOUZA, 2015).

Entre as técnicas disponíveis, destaca-se a utilização de *proxy* reverso aliado ao uso de mecanismos de *cache* em servidores *web*. O *proxy* reverso atua como intermediário entre os clientes e um ou mais servidores de aplicação, trazendo benefícios como balanceamento de carga, proteção contra acessos indevidos e armazenamento temporário de conteúdos em *cache* (CLOUDFLARE, 2024b). O *cache*, por sua vez, evita o processamento repetido de requisições idênticas, reduzindo a latência percebida pelo usuário e otimizando o uso de *Central Processing Unit* (CPU) e memória do servidor (AIVALIOTIS, 2016).

Nesse contexto, o servidor *Nginx* tem ganhado destaque devido à sua ampla adoção em ambientes corporativos e acadêmicos, além de sua eficiência no tratamento de requisições simultâneas. Reconhecido por sua capacidade de atuar tanto como servidor *web* quanto como *proxy* reverso, o *Nginx* oferece recursos avançados de caching que permitem melhorar o desempenho de aplicações *web* em diferentes cenários (AIVALIOTIS, 2016).

Diante disso, este trabalho busca investigar o impacto da utilização do servidor *Nginx* como *proxy* reverso com e sem *cache* em aplicações *web* estáticas, dinâmicas e ativas. Pretende-se verificar se o uso de *cache* contribui no desempenho dessas aplicações de web.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo geral do trabalho consiste em realizar uma análise comparativa do servidor *Nginx* em diferentes configurações, especificamente sua utilização como *proxy* reverso com *cache* e como *proxy* reverso sem cache, a fim de avaliar os impactos de desempenho sobre aplicações *web* estáticas, dinâmicas e ativas.

1.1.2 Objetivos específicos

- Configurar o *Nginx* como *proxy* reverso considerando as duas abordagens propostas (i.e., com e sem *cache*).

- Realizar um levantamento de aplicações/serviços *web* de diferentes portes para compor os cenários de testes experimentais.
- Sintetizar os dados coletados em testes de carga realizados em diferentes cenários de simulação.
- Avaliar os resultados coletados propondo boas práticas de configuração de *proxy* para os diferentes tipos de aplicações.

1.2 Justificativa

Com a crescente dependência de serviços *web* em diversos setores, o desempenho dos servidores tornou-se um fator crítico para a experiência do usuário. Páginas lentas não apenas impactam a satisfação e a retenção de usuários, mas também podem gerar prejuízos financeiros significativos (CLOUDFLARE, 2019). O tempo de resposta elevado, especialmente em requisições de conteúdo estático (como imagens, arquivos CSS e JavaScript) é um problema. Sem um *proxy* reverso, o servidor de origem precisa processar cada solicitação, gerando latência e sobrecarga. Além disso, o uso do *cache* no *Nginx* pode ajudar neste sentido. Ele armazena uma cópia do conteúdo na primeira requisição e, a partir da segunda, entrega a resposta diretamente do *cache*. Esse mecanismo tem como objetivo reduzir o tempo de espera do cliente, oferecendo uma experiência de navegação mais rápida e fluida.

Além disso, servidores *web* sem *cache* precisam de mais recursos de CPU e memória para lidar com o tráfego, o que pode levar a um gargalo de desempenho. A arquitetura orientada a eventos do *Nginx*, combinada com o *cache*, libera o servidor de origem de grande parte das requisições repetidas. Assim, com o *Nginx* servindo as respostas em *cache*, o servidor original pode focar apenas em processar novas requisições. Com isso, a carga computacional do servidor diminui e tem-se uma otimização dos recursos existentes.

Nesse contexto, a investigação do uso do *Nginx* como *proxy* e como *proxy* reverso com e sem *cache* justifica-se por seu potencial em solucionar problemas práticos de desempenho e escalabilidade. O *Nginx*, em sua configuração com *cache*, atua como uma camada intermediária que otimiza o fluxo de tráfego, servindo como uma solução eficiente e de baixo custo para ambientes de alta demanda (CLOUDFLARE, 2024b).

Além disso, este estudo contribui para a literatura e para a prática profissional. Ao oferecer uma análise aplicada e a demonstração dos benefícios do *Nginx*, o trabalho se posiciona como um estudo de caso valioso. A relevância no mercado é evidente, uma vez que a técnica é amplamente adotada por grandes empresas de tecnologia.

1.3 Estrutura do trabalho

Este trabalho está organizado da seguinte forma. O Capítulo 1 apresenta a introdução, objetivos e justificativa. O Capítulo 2 aborda o referencial teórico sobre servidores *web*, *Nginx*, *proxy* reverso e *cache*. Os materiais e métodos que serão utilizados para o desenvolvimento deste Trabalho de Conclusão de Curso são descritos no Capítulo 3 e o Capítulo 4 apresenta as considerações finais.

2 REFERENCIAL TEÓRICO

2.1 Servidores *web*

Os servidores *web* são *softwares* responsáveis por receber e processar requisições do *Hyper Transfer Protocol* (HTTP) realizadas por clientes (navegadores ou outras aplicações) e retornar os recursos solicitados, como páginas *Hypertext Markup Language* (HTML), imagens, arquivos ou dados estruturados (AIVALIOTIS, 2016). Tradicionalmente, os servidores *web* desempenham papel central na arquitetura cliente-servidor da *web*, atuando como ponto de contato entre o usuário e os sistemas hospedados (AIVALIOTIS, 2016). A Figura 1 ilustra um servidor servindo a dois usuários na Internet.

Dentre as características mais relevantes de um servidor *web*, destacam-se: capacidade de lidar com múltiplas conexões simultâneas, suporte a protocolos modernos (como HTTP/2 e HTTP/3), escalabilidade, segurança e compatibilidade com diferentes tecnologias de aplicação. O desempenho do servidor *web* é um fator determinante para a experiência do usuário final e para a eficiência no consumo de recursos computacionais (BUTLER, 2017).

Alguns exemplos de servidores *web* são o *Apache HTTP Server*, o *Lighttpd* e o *Nginx*.

2.1.1 O Servidor *Nginx*

O *Nginx* é um servidor *web* que utiliza uma arquitetura baseada em eventos assíncronos, permitindo o processamento simultâneo de múltiplas conexões sem a necessidade de criar uma nova *thread* para cada requisição (AIVALIOTIS, 2016). Essa arquitetura é composta por um processo principal (*master process*) e por processos de trabalho (*worker processes*) conforme ilustrado na Figura 2. O processo principal é responsável por carregar a configuração e

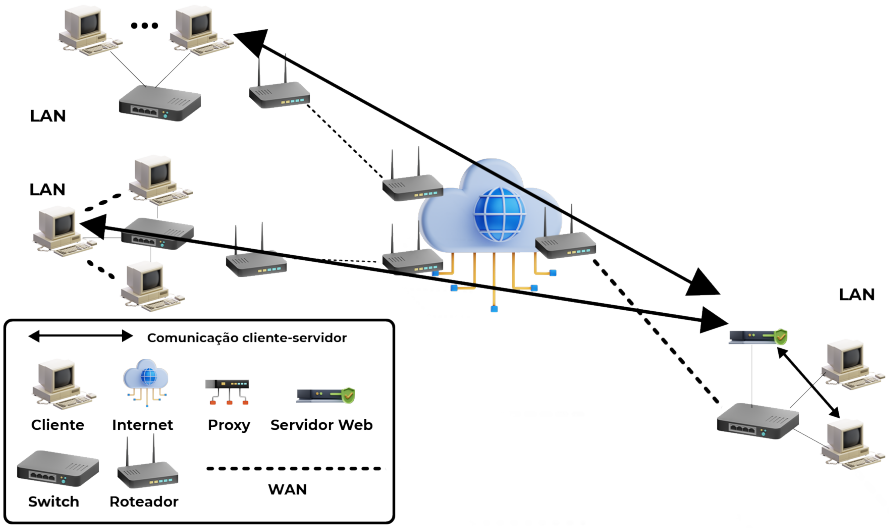

Figura 1 – Ilustração de um servidor web. (FOROUZAN; MOSHARRAF, 2013).

gerenciar os processos de trabalho. Os processos de trabalho utilizam um modelo de *loop* de eventos não bloqueante (*event loop*), isto é, um mecanismo em que um único processo monitora e gerencia diversas conexões simultaneamente, reagindo a eventos (como recebimento de dados ou novas requisições) sem a necessidade de aguardar a conclusão de cada operação. Dessa forma, o servidor não fica parado enquanto uma conexão está inativa. Como consequência, cada processo trabalhador pode lidar com milhares de conexões concorrentes utilizando uma única *thread*, o que reduz o consumo de memória e aumenta a escalabilidade do servidor.

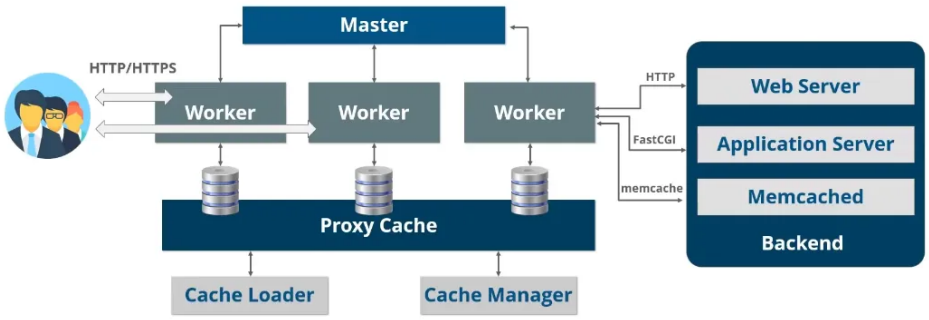

Figura 2 – Arquitetura do Nginx (YVS,).

Além de atuar como servidor *web*, o *Nginx* também pode funcionar como *proxy*, *proxy* reverso, balanceador de carga, *proxy* para servidor de e-mails (IMAP, POP3 e SMTP) e servidor de *streaming* (BUTLER, 2017).

O *Nginx Cookbook* (BUTLER, 2017) destaca casos práticos de configuração que demonstram ganhos de desempenho significativos em aplicações que lidam com tráfego intenso e requisições repetidas.

2.2 *Proxy* e *proxy reverso*

Um *proxy* é um servidor intermediário utilizado entre o cliente e o servidor (CLOUDFLARE, 2024b). Assim, quando um cliente faz uma requisição para o servidor, o *proxy* intercepta a requisição e se comunica com o servidor como um intermediário, ou seja, em nome do servidor. O *proxy* recebe a resposta e encaminha ao cliente. O cliente não acessa diretamente o servidor do domínio, mas passa pelo *proxy*, ou seja, acessa a Internet através dele. No contexto de redes e aplicações web, o *proxy* pode desempenhar funções como filtragem de tráfego, anonimização, restrição de acesso ou *caching* de conteúdos. A Figura 3 ilustra a arquitetura de um servidor *proxy*.

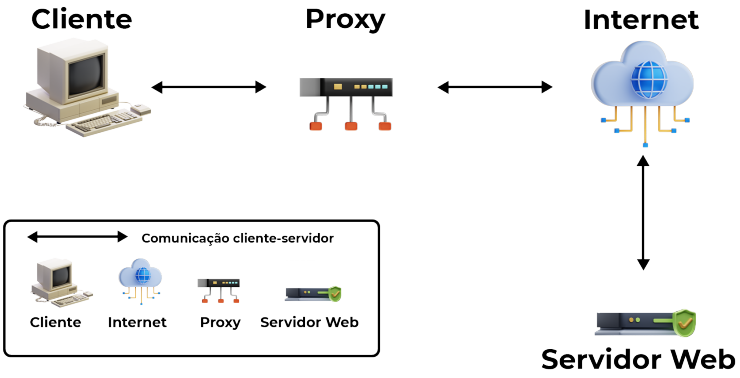

Figura 3 – Arquitetura de um servidor *proxy*.

O *proxy* reverso, por outro lado, atua de forma oposta, ficando próximo do servidor de origem (CLOUDFLARE, 2024b). O cliente sempre se comunica com esse *proxy* reverso como se ele fosse o próprio servidor do domínio. Logo, o *proxy* reverso intercepta as solicitações, encaminha-as ao servidor, recebe a resposta e envia para o cliente. A Figura 4 ilustra a arquitetura de um *proxy* reverso.

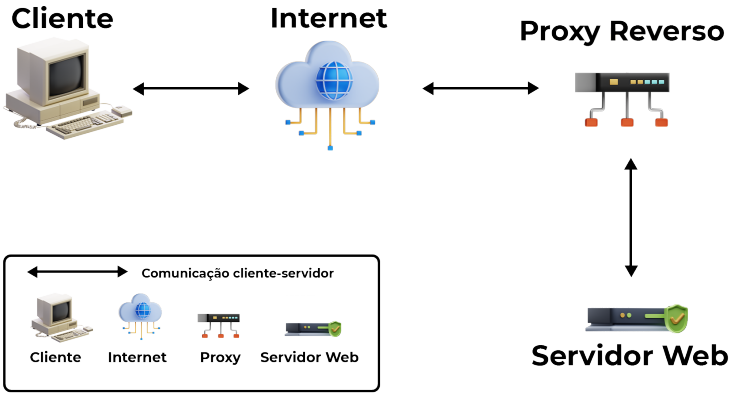

Figura 4 – Arquitetura de um *proxy* reverso.

Entre os principais benefícios do uso de *proxy* reverso, destacam-se (CLOUDFLARE, 2024b):

- Balanceamento de carga: que permite requisições sejam distribuídas entre múltiplos servidores.
- Segurança: para proteger os servidores de aplicação contra ataques diretos.
- *Caching*: para armazenar temporariamente conteúdos e reduzindo a necessidade de processamento repetitivo.

Em resumo, utilizando o *proxy*, o cliente acessa a Internet através deste, enquanto que com o *proxy* reverso, o cliente acessa um servidor, mas a requisição passa primeiro pelo *proxy* reverso que representa o servidor.

Cabe salientar que o balanceador de carga (*load balancer* no *Nginx* é um tipo específico de *proxy* reverso responsável por distribuir as requisições entre múltiplos servidores de aplicação (AIVALIOTIS, 2016). O *load balancer* no *Nginx* utiliza os algoritmos *Round Robin*, *Least Connections* e *IP-Hash*, o que melhora a disponibilidade e o desempenho do sistema.

2.3 Mecanismos de *cache* em aplicações *web*

Um *cache* é um local de armazenamento temporário para cópias de arquivos ou outros tipos de dados que busca oferecer às aplicações acesso mais rápido (CLOUDFLARE, 2024a). Em aplicações *web*, o mecanismo de *cache* também é utilizado.

Desta forma, existem diferentes níveis de *cache* para aplicações *web*:

1. *Cache* no navegador, armazenando recursos no dispositivo do cliente.
2. *Cache* em *proxies*, interceptando tráfego e guardando conteúdos frequentemente requisitados.
3. *Cache* no servidor de aplicação, otimizando respostas dinâmicas.

No contexto do *Nginx*, o cache integrado ao *proxy* reverso permite que respostas a requisições repetidas sejam entregues rapidamente sem necessidade de acessar o servidor de aplicação, diminuindo o uso de *Central Processing Unit* (CPU) e memória. Quanto à política de *cache* no *Nginx*, especialmente quando utilizado como *proxy* reverso, o *cache* precisa ser habilitado e especificado: o tamanho do *cache*, quais tipos de páginas (por exemplo, as que possuem código 200 e 404) e por quanto tempo serão armazenadas.

2.4 Trabalhos relacionados

Diversos estudos têm investigado o desempenho de servidores web, seja em análises experimentais ou em revisões de literatura. O objetivo comum desses trabalhos é identificar vantagens e limitações de soluções como Apache e *Nginx*, assim como propor arquiteturas para otimização em contextos específicos.

Sá e Souza Sá e Souza (2015) propuseram uma arquitetura baseada em *proxy reverso* utilizando o Varnish para melhorar o desempenho do Moodle, um Ambiente Virtual de Aprendizagem (AVA). Os testes foram realizados com diferentes números de usuários simultâneos (50, 100, 150 e 200) e mostraram ganhos significativos de latência e disponibilidade ao empregar o cache reverso.

Kunda, Chihana e Muwane Kunda, Chihana e Muwane (2017) realizaram uma revisão sistemática da literatura sobre o desempenho dos servidores Apache e *Nginx*. O estudo reuniu resultados de múltiplos experimentos já publicados, destacando que o *Nginx* apresenta melhor escalabilidade, menor consumo de memória e CPU em cenários de alto volume de conexões, enquanto o Apache tende a obter vantagem em conteúdos dinâmicos, como scripts PHP.

Recentemente, Pinastawa, Pradana e Maulana Pinastawa, Pradana e Maulana (2024) conduziram um estudo experimental comparando Apache e *Nginx* com o uso da ferramenta K6. Foram aplicados diferentes métodos de teste: carga (*load*), pico (*spike*), saturação (*soak*) e desempenho (*performance*). Os resultados mostraram que o *Nginx* obteve melhor desempenho geral, com tempos de resposta mais baixos, menor taxa de falhas e maior quantidade de requisições processadas por segundo, especialmente em cenários de saturação e carga constante.

Esses trabalhos evidenciam a relevância da análise comparativa de servidores web. Enquanto revisões de literatura consolidam tendências gerais de desempenho. Já estudos experimentais recentes reforçam a superioridade do *Nginx* em situações de alta concorrência, ao passo que o Apache mantém relevância em aplicações que demandam maior flexibilidade e suporte a conteúdos dinâmicos. Contudo, o presente trabalho irá realizar testes de desempenho mais abrangentes, do *Nginx* em diferentes ambientes com aplicações dinâmicas e ativas.

3 MATERIAIS E MÉTODOS

Neste capítulo são descritos os recursos que serão utilizados e os procedimentos que serão adotados para a realização dos experimentos. O estudo será conduzido em ambiente controlado, utilizando servidores configurados especificamente para a análise comparativa proposta.

3.1 Materiais

Os experimentos serão realizados em dois computadores com as seguintes especificações:

1. **Computador 01 - servidores web**:
    - **Sistema operacional**: Ubuntu Server 22.04 LTS;
    - **Processador**: i5-94000f 2.90GHz(6CPUs);
    - **Memória RAM**: 24 GB;
    - **Armazenamento**:100 GB SSD;
    - **Rede**: 1 Gbps.
2. **Computador 02 - clientes e servidor *proxy***:
    - **Sistema operacional**: Ubuntu Server 22.04 LTS;
    - **Processador**: AMD-Phenom II, 3.2GHz;
    - **Memória RAM**: 8 GB;
    - **Armazenamento**: 50 GB SSD;
    - **Rede**: 1 Gbps.

O servidor *Nginx* será instalado em sua versão estável, com configuração inicial padrão e, posteriormente, modificado para ser *proxy* reverso com cache e *proxy* reverso sem cache. Para a simulação dos cenários de uso serão utilizados três tipos de aplicações:

- **Aplicação estática**: conjunto de páginas HTML com recursos estáticos (imagens, CSS, JavaScript);
- **Aplicação dinâmica**: aplicação web baseada em PHP e banco de dados MySQL.
- **Aplicação ativa**: aplicação web baseada em PHP e banco de dados MySQL com processamento do lado cliente.

3.2 Ferramentas

Nos clientes web, a ferramenta que será utilizada será o Lighthouse [1]. O **Lighthouse** é uma ferramenta automatizada de auditoria desenvolvida pelo Google, capaz de analisar páginas *web* em diversos aspectos, como desempenho, acessibilidade, melhores práticas e otimização para mecanismos de busca. Já o **wrk** é uma ferramenta de linha de comando voltada para *benchmarking* de servidores HTTP, permitindo simular múltiplas conexões simultâneas e medir métricas de desempenho como latência, tempo médio de resposta e número de requisições por segundo, se necessário.

3.3 Métodos

O método adotado consiste em três etapas principais:

1. **Configuração do ambiente**: instalação do servidor *Nginx* e configuração em dois modos distintos — como *proxy* reverso com *cache* e como *proxy reverso* sem *cache*. O servidor *Nginx* será configurado para a versão 1.1 do protocolo *Hypertext Transfer Protocol* (HTTP) sem o uso do *Transport Layer Security* (TLS) para a comunicação entre clientes e servidores. A Figura 5 representa a configuração do ambiente em que o uso do cache não é representado.

2. **Definição dos cenários**: cada cenário será composto por 3 servidores e N clientes web considerando o servidor *Nginx* como *proxy* reverso com *cache* e *proxy* reverso sem *cache*. Cada servidor será responsável por um tipo específico de página web, e cada cliente realizará de 1 até *N* requisições para cada servidor específico. Logo, tem-se:

    - servidor 1 para páginas estáticas;
    - servidor 2 para páginas dinâmicas;
    - servidor 3 para páginas ativas.

Tabela 1 – Cenários de teste com o Nginx em diferentes modos de operação.

Modo de operação	Servidor 1	Servidor 2	Servidor 3
Nginx como *proxy* reverso **sem cache**	1 ... N	1 ... N	1 ... N
Nginx como *proxy* reverso **com cache**	1 ... N	1 ... N	1 ... N

Fonte: Autoria própria.

O valor N simulará acessos de 50, 100, 150 e 200 usuários simultâneos para o *proxy reverso* conforme Sá e Souza (2015). Além disso, cada cenário será isolado para evitar

[1] Ligthouse - https://developer.chrome.com/docs/lighthouse/overview?hl=pt-br

interferências externas, e cada cliente requisitará páginas estáticas, dinâmicas e ativas dos servidores específicos.

3. **Execução dos testes**: serão realizados testes de carga, simulando diferentes quantidades de requisições por usuário simultaneamente, e testes de desempenho para verificar a latência no carregamento das páginas (ou seja, a duração desde o início do processo de carregamento da página até o ponto em que a maior parte do conteúdo aparece na tela). A ferramenta `wrk` simulará diferentes quantidades de usuários simultâneos.

4. **Coleta e análise de métricas**: durante os experimentos a métrica *Speed Index* será utilizada nos clientes. O *Speed Index* é um indicador amplamente utilizado para avaliar a **velocidade de renderização visual** de uma página *web*. Essa métrica mede o tempo necessário para que o conteúdo visível da página seja exibido ao usuário, atribuindo um valor menor quanto mais rápida for a renderização. De acordo com o *Web Performance Working Group* e a ferramenta **Lighthouse**, o *Speed Index* é calculado com base na progressão visual da página durante o carregamento, refletindo a experiência real do usuário em termos de percepção de desempenho (DEVELOPERS, 2024). No servidor *proxy* as métricas utilizadas serão o consumo de CPU e o uso de memória.

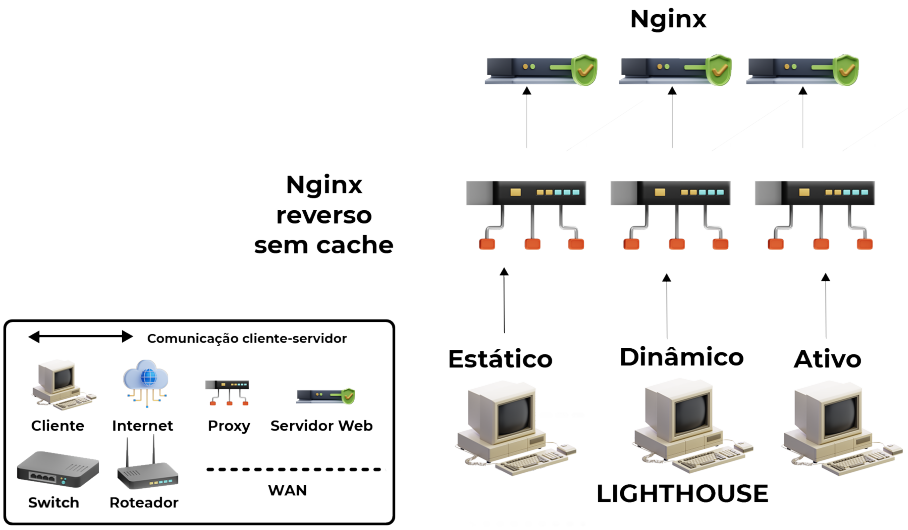

Figura 5 – Ilustração adaptada Nginx como *proxy* reverso (DEVELOPERS, 2024).

Os resultados obtidos serão organizados em tabelas e gráficos comparativos, permitindo avaliar o desempenho do servidor com as configurações escolhidas para os diferentes tipos de aplicações utilizadas e configurações do servidor. A análise comparativa buscará identificar qual configuração do servidor é melhor para cada tipo de aplicação.

4 CONSIDERAÇÕES FINAIS

Este projeto de Trabalho de Conclusão de Curso tem como foco a investigação dos efeitos da configuração do servidor *Nginx* como *proxy* reverso com *cache* e *proxy* reverso sem *cache* em aplicações web. A análise será conduzida de modo a verificar a experiência real do usuário em termos de percepção de desempenho no carregamento de suas páginas web (DEVELOPERS, 2024).

A importância desta pesquisa aplicada está diretamente ligada à crescente demanda por sistemas web mais ágeis, escaláveis e confiáveis, especialmente em um contexto em que as organizações dependem fortemente de plataformas digitais. Com isso, espera-se que os resultados beneficiem tanto o meio acadêmico — ao consolidar práticas recomendadas — quanto profissionais da área de tecnologia, ao fornecer orientações técnicas aplicáveis no cotidiano.

Entre os principais resultados almejados, destaca-se a comprovação de que o uso de *proxy* reverso com *cache* no *Nginx* pode diminuir significativamente a latência e melhorar o aproveitamento dos recursos computacionais, promovendo ambientes mais eficientes. O impacto dessa melhoria pode se refletir em uma experiência de usuário mais fluida e em maior competitividade para empresas que adotam tais estratégias.

Ainda assim, o projeto reconhece a existência de desafios, como a criação de cenários de teste que representem fielmente o uso real, a escolha criteriosa das métricas de avaliação e a consideração das variações de desempenho conforme o tipo de aplicação. Esses aspectos exigirão atenção especial ao longo da pesquisa para assegurar a precisão e a confiabilidade dos resultados obtidos.

REFERÊNCIAS

AIVALIOTIS, D. **Mastering NGINX - Second Edition**. Birmingham: Packt Publishing, 2016.

BUTLER, T. **NGINX Cookbook: Over 70 recipes for real-world configuration, deployment, and administration**. Birmingham: Packt Publishing, 2017.

CLOUDFLARE. **Getting Faster: Website Performance Whitepaper**. 2019. https://www.cloudflare.com/resources/assets/.../Getting-faster-website-performance-whitepaper_Q42019.pdf. Dados sobre impacto do tempo de carregamento em conversão, satisfação e retenção.

CLOUDFLARE. **O que é caching?** 2024. Https://www.cloudflare.com/pt-br/learning/cdn/what-is-caching. Acesso em: 10 set. 2025.

CLOUDFLARE. **O que é um proxy reverso?** 2024. Https://www.cloudflare.com/pt-br/learning/cdn/glossary/reverse-proxy/. Acesso em: 8 out. 2025.

DEVELOPERS, G. **Measure performance with Speed Index**. 2024. Acesso em: 6 out. 2025. Disponível em: https://developer.chrome.com/docs/lighthouse/performance/speed-index/.

FOROUZAN, B. A.; MOSHARRAF, F. **Redes de Computadores**. Porto Alegre: Grupo A, 2013. E-book. ISBN 9788580551693.

KUNDA, D.; CHIHANA, S.; MUWANE, S. Web server performance of apache and nginx: A systematic literature review. **International Journal of Computer Applications**, v. 179, n. 47, p. 1–7, nov. 2017.

PINASTAWA, I. W. R.; PRADANA, M. G.; MAULANA, N. Web server performance evaluation: Comparing nginx and apache using k6 testing methods for load, spike, soak, and performance. *In*: **Proceedings of the 2024 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)**. [*S.l.*]: IEEE, 2024. p. 1002–1010.

Sá, S. S. de; SOUZA, S. C. de. Uma proposta de arquitetura para melhoria de desempenho no ambiente virtual de aprendizagem moodle utilizando proxy reverso. *In*: **Quinta Conferência de Diretores de Tecnologia de Informação e Comunicação em Instituições de Educação Superior**. Brasil: [*s.n.*], 2015. p. 1–12.

YVS, S. A. **Nginx Architecture, Use Cases, and Examples**. Disponível em: https://www.linkedin.com/pulse/nginx-architecture-use-cases-examples-sai-ashish-yvs1c/.