INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

HERMÍNIO DE BARROS E SILVA NETO

ESTRATÉGIAS DE ESCALABILIDADE PARA SISTEMAS SOB ALTA DEMANDA:

análise comparativa de performance, custo e disponibilidade

TERESINA, PIAUÍ
2025

HERMÍNIO DE BARROS E SILVA NETO

ESTRATÉGIAS DE ESCALABILIDADE PARA SISTEMAS SOB ALTA DEMANDA:
análise comparativa de performance, custo e disponibilidade

Trabalho de Conclusão de Curso (monografia) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Orientador: Prof. M^{e}. Fernando Castelo Branco Gonçalves Santana

TERESINA, PIAUÍ
2025

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

Silva Neto, Hermínio de Barros e

S586e Estratégias de escalabilidade para sistemas sob alta demanda : análise comparativa de performance, custo e disponibilidade / Hermínio de Barros e Silva Neto. - 2025.

48 f.: il. color.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central, 2025.

Orientador : Prof Me. Fernando Castelo Branco Gonçalves Santana.

1. Escalabilidade vertical. 2. Escalabilidade horizontal. 3. Sistemas distribuídos. 4. Alta demanda. I.Título.

CDD - 004.21

Elaborado por Tanize Maria Sales CRB 3/1024

HERMÍNIO DE BARROS E SILVA NETO

ESTRATÉGIAS DE ESCALABILIDADE PARA SISTEMAS SOB ALTA DEMANDA:
análise comparativa de performance, custo e disponibilidade

Trabalho de Conclusão de Curso (monografia) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Aprovada em 17 de Dezembro de 2025.

BANCA EXAMINADORA:

Prof. Mᵉ. Fernando Castelo Branco Gonçalves Santana(Orientador)
Instituto Federal do Piauí (IFPI)

Prof. Dr. Ricardo Martins Ramos
Instituto Federal do Piauí (IFPI)

Profª. Drª. Nadia Mendes dos Santos
Instituto Federal do Piauí (IFPI)

Dedico este trabalho à minha família, aos meus amigos e aos meus educadores.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu forças para prosseguir em momentos que pensei em desistir.

Agradeço à minha família e aos meus amigos, que sempre estiveram ao meu lado e me deram todo o apoio necessário para que eu chegasse até aqui.

Agradeço aos meus gestores e colegas de trabalho que auxiliaram a mim e meus amigos durante toda essa etapa final do curso.

Agradeço ao meu orientador, Professor Fernando Santana, por todo o acolhimento e direcionamento, e a todos os meus demais professores que muito me ensinaram e me inspiraram nessa jornada.

"A imaginação é mais importante que o conhecimento,
porque o conhecimento é limitado, ao passo que
a imaginação abrange o mundo inteiro"
Albert Einstein

RESUMO

Este trabalho analisa comparativamente duas estratégias amplamente utilizadas para ampliar a capacidade de sistemas sujeitos a altos volumes de requisições: a escalabilidade vertical (*scale-up*) e a escalabilidade horizontal (*scale-out*). A investigação parte de uma revisão bibliográfica sistemática, que reúne estudos recentes sobre desempenho, elasticidade, custo e disponibilidade em ambientes distribuídos, e é complementada por um experimento empírico desenvolvido especificamente para esta pesquisa. No experimento, três cenários foram avaliados com medições de *throughput*, latência e comportamento sob carga crescente. Os resultados mostraram que, embora o aumento vertical proporcione ganhos relevantes de desempenho com baixa complexidade operacional, sua eficiência é limitada por restrições físicas de *hardware* e crescimento não linear de custos. A escalabilidade horizontal, por sua vez, demonstrou maior capacidade de absorver picos abruptos de tráfego, alcançando *throughput* significativamente superior e latências mais estáveis, além de reduzir riscos associados a pontos únicos de falha. Entretanto, essa abordagem exige maior maturidade técnica, ferramentas de orquestração e cuidados adicionais com consistência distribuída. A análise integrada evidencia que a escolha entre escalar verticalmente ou horizontalmente não é universal, mas depende do perfil da aplicação, do orçamento disponível, da previsibilidade da carga e dos requisitos de disponibilidade. O estudo conclui que sistemas sujeitos a grandes variações de demanda tendem a obter melhores resultados com escalabilidade horizontal, enquanto aplicações de menor porte ou com arquitetura legada podem se beneficiar de soluções verticais. Por fim, são apresentadas recomendações práticas para orientar decisões arquiteturais e sugestões de pesquisas futuras envolvendo abordagens híbridas e estratégias preditivas de *autoscaling*.

Palavras-chaves: escalabilidade vertical; escalabilidade horizontal; sistemas distribuídos; alta demanda; autoscaling.

ABSTRACT

This work presents a comparative analysis of two widely adopted strategies for increasing the capacity of systems that handle high request volumes: vertical scalability (scale-up) and horizontal scalability (scale-out). The study is based on a systematic literature review that brings together recent findings on performance, elasticity, cost, and availability in distributed environments, and is complemented by an empirical experiment designed specifically for this research. Three scenarios were evaluated using metrics such as throughput, latency, and behavior under increasing load. The results indicate that, although vertical scaling offers relevant performance improvements with lower operational complexity, it is constrained by physical hardware limits and exhibits nonlinear cost growth. Horizontal scaling, on the other hand, demonstrated a greater ability to absorb sudden traffic spikes, achieving significantly higher throughput and more stable latency, while reducing risks associated with single points of failure. However, this approach introduces additional challenges related to distributed consistency, orchestration tools, and operational overhead. The integrated analysis shows that the choice between scaling vertically or horizontally is not universal; instead, it depends on application characteristics, budget constraints, workload predictability, and availability requirements. The study concludes that systems exposed to highly variable or unpredictable demand tend to benefit more from horizontal scaling, whereas smaller or legacy applications may find vertical scaling more adequate. Finally, the work provides practical recommendations for architectural decision-making and suggests future research directions involving hybrid strategies and predictive autoscaling techniques.

Keywords: vertical scalability; horizontal scalability; distributed systems; high demand; autoscaling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
BASE	Basically Available, Soft state, Eventually consistent
CPU	Central Process Unit
FaaS	Function as a Service
HPA	Horizontal Pod Autoscaler
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
MTBF	Mean Time Between Failures
MTTF	Mean Time To Repair
OSI	Open Systems Interconnection
RAM	Random Access Memory
REST	Representational State Transfer
SLA	Service Level Agreement
SPOF	Single Point of Failure
TCO	Total Cost of Ownership
VPA	Vertical Pod Autoscaler

SUMÁRIO

1 INTRODUÇÃO

Com o uso cada vez mais intenso da tecnologia no dia a dia, desde serviços digitais básicos até plataformas complexas utilizadas globalmente, a necessidade de projetar sistemas capazes de lidar com altos volumes de acesso se tornou uma preocupação central no desenvolvimento de aplicações modernas (ATIEWI *et al.*, 2020; GORTON, 2021). Esse cenário é ampliado pelo avanço de soluções conectadas, como dispositivos de Internet das Coisas (IoT), e pelo crescimento contínuo do comércio eletrônico, que elevam significativamente a demanda por infraestrutura escalável (ATIEWI *et al.*, 2020). Em especial, sistemas que enfrentam grandes picos de tráfego, como plataformas de *e-commerce* durante eventos promocionais como *Black Friday* e *Cyber Monday*, portais governamentais durante períodos de inscrição para concursos públicos e serviços de *streaming* durante grandes lançamentos, apresentam desafios de arquitetura em que a escolha de uma boa estratégia de escalabilidade impacta diretamente em receita, disponibilidade e custos operacionais (ARAÚJO, 2025; GUPTA *et al.*, 2023).

Diante de aplicações que utilizam tecnologias cada vez mais sofisticadas e que precisam operar de forma contínua em ambientes de produção, a adoção de arquiteturas capazes de absorver grandes variações de carga sem perda significativa de desempenho ou violações de *Service Level Agreements* (SLA) se torna essencial, principalmente em ambientes de computação em nuvem, onde elasticidade e alta disponibilidade são requisitos críticos (CÂNDIDO; SOUSA; SILVA, 2022; CASALICCHIO; SILVESTRI, 2013). Nesse contexto, engenheiros de software frequentemente se deparam com uma decisão de arquitetura central: adotar escalabilidade vertical (*scale-up*), por meio do aumento de recursos computacionais em um único servidor, ou escalabilidade horizontal (*scale-out*), que distribui a carga de processamento entre múltiplos nós autônomos (GORTON, 2021; JEONG; KIM; LEE, 2025).

Ambas as estratégias apresentam *trade-offs* técnicos, econômicos e operacionais distintos, que dependem do contexto de aplicação, orçamento disponível, previsibilidade da carga e requisitos de disponibilidade. Uma análise sistemática desses *trade-offs*, considerando não apenas performance mas também custo e disponibilidade, é essencial para orientar decisões arquiteturais fundamentadas em evidências (FÉ *et al.*, 2022; JEONG; KIM; LEE, 2025)

Estudos na literatura indicam que, em sistemas com demanda alta e volátil, caracterizados por variações de carga superiores a 300% entre períodos regulares e picos, a escalabilidade horizontal tende a oferecer melhor relação custo–benefício no longo prazo e maior resiliência operacional, desde que acompanhada de práticas adequadas de monitoramento e operação (FÉ *et al.*, 2022; LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014; GUNTHER, 2000). Ainda assim, essa estratégia exige investimento inicial

maior em arquitetura, ferramentas de orquestração e capacitação da equipe técnica (HASSAN, 2015).

Nesse cenário, torna-se relevante analisar de forma sistemática como as estratégias de escalabilidade vertical e horizontal se comportam sob alta demanda de modo a orientar decisões arquiteturais fundamentadas em evidências.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho foi analisar e comparar as estratégias de escalabilidade vertical e horizontal em aplicações que lidam com alto volume de usuários e picos extremos de requisições, de modo a avaliar seus impactos em desempenho, custo total de propriedade e disponibilidade.

1.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos:

- Definir e caracterizar os conceitos, mecanismos operacionais e implicações técnicas das escalabilidades vertical e horizontal, evidenciando suas principais vantagens, limitações e *trade-offs* em contextos de alta demanda.
- Analisar criticamente os limites físicos, econômicos e operacionais das duas abordagens nesses cenários, com base tanto em um experimento próprio quanto em estudos documentados na literatura e em análises quantitativas.
- Comparar as estratégias de escalabilidade vertical e horizontal nos pilares de desempenho, TCO (*Total Cost of Ownership*) e disponibilidade, a partir das evidências coletadas dos estudos mencionados anteriormente e do experimento conduzido.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais necessários para compreender as estratégias de escalabilidade em sistemas distribuídos, com foco em cenários de alta demanda por requisições. São discutidos os princípios de sistemas distribuídos, as definições e características das escalabilidades vertical e horizontal, o papel do Teorema CAP, mecanismos de balanceamento de carga e *autoscaling*, bem como os três pilares de análise adotados neste trabalho: desempenho, custo total de propriedade e disponibilidade.

2.1 SISTEMAS DISTRIBUÍDOS E ESCALABILIDADE

Sistemas distribuídos são compostos por múltiplos componentes computacionais autônomos que se comunicam por meio de redes para cooperar na execução de tarefas comuns, aparentando ao usuário final o funcionamento de um único sistema coerente (GORTON, 2021). Esses ambientes são caracterizados, em geral, por concorrência entre componentes, ausência de um relógio global perfeitamente sincronizado e possibilidade de falhas independentes em diferentes partes do sistema (ATIEWI *et al.*, 2020).

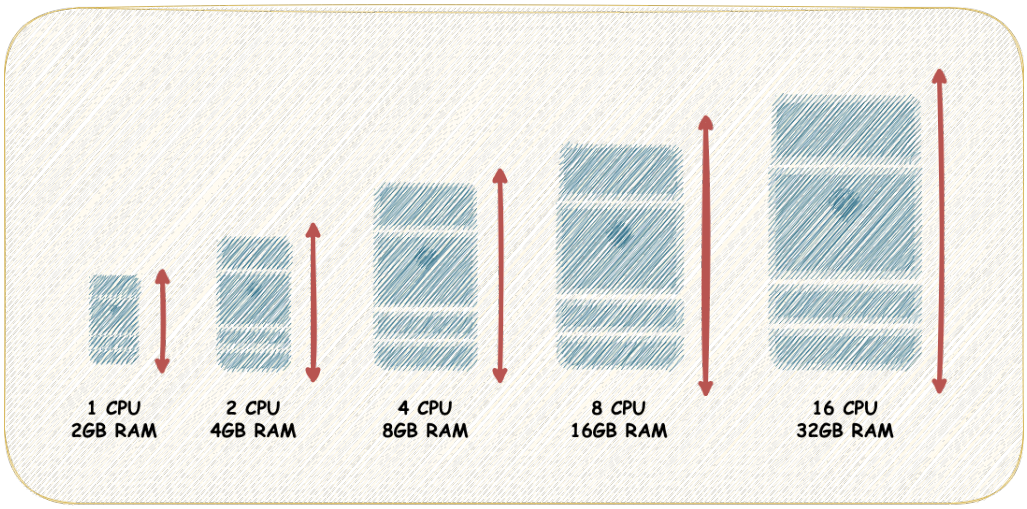
Nesse contexto, escalabilidade pode ser entendida como a capacidade de um sistema manter ou melhorar seus níveis de desempenho à medida que a demanda por recursos aumenta, seja pelo crescimento do volume de dados, do número de usuários ou da quantidade de transações processadas (JOGALEKAR; WOODSIDE, 2000; ARAÚJO, 2025). A literatura costuma destacar três dimensões principais de escalabilidade: de espaço, que diz respeito à capacidade de aumentar recursos para lidar com volumes maiores de dados; de tempo, relacionada à manutenção de um desempenho aceitável conforme o sistema cresce; e a estrutural, que se refere à facilidade de expandir a arquitetura sem provocar um aumento desproporcional de complexidade (GORTON, 2021).

Entre as métricas mais utilizadas para avaliar escalabilidade estão o *throughput*, entendido como o número de requisições processadas por unidade de tempo; a latência, que corresponde ao tempo necessário para responder a cada requisição; e a utilização de recursos, que envolve os percentuais de uso de CPU (*Central Process Unit*), memória, armazenamento e rede (FÉ *et al.*, 2022). Em cenários de alta demanda, também é importante diferenciar escalabilidade de elasticidade: enquanto a primeira se relaciona à capacidade de crescer com eficiência, a elasticidade diz respeito à habilidade de o sistema provisionar e desprovisionar recursos dinamicamente em resposta a variações de carga em tempo quase real (JEONG; KIM; LEE, 2025).

2.2 ESCALABILIDADE VERTICAL (*SCALE-UP*)

A escalabilidade vertical (Figura 1), ou *scale-up*, consiste em aumentar os recursos computacionais de um único servidor para lidar com cargas de trabalho crescentes (GORTON, 2021; ARAÚJO, 2025). Na prática, isso ocorre por meio de *upgrades* de componentes de *hardware*, como processadores, memória RAM (*Random Access Memory*), capacidade de armazenamento e largura de banda de entrada e saída (*input/output* ou I/O).

Figura 1 – Escalabilidade Vertical



Fonte: (FIDELIS, 2024)

2.2.1 Vantagens da Escalabilidade Vertical

A abordagem vertical apresenta vantagens relevantes em determinados contextos, sobretudo quando se trata de aplicações legadas ou sistemas já consolidados. Uma primeira vantagem é a simplicidade de adoção: geralmente não é necessário refatorar o código da aplicação nem alterar significativamente sua arquitetura, o que reduz o risco e o custo inicial de mudanças (LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014). Além disso, o ambiente de operação tende a ser menos complexo, pois não há necessidade de coordenar múltiplos nós nem de empregar mecanismos sofisticados de sincronização de estado ou de descoberta de serviços (CÂNDIDO; SOUSA; SILVA, 2022).

Outro ponto favorável é que a comunicação dentro de um único servidor apresenta, em geral, latência menor do que a comunicação entre máquinas distintas por meio de rede. Isso pode ser particularmente vantajoso em aplicações sensíveis a tempo de resposta, em que pequenas variações de latência impactam diretamente a experiência do usuário ou o cumprimento de SLAs (FÉ *et al.*, 2022).

2.2.2 Limitações e Desafios da Escalabilidade Vertical

Apesar dessas vantagens, a escalabilidade vertical enfrenta limitações estruturais importantes em cenários de alta demanda. A primeira diz respeito aos limites físicos do *hardware*: mesmo servidores de última geração possuem capacidade máxima de CPU, memória e I/O que não pode ser ampliada indefinidamente. Exemplos típicos incluem servidores com até 96 núcleos de CPU e 24 *terabytes* (TB) de memória RAM, patamares que, uma vez alcançados, impedem novo crescimento por simples *upgrade* de componentes (YU; LIU; CHEN, 2023).

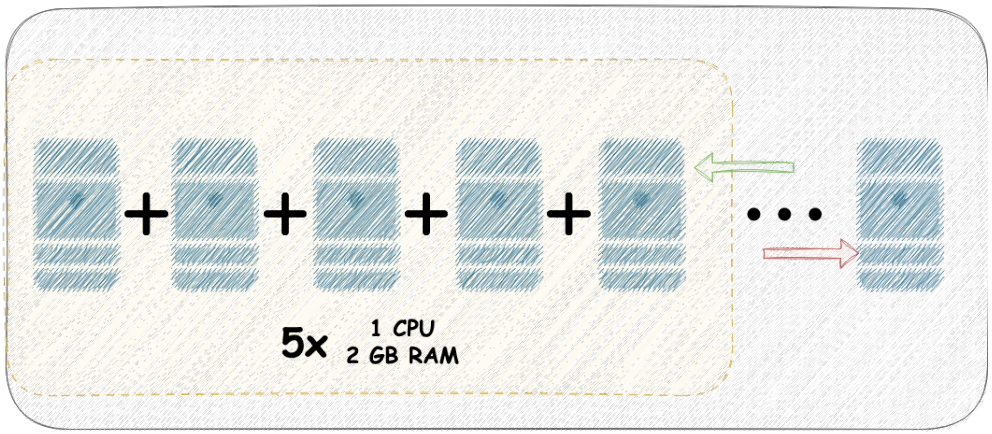
A segunda limitação está relacionada ao padrão de crescimento de custos. Em muitos provedores de nuvem o aumento de capacidade em um único servidor não segue uma escala linear de preços: dobrar a capacidade pode implicar triplicar ou quadruplicar o custo, em função da escassez e da especialização de componentes de alto desempenho (FÉ *et al.*, 2022). Isso torna, em diversos casos, a solução vertical economicamente menos atrativa à medida que se aproxima do topo das configurações disponíveis.

A terceira limitação é o risco de ponto único de falha (*Single Point of Failure* ou SPOF). Quando toda a aplicação depende de um único servidor, uma falha nesse nó implica indisponibilidade total do sistema, situação especialmente grave em períodos de pico de acesso (SOUZA *et al.*, 2019). Além disso, procedimentos de *upgrade* ou manutenção frequentemente exigem janelas de indisponibilidade (*downtime*) programadas, o que dificulta a obtenção de níveis elevados de disponibilidade contínua (ARAÚJO, 2025). Em conjunto, esses fatores fazem com que a escalabilidade vertical atinja, em muitos casos, um “teto“ de capacidade técnica ou econômica.

2.3 ESCALABILIDADE HORIZONTAL (*SCALE-OUT*)

A escalabilidade horizontal (Figura 2), ou *scale-out*, baseia-se na distribuição da carga de processamento entre múltiplos nós computacionais independentes que operam em paralelo (GORTON, 2021; LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014). Em vez de concentrar recursos em um único servidor cada vez mais potente, adicionam-se novas instâncias de aplicação ou novos servidores ao sistema para aumentar a capacidade total.

Figura 2 – Escalabilidade Horizontal



Fonte: (FIDELIS, 2024)

2.3.1 Vantagens da Escalabilidade Horizontal

A principal vantagem da abordagem horizontal é a elasticidade. Em ambientes adequadamente configurados, torna-se possível adicionar ou remover nós em tempo quase real, em resposta a variações de carga, por meio de mecanismos automáticos de *autoscaling* (JEONG; KIM; LEE, 2025; LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014). Isso é especialmente relevante em sistemas sujeitos a picos imprevisíveis, pois é possível absorver variações de tráfego sem degradação acentuada de desempenho.

Além disso, arquiteturas horizontais tendem a ser mais tolerantes a falhas, pois a indisponibilidade de um nó individual não implica, necessariamente, a interrupção total do serviço. Com o apoio de um balanceador de carga, as requisições podem ser redirecionadas automaticamente para instâncias saudáveis, o que reduz o impacto de falhas pontuais (SOUZA *et al.*, 2019).

Outro benefício é o potencial de economia de escala obtido por meio do uso de *hardware commodity*, isto é, servidores de especificação padrão e custo unitário relativamente baixo, cujo acréscimo ao sistema tende a seguir um padrão de custo mais próximo do linear (YU; LIU; CHEN, 2023; CÂNDIDO; SOUSA; SILVA, 2022). Em muitos casos, é mais vantajoso, em termos financeiros, operar um conjunto de máquinas moderadas do que um único servidor extremamente robusto. Adicionalmente, técnicas como *rolling updates* permitem realizar atualizações graduais, reduzindo ou eliminando *downtime* durante janelas de manutenção (ARAÚJO, 2025).

2.3.2 Desafios e Complexidades da Escalabilidade Horizontal

Por outro lado, a escalabilidade horizontal introduz complexidades técnicas e organizacionais consideráveis. Um primeiro desafio é o gerenciamento de consistência de dados distribuídos. Em sistemas com múltiplos nós que podem processar requisições em paralelo e acessar os mesmos dados, é necessário definir estratégias para manter uma visão coerente do estado do sistema, respeitando as restrições estabelecidas pelo Teorema CAP (abordado na seção 2.4) (CÂNDIDO; SOUSA; SILVA, 2022).

Outro ponto crítico é o aumento do tráfego de rede e do *overhead* de comunicação entre nós. Operações que exigem coordenação frequente, como transações distribuídas ou sincronização de estado entre instâncias, podem sofrer com aumento de latência, especialmente quando os nós estão distribuídos em diferentes zonas de disponibilidade ou regiões geográficas (FILHO; MENDONÇA, 2024). Isso demanda atenção ao desenho da arquitetura de dados e, muitas vezes, o uso de técnicas como particionamento (*sharding*), replicação e *caching* distribuído (SOUZA *et al.*, 2019).

A orquestração de múltiplas instâncias também requer ferramentas especializadas, como Kubernetes, Docker Swarm ou Apache Mesos, responsáveis por gerenciar o ciclo de vida de *containers*, descoberta de serviços, balanceamento de carga interno e recuperação automática em caso de falha (LAVIOLA; KREUTZ; MANSILHA, 2025; ROSSI *et al.*, 2020). Tais ferramentas aumentam a complexidade operacional e exigem equipes com maior nível de especialização em sistemas distribuídos e práticas de DevOps.

Por fim, o monitoramento e a depuração tornam-se mais desafiadores em ambientes distribuídos. Em vez de acompanhar um único processo ou servidor, é necessário coletar e correlacionar *logs* (registros de eventos), métricas e rastreamentos distribuídos (*distributed tracing*) provenientes de vários serviços simultaneamente, o que demanda soluções de observabilidade mais robustas (ARAÚJO, 2025).

2.4 TEOREMA CAP E CONSISTÊNCIA EM SISTEMAS DISTRIBUÍDOS

O Teorema CAP, proposto inicialmente por Eric Brewer e posteriormente formalizado por Gilbert e Lynch, estabelece uma limitação fundamental para sistemas de dados distribuídos: em presença de particionamentos de rede, não é possível garantir simultaneamente consistência forte, disponibilidade total e tolerância a partições (GORTON, 2021; GILBERT; LYNCH, 2002). As três propriedades envolvidas são:

- Consistência (*Consistency*): todos os nós veem os mesmos dados ao mesmo tempo; isto é, leituras sucessivas após uma escrita retornam o valor mais recente.
- Disponibilidade (*Availability*): todo pedido recebe uma resposta, com sucesso ou falha, sem garantia de conter necessariamente o dado mais atualizado.

- Tolerância a Partições (*Partition Tolerance*): o sistema continua operando mesmo diante de falhas de comunicação entre nós ou subdivisões da rede.

Na presença de particionamento, inevitável em sistemas reais distribuídos em múltiplos nós, projetistas precisam priorizar quais propriedades serão mantidas. Sistemas classificados como CP (*Consistency + Partition Tolerance*) privilegiam consistência, mesmo que, em alguns cenários, rejeitem requisições ou apresentem indisponibilidade parcial para evitar estados inconsistentes. Já sistemas AP (*Availability + Partition Tolerance*) optam por manter alta disponibilidade, aceitando que dados possam permanecer temporariamente inconsistentes até que um processo de reconciliação seja concluído (SOUZA *et al.*, 2019; CÂNDIDO; SOUSA; SILVA, 2022).

Nesse contexto, o modelo BASE (*Basically Available, Soft state, Eventually consistent*) surge como alternativa ao modelo ACID (*Atomicity, Consistency, Isolation, Durability*) tradicional de bancos de dados transacionais. Enquanto ACID enfatiza consistência forte, BASE aceita consistência eventual em troca de maior disponibilidade e tolerância a falhas, o que se mostra adequado para diversos sistemas web de larga escala, como redes sociais e aplicações de *e-commerce* (GORTON, 2021; GILBERT; LYNCH, 2002). Em contrapartida, domínios como o financeiro e o bancário frequentemente demandam garantias mais próximas de CP, priorizando a exatidão dos dados sobre a disponibilidade em situações de particionamento.

2.5 BALANCEAMENTO DE CARGA E *AUTOSCALING*

O balanceamento de carga desempenha papel central na viabilização da escalabilidade horizontal em sistemas de alta demanda. Seu objetivo é distribuir requisições de entrada entre múltiplos servidores de *backend* de forma a maximizar a utilização eficiente de recursos, reduzir a latência percebida pelo usuário e evitar a sobrecarga de instâncias específicas (LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014).

2.5.1 Arquitetura e Tipos de *Load Balancers*

Os *load balancers* podem atuar em diferentes camadas do modelo OSI (*Open Systems Interconnection*). Em implementações de Camada 4 (camada de transporte), as decisões de roteamento são tomadas com base em informações como endereços IP de origem e destino e portas TCP/UDP, o que garante alta performance, porém com menor flexibilidade de regras (GORTON, 2021). Já os balanceadores de Camada 7 (camada de aplicação) inspecionam o conteúdo das requisições HTTP/HTTPS, o que permite decisões mais sofisticadas, como roteamento por caminho de URL, domínio, *cookies* ou cabeçalhos específicos, ao custo de maior *overhead* de processamento (GORTON, 2021).

2.5.2 Algoritmos de Balanceamento de Carga

Diversos algoritmos podem ser empregados na distribuição de requisições, variando entre abordagens estáticas e dinâmicas. O algoritmo *Round Robin* distribui as requisições de forma sequencial entre os servidores, sendo adequado para ambientes relativamente homogêneos. Em cenários em que há servidores com capacidades distintas, o *Weighted Round Robin* associa pesos a cada instância, de modo que servidores mais potentes recebam maior parcela das requisições (LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014).

Algoritmos dinâmicos, como *Least Connections*, direcionam novas requisições ao servidor com menor número de conexões ativas, se adaptando automaticamente a variações momentâneas de carga. Já o *Least Response Time* leva em conta tanto o número de conexões quanto o tempo de resposta recente dos servidores, o que favorece aqueles com menor latência média (GORTON, 2021; LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014).

2.5.3 *Autoscaling* em Contextos de Alta Demanda

O *autoscaling* refere-se à capacidade de ajustar automaticamente a quantidade de recursos computacionais alocados a um sistema com base em métricas de demanda, como utilização de CPU, memória, taxa de requisições ou filas de mensagens (JEONG; KIM; LEE, 2025). Em abordagens reativas, o sistema monitora essas métricas em tempo real e adiciona ou remove instâncias quando limiares predefinidos são ultrapassados (FÉ *et al.*, 2022). Já em abordagens preditivas, modelos de aprendizado de máquina utilizam dados históricos para antecipar picos de carga e realizar o provisionamento de forma preventiva (JEONG; KIM; LEE, 2025).

Apesar de seus benefícios, o *autoscaling* enfrenta desafios importantes. Um deles é a latência de provisionamento: iniciar novas instâncias pode levar de dezenas de segundos a alguns minutos, período em que o sistema permanece sob pressão de carga elevada (HASSAN, 2015; LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2014). Outro desafio é o problema de *thrashing*, caracterizado por oscilações frequentes entre aumento e redução de capacidade, que podem gerar instabilidade operacional. Além disso, arquiteturas *serverless* e funções como serviço (*Function as a Service* ou FaaS) introduzem o fenômeno de *cold start*, isto é, uma penalidade de latência na primeira requisição atendida por uma nova instância (ALMEIDA; MORAIS, 2022).

2.6 DESEMPENHO, CUSTO E DISPONIBILIDADE

Esta seção apresenta os três pilares utilizados como base para a análise comparativa das estratégias de escalabilidade: desempenho, custo total de propriedade e disponibilidade.

2.6.1 Desempenho

O desempenho em sistemas distribuídos é um conceito multidimensional, que envolve diferentes métricas inter-relacionadas. O *throughput* mede a quantidade de requisições processadas por unidade de tempo e está associado à capacidade total do sistema (FÉ *et al.*, 2022). A latência corresponde ao tempo decorrido entre o envio de uma requisição e o recebimento da resposta, sendo frequentemente analisada por meio de percentis, como P50, P95 e P99, para capturar o comportamento sob diferentes níveis de carga (GORTON, 2021).

A utilização de recursos, por sua vez, avalia o quão intensamente CPU, memória, rede e armazenamento estão sendo explorados. Taxas de utilização persistentemente muito elevadas (próximas ou superiores a 80%) podem indicar risco de saturação e degradação de desempenho (ARAÚJO, 2025). Em cenários de picos extremos, ganha relevância o conceito de elasticidade de desempenho, isto é, a capacidade de manter latências aceitáveis mesmo diante de variações abruptas de carga (FÉ *et al.*, 2022; JEONG; KIM; LEE, 2025).

2.6.2 Custo Total de Propriedade (TCO)

O *Total Cost of Ownership* (TCO) engloba todos os custos associados à operação de uma solução de infraestrutura ao longo de um período determinado, geralmente entre 3 e 5 anos (CÂNDIDO; SOUSA; SILVA, 2022). Esse valor abrange desde despesas de infraestrutura, como aquisição ou aluguel de *hardware*, licenças de *software* e conectividade de rede, até despesas operacionais, incluindo energia elétrica, refrigeração, área física em *datacenters* e equipe técnica. Também incorpora impactos financeiros decorrentes de indisponibilidades, como perdas de receita e danos à imagem da organização (ARAÚJO, 2025).

Em termos de estratégia de escalabilidade, a abordagem vertical tende a apresentar um crescimento de custos não linear: quanto mais próximos do limite máximo de configuração se encontra o servidor, maior o acréscimo de preço para pequenos ganhos adicionais de capacidade (FÉ *et al.*, 2022). Já a escalabilidade horizontal, ao se basear em nós de *hardware commodity*, costuma exibir um padrão de crescimento mais próximo do linear, ainda que introduza investimentos adicionais em orquestração, observabilidade e capacitação da equipe (YU; LIU; CHEN, 2023; CÂNDIDO; SOUSA; SILVA, 2022).

Em sistemas sujeitos a picos de demanda, vale destacar ainda o conceito de custo por *burst*, que é o valor incremental necessário para absorver aumentos temporários e intensos de tráfego. Estratégias horizontais combinadas com *autoscaling* costumam ser vantajosas nesse aspecto, pois permitem pagar apenas pelos recursos efetivamente utilizados durante os períodos de pico, enquanto soluções exclusivamente

verticais frequentemente exigem superprovisionamento permanente para acomodar cenários de pior caso (FÉ *et al.*, 2022; GUPTA *et al.*, 2023).

2.6.3 Disponibilidade e Confiabilidade

A disponibilidade representa o percentual de tempo em que um sistema permanece operacional e acessível aos usuários, sendo comumente expressa como porcentagem de *uptime* (por exemplo, 99,0%, 99,9% ou 99,99%) (GORTON, 2021). Dois indicadores amplamente utilizados para caracterizar confiabilidade são o *Mean Time Between Failures* (MTBF), que mede o tempo médio de operação entre falhas, e o *Mean Time To Repair* (MTTR), que quantifica o tempo médio necessário para restaurar o funcionamento pleno após uma falha (ARAÚJO, 2025).

A relação entre esses parâmetros pode ser expressa pela fórmula de disponibilidade:

$$A = \frac{MTBF}{MTBF + MTTR}$$

em que A representa a disponibilidade do sistema (SOUZA *et al.*, 2019). Essa expressão evidencia que aumentar a disponibilidade requer tanto elevar o MTBF (reduzindo a frequência de falhas) quanto diminuir o MTTR (acelerando a recuperação).

No contexto da escalabilidade, arquiteturas estritamente verticais tendem a ser mais vulneráveis a SPOFs, o que reduz sua resiliência a falhas graves. Arquiteturas horizontais com redundância entre nós contribuem para elevar o MTBF, pois toleram a falha de componentes individuais sem interromper o serviço como um todo (CÂNDIDO; SOUSA; SILVA, 2022). Em ambientes de alta demanda, falhas durante períodos de pico podem gerar impactos financeiros e reputacionais significativamente maiores, reforçando a importância de projetar sistemas com foco explícito em disponibilidade e recuperabilidade (GUPTA *et al.*, 2023; CÂNDIDO; SOUSA; SILVA, 2022).

3 TRABALHOS RELACIONADOS

A comparação entre escalabilidade vertical e horizontal tem sido objeto de diversos estudos na literatura, em especial no contexto de sistemas distribuídos e computação em nuvem. No entanto, muitos estudos abordam apenas métricas específicas ou contextos restritos, deixando de integrar de forma equilibrada aspectos decisivos. Este capítulo apresenta dois trabalhos representativos e destaca como eles motivam a realização desta pesquisa.

3.1 APPUSWAMY ET AL. (2013): *SCALE-UP* VS *SCALE-OUT* PARA HADOOP

Appuswamy et al. (2013) questionam a visão amplamente difundida de que arquiteturas *scale-out*, baseadas em *clusters* de máquinas comuns, seriam sempre superiores a servidores *scale-up* para fluxos de trabalho de *Big Data*. Para isso, os autores comparam empiricamente o desempenho de Hadoop MapReduce, um *framework* de processamento de dados dentro do ecossistema Apache Hadoop, em cenários de *scale-up* e *scale-out*.

O estudo utiliza um conjunto de 11 *jobs* representativos (análise de *logs*, consultas analíticas, aprendizado de máquina, indexação, entre outros) com tamanhos de entrada majoritariamente inferiores a 100 GB (*gigabytes*), refletindo a distribuição real de *workloads* observada em ambientes de produção de grandes provedores. São avaliadas configurações com um único servidor robusto (*scale-up*) e *clusters* com múltiplos nós (*scale-out*), mantendo aproximadamente o mesmo número total de núcleos de CPU em cada comparação. Para garantir justiça na avaliação, o Hadoop é otimizado em ambas as configurações, ajustando parâmetros como concorrência, *heap* de memória e uso de armazenamento local ou em RAM.

Os resultados mostram que o servidor *scale-up* é competitivo em todos os *jobs* e, em muitos casos, supera o *cluster scale-out*. Em particular, a máquina única com 32 núcleos e 256 GB de RAM supera um *cluster* de 8 nós (também com 32 núcleos no total) em 9 dos 11 *jobs*, com diferenças de desempenho significativas. Nos dois casos restantes, a diferença é inferior a 11%. Além disso, o servidor único apresenta melhor relação performance/custo, melhor eficiência energética (performance por watt) e melhor utilização de espaço físico. Os autores identificam um ponto de transição em que o *scale-out* passa a ser mais vantajoso: *workloads* acima de aproximadamente 100–200 GB de dados.

Apesar de suas contribuições, o estudo possui limitações importantes. O foco está restrito ao Hadoop/MapReduce, não avaliando outras tecnologias populares do mesmo segmento ou arquiteturas baseadas em contêineres e Kubernetes. O perfil de

carga é predominantemente *batch* (em lotes) e relativamente estável, sem considerar cenários de picos abruptos ou alta variabilidade de demanda. A disponibilidade e a resiliência não são avaliadas empiricamente, e aspectos como *autoscaling* dinâmico não são explorados. Ainda assim, o trabalho é um marco ao demonstrar que, para muitos cenários reais de *Big Data*, o *scale-up* pode ser mais competitivo do que se supunha.

3.2 JÄÄSKELÄINEN (2019): ARQUITETURAS EM NUVEM E ESCALABILIDADE

Jääskeläinen (2019) investiga o impacto de diferentes arquiteturas de nuvem na performance e na escalabilidade de aplicações web, com foco explícito na comparação entre estratégias de escalabilidade vertical e horizontal em ambientes de computação em nuvem. O estudo é conduzido na plataforma Microsoft Azure, comparando três abordagens: instâncias de máquinas virtuais (IaaS), Cloud Services (PaaS) e Service Fabric Mesh.

A metodologia consiste em implementar a mesma *API REST* simples nas três arquiteturas e submetê-la a testes de carga com uso de uma ferramenta de geração de requisições (como o JMeter), variando o número de usuários simultâneos até atingir patamares de 10 mil ou 12 mil requisições concorrentes. São avaliados cenários de escalabilidade vertical, aumentando o tamanho das máquinas virtuais (número de núcleos de CPU e quantidade de memória RAM), e de escalabilidade horizontal, aumentando o número de instâncias de serviço sob um balanceador de carga.

Os resultados indicam que, em cargas baixas e moderadas, as três arquiteturas apresentam tempos de resposta médios semelhantes e taxas de erro desprezíveis. À medida que a carga aumenta, porém, surgem diferenças. Em geral, as aplicações hospedadas em Cloud Services (PaaS) demonstram comportamento mais estável sob carga crescente, com menores tempos de resposta e maior capacidade antes de atingir taxas significativas de *timeout*, enquanto as implementações baseadas apenas em máquinas virtuais (IaaS) tendem a degradar mais rapidamente e a apresentar maior proporção de erros em níveis elevados de concorrência. As abordagens de escalabilidade horizontal, com múltiplas instâncias sob um balanceador de carga, mostram ganhos de capacidade em relação ao *scale-up* puro, ainda que com retornos decrescentes à medida que mais réplicas são adicionadas.

O estudo, entretanto, se concentra quase exclusivamente em métricas de performance (tempo de resposta, taxa de erros) e em uma visão qualitativa de escalabilidade, sem uma análise sistemática de custo por unidade de desempenho ou de custo total de propriedade ao longo do tempo. A disponibilidade e a resiliência não são avaliadas por meio de injeção de falhas ou cenários de indisponibilidade de instâncias. Além disso, a API utilizada é relativamente simples e não representa workloads intensivos em CPU ou fluxos de entrada e saída, o que limita a generalização para aplicações

de alta complexidade. Por fim, o estudo não contempla mecanismos de *autoscaling* automático, operando com escalabilidade manual em degraus.

3.3 LACUNAS IDENTIFICADAS E DIFERENCIAL DA PESQUISA

A análise dos trabalhos de Appuswamy et al. (2013) e Jääskeläinen (2019) revelou algumas lacunas relevantes:

- Os estudos tratam, em geral, de subconjuntos de métricas, sem integrar de forma explícita e sistemática os três pilares centrais considerados neste trabalho: desempenho, custo total de propriedade e disponibilidade.
- O foco recai sobre contextos específicos (Hadoop para analytics, APIs simples em um único provedor de nuvem), limitando a generalização para sistemas web sob alta demanda com carga altamente variável.
- A disponibilidade e a resiliência são mencionadas de forma teórica, mas não avaliadas empiricamente por meio de cenários de falha, análise de SPOFs ou medições de tempo de recuperação.
- A dimensão de *autoscaling* dinâmico, tanto reativo quanto preditivo, não é abordada, embora desempenhe papel crucial em ambientes de alta demanda e custos sensíveis ao uso de recursos.
- Não há, nesses estudos, uma síntese que auxilie arquitetos e engenheiros a escolher entre escalabilidade vertical e horizontal considerando simultaneamente diferentes perfis de aplicação, padrão de carga e restrições orçamentárias.

O presente trabalho busca preencher esses espaços ao adotar uma abordagem integrada baseada nos três pilares mencionados, direcionando a análise para sistemas com alta variabilidade de carga e requisitos rigorosos de SLA. Além disso, combina revisão sistemática, experimento controlado e análise de estudos recentes, consolidando os achados em um *framework* comparativo e em uma matriz decisória que apoia escolhas arquiteturais entre *scale-up* e *scale-out* em ambientes de alta demanda.

4 METODOLOGIA

Este capítulo apresenta a metodologia adotada para conduzir a pesquisa, detalhando as etapas necessárias para responder ao problema proposto e alcançar os objetivos definidos. O estudo combinou uma revisão bibliográfica sistemática com a realização de um experimento empírico controlado, o que tornou possível analisar de forma prática o impacto das estratégias de escalabilidade vertical e horizontal em um sistema sujeito a alta demanda. A seguir, são descritos os procedimentos adotados em cada etapa da investigação.

4.1 REVISÃO BIBLIOGRÁFICA SISTEMÁTICA

A revisão bibliográfica teve como objetivo identificar, selecionar e analisar estudos que abordam escalabilidade de sistemas, desempenho em ambientes distribuídos e estratégias de acesso sob carga intensa. Para isso, foram consultadas bases reconhecidas, como IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink e Google Scholar.

Os termos de busca incluíram combinações como *vertical scalability*, *horizontal scalability*, *distributed systems performance*, *autoscaling* e *high-demand architectures*. Os critérios de inclusão priorizaram publicações dos últimos cinco anos, artigos revisados por pares e estudos que apresentassem resultados experimentais ou análises comparativas.

A partir desse processo, foi possível identificar tendências, limitações e recomendações presentes na literatura atual, que serviram de base tanto para a definição do experimento quanto para a interpretação dos resultados.

4.2 EXPERIMENTO EMPÍRICO CONTROLADO

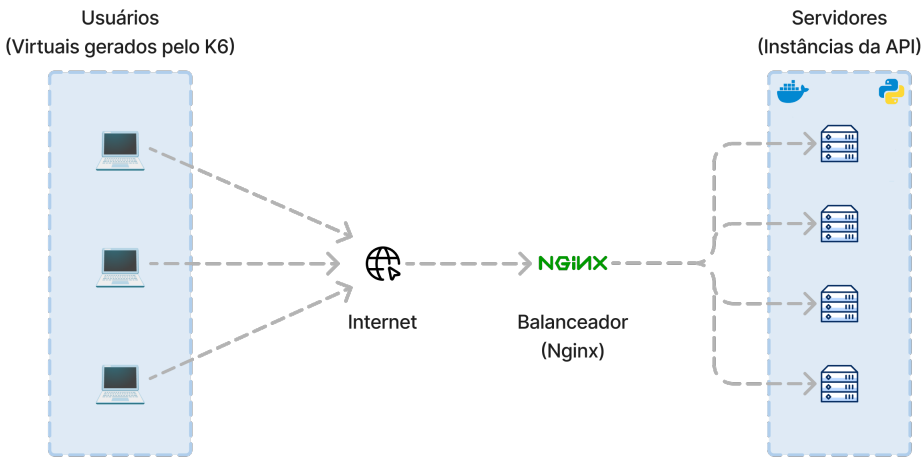
4.2.1 Objetivo do Experimento

O experimento foi desenvolvido para validar empiricamente os *trade-offs* teóricos identificados na revisão bibliográfica, comparando de forma controlada o desempenho, custo-benefício e comportamento sob carga das estratégias de escalabilidade vertical e horizontal.

4.2.2 Arquitetura da Aplicação

Foi desenvolvida uma *API REST* em Flask, expondo um único *endpoint* que simula um processamento de dados com carga computacional intensa a fim de criar um ambiente mais realista. A Figura 3 representa a arquitetura que foi utilizada:

Figura 3 – Arquitetura da Aplicação



Fonte: Elaborado pelo autor

O Nginx foi a ferramenta adotada para realizar o balanceamento de carga nos testes. De modo geral, as tecnologias utilizadas foram:

- Flask: *framework* web minimalista para Python;
- Nginx: *load balancer* reverso;
- Grafana K6: ferramenta de geração de carga sintética;
- Docker: *containerização* e isolamento de recursos;
- Ubuntu 24.04 LTS: sistema operacional do *host*.

4.2.3 Cenários de Teste e Coleta de Métricas

Foram definidos três cenários comparativos:

- Cenário 1: 1 *container* Flask com 0,5 CPU-core e 256MB RAM, representando servidor mínimo viável para pequeno e-commerce, por exemplo.
- Cenário 2: 1 *container* Flask com 2 CPU-cores e 2GB RAM, simulando *upgrade* de *hardware* (*scale-up*).
- Cenário 3: 4 *containers* Flask (0,5 CPU-core, 256MB RAM cada) com Nginx como *load balancer*, simulando scale-out distribuído.

As métricas coletadas incluem *throughput* (requisições por segundo), latência média e percentis (P50, P95, P99).

4.2.4 Ambiente e Procedimento de Execução

O experimento foi conduzido em um ambiente controlado, utilizando um computador pessoal executando o sistema operacional Ubuntu 24.04 LTS. A máquina dispõe de um processador Intel Core i3 com 8 núcleos lógicos e 8 GB (gigabytes) de memória RAM, configuração suficiente para executar simultaneamente os contêineres da aplicação, o balanceador Nginx e a ferramenta de testes de carga. Todas as etapas foram realizadas por meio do terminal do próprio sistema.

A aplicação utilizada no experimento foi containerizada, assim como o serviço Nginx empregado no Cenário 3 para atuar como *load balancer*. Ambos foram orquestrados com Docker Compose, responsável por criar a rede interna dos serviços, gerenciar a inicialização dos contêineres e garantir que todos os cenários fossem executados nas mesmas condições. Nos testes, os limites de CPU e memória foram configurados diretamente no arquivo “docker-compose.yml“ (Figura 4), utilizando a diretiva “resources“ para definir as restrições de *hardware* de cada contêiner, de modo a assegurar que as condições de teste refletissem fielmente os parâmetros estabelecidos no experimento.

Figura 4 – Docker Compose

```yaml
services:
  nginx:
    image: nginx:alpine
    ports:
      - "8080:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - api1
    deploy:
      resources:
        limits:
          cpus: "1"
          memory: 512M

  api1:
    build: ./api
    deploy:
      resources:
        limits:
          cpus: "0.5"
          memory: 256M
```

Fonte: Elaborado pelo autor

Diferentemente da aplicação e do Nginx, o K6 não foi containerizado. A ferramenta foi executada diretamente no sistema hospedeiro, utilizando a rede nativa do próprio *host*. Durante cada execução, o K6 produziu métricas detalhadas, como tempo médio de resposta, latência, taxa de requisições por segundo e erros, registradas diretamente no terminal. Todas essas informações foram extraídas a partir dos *logs* gerados ao final de cada teste. O procedimento adotado para todos os cenários seguiu a mesma sequência:

1. Inicialização da aplicação (e do Nginx, quando aplicável) via Docker Compose;
2. Verificação do estado dos contêineres e aguardo da estabilização dos serviços;
3. Execução do teste de carga com o K6 a partir do *host*;
4. Coleta e armazenamento das métricas geradas pela ferramenta;
5. Repetição do processo de forma idêntica para os demais cenários.

Para garantir reprodutibilidade, todo o código-fonte, arquivos de configuração, e scripts de teste foram disponibilizados em um repositório público, permitindo que outros usuários executem o experimento nas mesmas condições.

4.3 SELEÇÃO DE ESTUDOS DE CASO

Além do experimento próprio, foram selecionados dois estudos acadêmicos de referência para análise comparativa, cada um abordando contextos e perspectivas complementares sobre escalabilidade vertical e horizontal:

- Estudo 1: Investigação de custo-eficiência em estratégias de escalabilidade vertical e horizontal aplicadas a um sistema financeiro de alto *throughput*. O estudo implementa tanto uma arquitetura monolítica quanto uma arquitetura de *microservices*, ambas em Kubernetes, permitindo análise comparativa de desempenho, custo e eficiência de recursos em diferentes cenários de carga (HOLMGREN, 2025).
- Estudo 2: Análise comparativa de *autoscaling* horizontal (HPA ou *Horizontal Pod Autoscaler*) e vertical (VPA ou *Vertical Pod Autoscaler*) no Kubernetes, publicado em 2022, que avalia tempo de resposta, latência, tempo de reação a variações de carga e comportamento de ajuste de recursos sob diferentes perfis de carga (AKSHLLEY; CARVALHO; LOPES, 2022).

A seleção foi fundamentada em critérios de relevância, publicação recente, rigor metodológico e alinhamento com os objetivos deste trabalho.

4.4 *FRAMEWORK* DE ANÁLISE COMPARATIVA

Para estruturar a análise, foi adotado um *framework* baseado em três pilares principais já detalhados no capítulo anterior:

- Desempenho: Avaliação de *throughput*, latência média e percentis, tempo de resposta de *autoscaling* e eficiência de utilização de recursos em cada cenário.
- Custo-benefício: Análise do TCO, incluindo custos de infraestrutura, operacionais e de *downtime*, comparando o custo por requisição em cenários de carga média e de pico.
- Disponibilidade e Resiliência: Avaliação de MTBF, MTTR, tolerância a falhas e capacidade de manter SLAs em cenários de falha de componentes ou picos de demanda.

4.5 LIMITAÇÕES METODOLÓGICAS

É importante reconhecer as limitações deste estudo, uma vez que o experimento foi conduzido em escala controlada, o que pode não refletir integralmente a complexidade e as nuances de sistemas de produção em larga escala. Além disso, como os estudos de caso selecionados abordam contextos distintos, é necessária cautela na generalização dos resultados. Apesar dessas restrições, a combinação do experimento prático com a análise de estudos acadêmicos robustos fornece uma base sólida para as conclusões deste trabalho.

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados da análise comparativa entre estratégias de escalabilidade vertical e horizontal com base em três estudos de caso complementares: um experimento próprio em ambiente controlado e dois estudos acadêmicos com dados experimentais documentados. A combinação desses estudos permitiu aplicar, de forma concreta, o *framework* de análise definido na Metodologia.

5.1 VISÃO GERAL DOS ESTUDOS DE CASO

A Tabela 1 sintetiza os três estudos utilizados na análise comparativa, destacando seus focos e contextos.

Tabela 1 – Estudos Utilizados na Análise Comparativa

Estudo	**Foco Principal**	**Contexto**
Experimento próprio	Comparar *scale-up* vs *scale-out* em API simples	Carga sintética com 200 usuários, três cenários de escalabilidade
Westerlund Holmgren (2025)	Custo-eficiência de escalabilidade vertical e horizontal	Sistema financeiro de corretora, monólito vs *microservices* em Kubernetes
Akshlley et al. (2022)	Desempenho de *autoscaling* horizontal (HPA) vs vertical (VPA)	Aplicação web em Kubernetes sob diferentes perfis de carga

Fonte: Elaborado pelo autor a partir de (HOLMGREN, 2025; AKSHLLEY; CARVALHO; LOPES, 2022).

Os três estudos foram analisados à luz do *framework* proposto no Capítulo 3, de modo a identificar, de forma comparativa, em quais contextos cada estratégia tende a ser mais adequada.

5.2 ESTUDO 1: EXPERIMENTO PRÓPRIO DE ESCALABILIDADE

5.2.1 Contextualização e Motivação

Para complementar a análise bibliográfica e validar empiricamente os *trade-offs* teóricos discutidos, foi desenvolvido um experimento controlado com o objetivo de comparar desempenho e capacidade de processamento sob carga crescente entre as duas principais estratégias de escalabilidade: vertical (*scale-up*) e horizontal (*scale-out*).

5.2.2 Resultados Experimentais

Para simular um ambiente com elevada demanda de requisições, foi utilizado um script K6 configurado para gerar uma carga de 200 usuários virtuais simultâneos durante 30 segundos. Cada cenário foi submetido ao mesmo procedimento cinco vezes, de forma consecutiva, a fim de reduzir variações pontuais e garantir maior consistência e confiabilidade aos resultados obtidos. Os três cenários definidos anteriormente foram avaliados sob as mesmas condições, permitindo uma comparação direta entre as diferentes estratégias de escalabilidade. A Tabela 2 apresenta os resultados médios dos cinco testes em cada cenário.

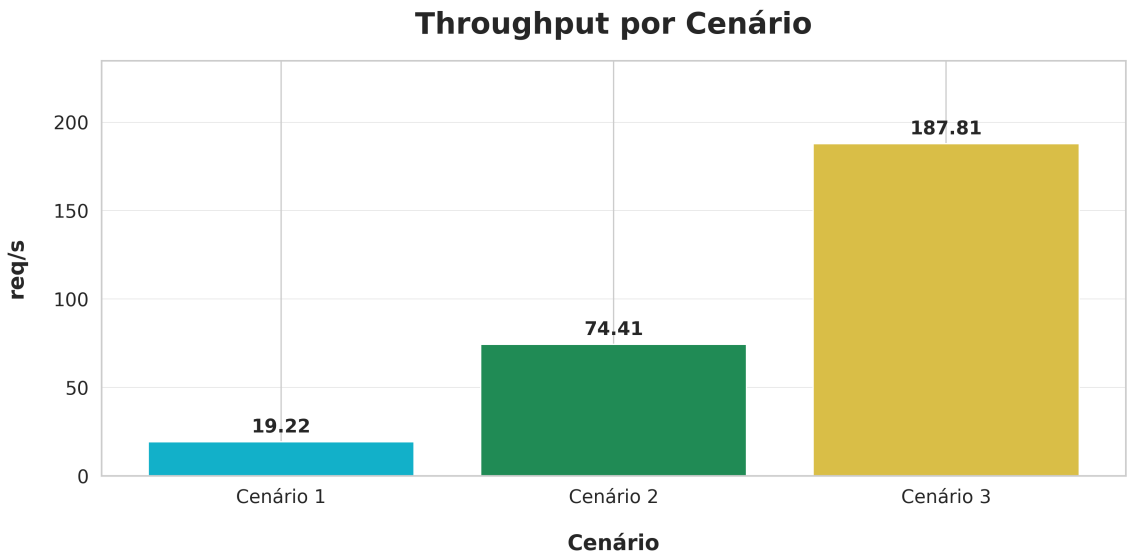
Tabela 2 – Resultados do Experimento de Escalabilidade

Métrica	**Cenário 1 (Base)**	**Cenário 2 (Vertical)**	**Cenário 3 (Horizontal)**
Throughput (req/s)	19,22	74,41	**187,81**
Latência média (ms)	6.670	1.500	**885**
Latência P95 (ms)	9.630	2.170	**1.950**
Total de requisições	1.109	4.166	**6.420**

Fonte: Dados do experimento próprio.

Em termos de *throughput*, o Cenário 3 (horizontal) alcançou 187,81 requisições por segundo, representando aumento de aproximadamente 9,77 vezes em relação ao Cenário 1 (base) e 2,52 vezes em relação ao Cenário 2 (vertical). O ganho supera o aumento linear esperado proporcional ao número de instâncias, o que sugere que a distribuição de carga via Nginx mitigou gargalos presentes nos cenários centralizados. A Figura 5 sintetiza esses resultados em forma de gráfico:

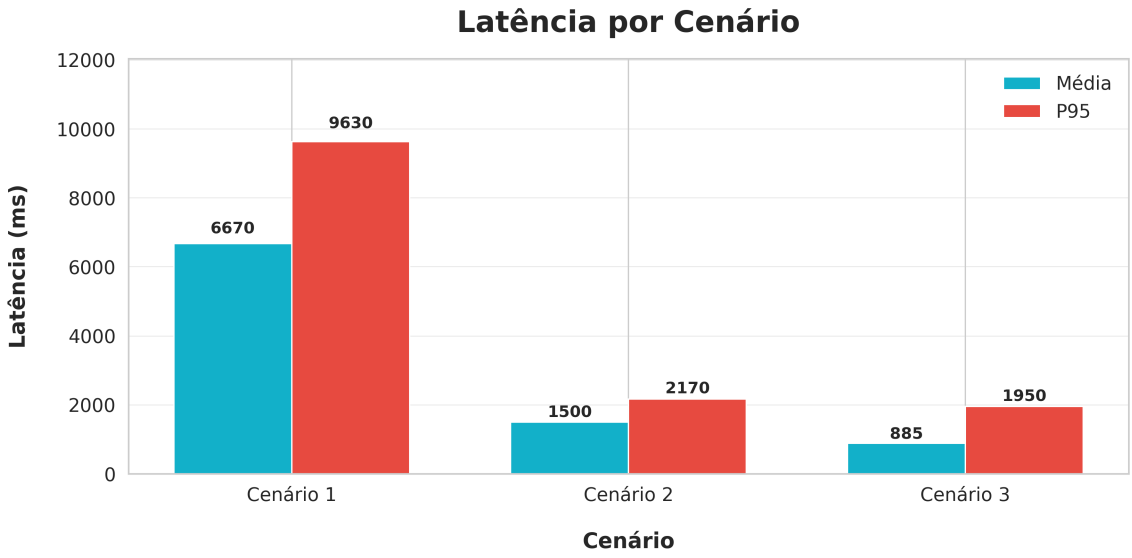
Figura 5 – Análise de Throughput



Fonte: Elaborado pelo autor

Quanto à latência, o Cenário 3 apresentou latência média de 885 ms, contra 6.670 ms no Cenário 1 e 1.500 ms no Cenário 2. A latência P95 seguiu tendência semelhante, alcançando 1.950 ms no Cenário 3, versus 9.630 ms e 2.170 ms nos Cenários 1 e 2, respectivamente, demonstrando que o *scale-out* horizontal não apenas reduz a média, mas também comprime a cauda de distribuição, diminuindo ocorrência de respostas muito lentas. Na Figura 6, é possível visualizar esses resultados:

Figura 6 – Análise de Latência Média e P95



Fonte: Elaborado pelo autor

5.2.3 Discussão: Implicações para Sistemas de Alta Demanda

Os resultados indicam que, mesmo em pequena escala, a escalabilidade horizontal oferece vantagens consistentes para sistemas sob alta demanda. A distribuição da carga entre múltiplas instâncias melhora a capacidade de processamento e reduz a probabilidade de degradação sob picos repentinos.

Além disso, embora o experimento não tenha incluído injeção explícita de falhas, a presença de várias instâncias no cenário horizontal cria uma camada natural de tolerância a falhas, dado que a queda de uma unidade reduz a capacidade, mas não interrompe o serviço por completo. Isso contrasta com a abordagem vertical, que depende inteiramente de um único nó e, portanto, mantém um ponto central de falha.

Assim, mesmo em um ambiente simplificado, o experimento demonstra que o *scale-out* tende a entregar maior previsibilidade operacional e melhor continuidade de serviço em cenários de maior demanda.

5.2.4 Limitações do Experimento

O experimento apresenta algumas limitações importantes que precisam ser consideradas na interpretação dos resultados. Em primeiro lugar, a escala utilizada, limitada a até 200 usuários virtuais, está muito distante dos cenários reais de grandes plataformas, que lidam com milhões de requisições por segundo. Além disso, todos os testes foram realizados em um ambiente controlado de rede local, sem latência de internet, falhas de *hardware* ou oscilações típicas de infraestrutura distribuída, o que reduz

a imprevisibilidade normalmente presente em ambientes de produção. A configuração dos recursos também permaneceu estática durante toda a execução, não permitindo analisar comportamentos relacionados a mecanismos de autoscaling dinâmico. Outro ponto é que a aplicação utilizada no teste é deliberadamente simples, composta por um único endpoint, e não reproduz a complexidade de sistemas reais, que envolvem múltiplos serviços, banco de dados, cache e outras camadas de processamento.

Por fim, é importante destacar que, por ter sido conduzido integralmente em ambiente local, o experimento não permite avaliar impactos financeiros ou custos operacionais associados a cada estratégia de escalabilidade, algo que somente seria possível em uma infraestrutura de nuvem ou ambiente de produção. Ainda assim, os resultados obtidos fornecem um exemplo concreto e replicável dos *trade-offs* discutidos na teoria e funcionam como referência útil para estudos mais complexos e realistas.

5.3 ESTUDO 2: CUSTO E DESEMPENHO EM FLUXOS FINANCEIROS

5.3.1 Contextualização e Metodologia

O segundo estudo analisado é o trabalho de Westerlund Holmgren (2025), que investiga a eficiência de custo de estratégias de escalabilidade vertical e horizontal em arquiteturas de software utilizadas em sistemas financeiros de alta demanda (HOLMGREN, 2025). O autor implementa duas versões de um sistema de corretora de alto *throughput*: uma arquitetura monolítica e uma arquitetura de *microservices*, ambas implantadas em ambiente Kubernetes com recursos equivalentes.

O sistema simulado processa dois tipos principais de fluxo de trabalho: processamento de ordens (fortemente dependente de banco de dados) e atualização de preços (*price updates*) intensiva em CPU. Em ambos os casos, são avaliados:

- Desempenho: *throughput* de ordens e de atualizações de preço, e tempos de resposta;
- Custo: custo por hora de execução em diferentes níveis de escalabilidade;
- Eficiência: relação desempenho/custo em cada configuração.

A escalabilidade vertical é representada pelo aumento de recursos (CPU e memória) em um único *pod* (unidade mínima de implantação e execução) que executa o monólito, enquanto a escalabilidade horizontal é representada pelo acréscimo de instâncias de *microservices* responsáveis por partes do fluxo de trabalho.

5.3.2 Resultados Quantitativos e Econômicos

Os resultados do estudo indicam que:

- A arquitetura monolítica, escalada verticalmente, apresenta maior *throughput* inicial para o fluxo de ordens, especialmente em cenários em que a lógica de negócio está fortemente acoplada ao banco de dados.
- A arquitetura de *microservices*, escalada horizontalmente, escala melhor ao longo do tempo para cenários com múltiplos fluxos de trabalho concorrentes (ordens e atualizações de preços simultâneas), ainda que parta de um *throughput* inicial menor.
- Do ponto de vista de custo por unidade de trabalho, a combinação monólito + *scale-up* vertical mostra-se mais eficiente em sistemas com um único fluxo predominante e forte dependência de banco de dados, enquanto a combinação *microservices* + *scale-out* horizontal torna-se mais vantajosa em cenários com múltiplos fluxos heterogêneos e necessidade de controle granular de recursos (HOLMGREN, 2025).

Em outras palavras, o estudo não aponta um “vencedor absoluto”, mas identifica faixas de operação em que cada abordagem atinge melhor relação desempenho/custo.

5.3.3 Discussão: Quando Vertical se Mostra Vantajosa

O estudo de Westerlund Holmgren reforça que a escalabilidade vertical continua sendo competitiva e, em alguns casos, até preferível, dependendo do contexto e das características da aplicação. Para fluxos de trabalho únicos e fortemente dependentes de operações de banco de dados, o modelo monolítico escalado verticalmente tende a alcançar maior *throughput* e menor latência, principalmente por reduzir a sobrecarga de comunicação que aparece quando a aplicação é fragmentada em múltiplos serviços. Em situações nas quais a carga de trabalho é estável, previsível e bem compreendida, a manutenção de uma arquitetura distribuída pode introduzir custos adicionais, tanto operacionais quanto de infraestrutura, que não se justificam, o que torna a verticalização uma alternativa mais econômica.

O estudo também destaca que, embora a escalabilidade vertical mantenha um ponto único de falha, essa limitação pode ser mitigada em domínios como o financeiro, onde há forte exigência de consistência e controle transacional. Nesses ambientes, a simplicidade arquitetural pode até reduzir certos tipos de problemas operacionais, desde que existam mecanismos robustos de recuperação e replicação em nível de banco de dados. Assim, o trabalho funciona como um contraponto relevante à narrativa de que a escalabilidade horizontal é sempre superior, demonstrando que, dependendo do perfil de carga e do domínio de aplicação, a combinação entre arquitetura monolítica e *scale-up* continua sendo uma escolha técnica plenamente racional.

5.4 ESTUDO 3: AUTOSCALING HORIZONTAL E VERTICAL NO KUBERNETES

5.4.1 Contextualização e Metodologia

O terceiro estudo, de Akshlley, Carvalho e Lopes (2022), analisa comparativamente o desempenho de estratégias de *autoscaling* horizontal e vertical no Kubernetes, por meio de um estudo de caso com uma aplicação web submetida a cargas sintéticas variadas (AKSHLLEY; CARVALHO; LOPES, 2022).

O ambiente experimental utiliza Kubernetes em configuração de *cluster* controlado. Nesse contexto, o *cluster* corresponde ao conjunto de máquinas (físicas ou virtuais) que trabalham de forma integrada para executar a aplicação, enquanto os *pods* são as menores unidades de execução dentro do cluster, cada um contendo um ou mais contêineres. A aplicação é implantada nesses *pods*, que inicialmente recebem recursos idênticos. São avaliadas duas abordagens:

- HPA (*Horizontal Pod Autoscaler*): ajusta a quantidade de réplicas de *pods* com base em métricas como utilização de CPU.
- VPA (*Vertical Pod Autoscaler*): ajusta as requisições de CPU e memória de cada *pod*, aumentando ou reduzindo os recursos de forma vertical.

Os autores definem quatro cenários de carga, incluindo aumentos graduais, picos abruptos e variações frequentes, medindo tempo de resposta, tempo de reação do *autoscaling* e comportamento de *upscale*/*downscale* ao longo do tempo.

5.4.2 Resultados Quantitativos

De forma geral, o estudo mostra que:

- O HPA mantém tempo de resposta próximo ao ideal em praticamente todos os cenários, reagindo mais rapidamente às variações de carga e ajustando o número de réplicas em escala de dezenas de segundos.
- O VPA apresenta atrasos significativos para aumentar os recursos dos *pods*, e, em alguns cenários, falha em reduzir recursos (*downscale*) quando a carga diminui, mantendo recursos superprovisionados (AKSHLLEY; CARVALHO; LOPES, 2022).
- Em cenários com variações frequentes ou picos abruptos, o VPA não acompanha adequadamente a dinâmica de carga, resultando em tempos de resposta mais elevados e menos previsíveis.

Em síntese, nas condições e versões de Kubernetes avaliadas, a escalabilidade horizontal automatizada (HPA) mostra-se mais eficaz que a escalabilidade vertical automatizada (VPA) para manter desempenho estável sob cargas variáveis.

5.4.3 Discussão: Quando Horizontal se Mostra Vantajosa

A análise dos resultados evidenciou que a escalabilidade horizontal apresenta vantagens claras em cenários marcados por variações rápidas e intensas de carga. Do ponto de vista de desempenho, o mecanismo de HPA demonstra capacidade de ajustar o número de *pods* de forma ágil, acompanhando picos e oscilações repentinas e mantendo a latência próxima aos níveis desejados. Já o VPA, por depender de realocação de recursos em instâncias já em execução, tende a reagir de maneira mais lenta, o que pode comprometer a estabilidade durante momentos de alta demanda.

Sob a perspectiva de custo-benefício, o estudo mostra que o VPA frequentemente opera com risco maior de superprovisionamento, pois nem sempre consegue reduzir recursos quando a carga diminui. Em contraste, o HPA reduz automaticamente o número de réplicas conforme a demanda cai, aproximando o consumo de recursos ao uso real e, potencialmente, reduzindo custos operacionais.

No que diz respeito à disponibilidade e resiliência, a escalabilidade horizontal se destaca pela própria natureza de replicação distribuída: múltiplos *pods* executando em paralelo fornecem redundância imediata, de modo que a falha de uma instância não compromete o serviço como um todo. O VPA, por operar com menos réplicas e depender mais fortemente de cada instância individual, apresenta maior fragilidade nesses cenários.

Assim, os achados reforçam que, em ambientes modernos de orquestração como o Kubernetes, a escalabilidade horizontal se mostra particularmente vantajosa para aplicações sujeitas a cargas dinâmicas, proporcionando maior adaptabilidade, melhor aproveitamento de recursos e resiliência superior diante de flutuações de demanda.

5.5 DISCUSSÃO SINTÉTICA: APLICAÇÃO DO *FRAMEWORK* COMPARATIVO

A seguir, é discutido como os três estudos convergem (ou divergem) quando analisados pelos três pilares definidos no Capítulo 3.

5.5.1 Pilar 1: Desempenho

No pilar de desempenho, os três estudos convergem em pontos importantes. O experimento próprio evidencia que a escalabilidade horizontal melhora significativamente a capacidade de processamento e a estabilidade sob carga contínua.

Westerlund Holmgren (2025) acrescenta nuance ao mostrar que arquiteturas monolíticas verticalmente escaladas podem oferecer desempenho superior em fluxos únicos, mas perdem eficiência quando múltiplos fluxos concorrentes são introduzidos.

Já o estudo de Akshlley et al. (2022) demonstra, no contexto de autoscaling, que o HPA acompanha variações de carga de forma mais consistente que o VPA,

preservando o desempenho em cenários dinâmicos.

De modo geral, cargas diversificadas e sujeitas a picos favorecem abordagens horizontais, enquanto fluxos únicos e fortemente acoplados podem se beneficiar da verticalização.

5.5.2 Pilar 2: Custo-benefício

No pilar de custo-benefício, é notório que cada abordagem tende a ser mais vantajosa em contextos distintos. Embora o experimento próprio não permita análise financeira direta por ter sido conduzido em ambiente local, ele indica que a verticalização exigiria maior provisionamento para atingir níveis comparáveis de desempenho.

O estudo de Westerlund Holmgren (2025) mostra que, em sistemas com um único fluxo predominante e carga previsível, a escalabilidade vertical pode ser mais econômica devido à menor complexidade operacional.

Por outro lado, Akshlley et al. (2022) destacam que abordagens horizontais, quando combinadas com *autoscaling* eficiente, aproximam o consumo de recursos da demanda real, o que favorece a otimização de custos em ambientes com variação de carga.

Assim, cargas estáveis tendem a favorecer o *scale-up*, enquanto cargas variáveis e sistemas em crescimento se beneficiam do *scale-out*.

5.5.3 Pilar 3: Disponibilidade e Resiliência

A análise conjunta dos estudos evidencia que estratégias horizontais oferecem base mais robusta para alta disponibilidade, graças à distribuição natural da carga entre instâncias e ao isolamento de falhas. O experimento próprio ilustra esse potencial ao mostrar que a queda de uma instância não interromperia o serviço por completo.

Westerlund Holmgren (2025) observa que arquiteturas distribuídas adicionam complexidade, mas, quando bem implementadas, permitem controlar e isolar falhas com maior precisão.

Já Akshlley et al. (2022) mostram que, em ambientes orquestrados como Kubernetes, o *autoscaling* horizontal reduz riscos de saturação e degradação, reforçando a resiliência operacional.

Embora a verticalização possa atender a níveis adequados de disponibilidade em cenários mais simples, sistemas que demandam continuidade rigorosa tendem a se beneficiar mais da abordagem horizontal.

5.5.4 Matriz Decisória Consolidada

A análise integrada dos estudos de caso e do experimento demonstrou que não existe uma estratégia de escalabilidade universalmente superior. A escolha ideal

depende diretamente do perfil de uso da aplicação e das restrições do projeto. Para sintetizar esses achados e instrumentalizar a decisão arquitetural, a Tabela 3 consolida os cenários onde cada abordagem se mostrou mais vantajosa, cruzando os pilares de desempenho, complexidade e custo.

Tabela 3 – Matriz Decisória: Escalabilidade Vertical vs. Horizontal

Critério	**Escalabilidade Vertical (*Scale-up)***	**Escalabilidade Horizontal (*Scale-out*)**
Padrão de Carga	Previsível e Estável	Variável com Picos Abruptos
Arquitetura	Monolítica / Legada	Microsserviços / Cloud-Native
Complexidade	Baixa (Simples administração)	Alta (Exige orquestração e observabilidade)
Disponibilidade	Ponto Único de Falha (SPOF)	Alta Redundância / Resiliência
Custo (TCO)	Menor em cargas baixas/fixas	Melhor custo-benefício em alta escala

Fonte: Elaborado pelo autor.

5.6 IMPLICAÇÕES PARA A PRÁTICA

Do ponto de vista de profissionais de arquitetura e desenvolvimento de sistemas, os resultados dos três estudos indicam que:

- Decisões de escalabilidade devem ser guiadas por requisitos explícitos de desempenho (latência, *throughput*), custo (TCO em horizonte de 3–5 anos) e disponibilidade (SLAs de *uptime*), e não por preferências genéricas por “moderno” ou “legado”.
- A escalabilidade horizontal se mostra especialmente adequada para sistemas destinados a operar sob alta demanda com picos imprevisíveis, mas exige investimento consistente em observabilidade, automação, testes de resiliência e capacitação da equipe.
- A escalabilidade vertical continua sendo uma estratégia válida para sistemas monolíticos bem entendidos, com demanda relativamente estável e em contextos em que a migração para arquiteturas distribuídas não se paga em termos de custo-benefício.

6 CONCLUSÃO E PERSPECTIVAS FUTURAS

6.1 SÍNTESE E CONCLUSÕES

Este trabalho se propôs a investigar como as estratégias de escalabilidade vertical e horizontal impactam desempenho, custo total de propriedade (TCO) e disponibilidade em sistemas de alta demanda, buscando identificar em quais contextos cada uma é mais apropriada. Por meio de revisão bibliográfica estruturada, análise de dois estudos acadêmicos complementares e um experimento controlado, foi possível chegar a conclusões matizadas que desafiam narrativas simplificadas sobre o tema.

A pesquisa demonstrou que escalabilidade horizontal apresenta superioridade técnica consistente em cenários de alta demanda com picos imprevisíveis e múltiplos fluxos de trabalho. No experimento próprio, o cenário horizontal alcançou *throughput* 9,77 vezes superior ao cenário base e 2,52 vezes superior ao cenário vertical, com latência significativamente mais estável. O estudo de Akshlley et al. (2022) confirmou que mecanismos de *autoscaling* horizontal (HPA) reagem rapidamente a variações de carga, enquanto verticais frequentemente apresentam atrasos consideráveis. Esses achados validam a tese central de que horizontal é superior para picos extremos de requisições.

Entretanto, o estudo de Westerlund Holmgren (2025) revelou nuança crítica: superioridade técnica não traduz necessariamente em vantagem econômica em todos os contextos. Para aplicações com carga previsível, fluxo de trabalho único e forte acoplamento a banco de dados, a escalabilidade vertical em uma arquitetura monolítica pode ser mais custo-efetiva que migrar para *microservices* escalados horizontalmente em Kubernetes. O estudo sugere que a expectativa de pequenas empresas em obter benefícios similares aos de gigantes tecnológicos ao refatorar monolitos frequentemente é ilusória (HOLMGREN, 2025).

Portanto, a conclusão central desta pesquisa é que decisão arquitetural não deve ser ideológica, mas pragmática e contextual. Para aplicações críticas com picos extremos de acesso e requisitos rígidos de disponibilidade, escalabilidade horizontal se destaca como estratégia claramente superior. Para sistemas com carga estável, base limitada de usuários, fluxo de trabalho predominante e requisitos de SLA mais flexíveis, escalabilidade vertical tende a ser uma opção mais vantajosa, por oferecer simplicidade operacional e menor custo total, que compensam suas limitações técnicas.

6.2 CONTRIBUIÇÕES DO TRABALHO

As contribuições deste trabalho podem ser entendidas em duas dimensões complementares. Em termos teóricos, a pesquisa reúne e organiza uma literatura

frequentemente fragmentada sobre escalabilidade, propondo um *framework* coerente. Essa síntese articula avanços recentes em *autoscaling* e arquiteturas distribuídas, oferecendo um modelo estruturado que facilita a avaliação sistemática de diferentes estratégias de escalabilidade.

No âmbito prático, o trabalho disponibiliza um guia orientado a evidências para arquitetos e desenvolvedores, combinando resultados de estudos acadêmicos consolidados com experimentação própria. Essa integração permite formular recomendações objetivas sobre quando priorizar cada abordagem de escalabilidade, quais métricas devem ser monitoradas e quais riscos precisam ser considerados no que diz respeito à disponibilidade e aos custos operacionais.

6.3 RECOMENDAÇÕES PRÁTICAS

Com base nos achados, vale ressaltar as seguintes recomendações para profissionais responsáveis por decisões arquiteturais em sistemas de alta demanda:

1. Definir SLAs explícitos: Estabelecer metas claras de disponibilidade (99,0%, 99,9%, 99,99% ou superiores), latência (P95/P99 alvo) e *throughput* esperado. Esses parâmetros devem orientar diretamente a escolha entre escalabilidade vertical, horizontal ou híbrida, não preferências genéricas ou tendências de mercado.

2. Avaliar TCO em horizonte adequado: Decisões de arquitetura devem considerar não apenas custos de implementação, mas também custos operacionais ao longo de 3 a 5 anos, incluindo infraestrutura, equipe técnica, ferramentas de observabilidade e possíveis janelas de indisponibilidade (HOLMGREN, 2025).

3. Adotar observabilidade como requisito não-negociável: Em ambientes distribuídos, mecanismos de *logging* centralizado, métricas em tempo real e *tracing* distribuído são essenciais. Sem observabilidade adequada, se torna muito mais difícil identificar gargalos, dimensionar corretamente recursos ou reagir a incidentes.

4. Planejar migrações de forma incremental: Ao migrar de soluções verticais para horizontais ou de monólitos para *microservices*, adotar abordagem progressiva, priorizando componentes críticos e reduzindo riscos de grandes refatorações que podem falhar ou ultrapassar orçamentos.

5. Investir em capacitação da equipe: O sucesso de estratégias horizontais depende fortemente da maturidade técnica em sistemas distribuídos, Kubernetes, automação de infraestrutura e práticas de DevOps. Treinamento deve ser considerado investimento essencial, não custo opcional.

6.4 PERSPECTIVAS FUTURAS

A área de escalabilidade em sistemas distribuídos continua evoluindo rapidamente, abrindo novas e promissoras linhas de investigação. Um ponto de destaque são as arquiteturas híbridas, que combinam elementos de escalabilidade vertical e horizontal. Ainda há espaço para entender melhor em quais padrões de carga essas combinações superam as estratégias puras, com o intuito de otimizar o uso de recursos e o custo operacional.

Outra vertente relevante envolve o uso de técnicas preditivas, como aprendizado de máquina, para ajustar recursos de forma proativa, antes que o sistema apresente lentidão perceptível. Esse tipo de abordagem preventiva pode reduzir tempos de resposta e melhorar a experiência do usuário.

Também há lacunas importantes relacionadas aos custos em provedores de nuvem nacionais, uma vez que aspectos como preço, desempenho, requisitos legais e infraestrutura diferem significativamente dos serviços internacionais. Além disso, vale aprofundar a investigação sobre a real capacidade das soluções *serverless* de lidar com picos de demanda intensos sem perda de desempenho, especialmente durante a inicialização das funções.

No campo dos bancos de dados distribuídos, o tema permanece pertinente. Comparar abordagens SQL, NoSQL e NewSQL é essencial para compreender as diferenças em consistência, desempenho e resiliência. Testes que simulem falhas simultâneas em larga escala podem ajudar a medir a robustez dessas soluções e oferecer insights valiosos sobre confiabilidade.

Por fim, perspectivas emergentes, como *edge computing* (voltada para latência ultra-baixa), computação heterogênea (com uso combinado de CPUs, GPUs e TPUs) e a escalabilidade verde (com foco em eficiência energética e redução de pegada de carbono), representam caminhos inovadores. Explorar essas frentes pode ampliar significativamente a compreensão sobre como garantir escalabilidade de forma sustentável e eficiente em contextos cada vez mais complexos e distribuídos globalmente.

6.5 CONSIDERAÇÕES FINAIS

Os resultados desta pesquisa indicam que a escalabilidade horizontal tende a se consolidar como estratégia predominante para sistemas modernos de alta demanda, especialmente aqueles expostos à internet com requisitos rígidos de disponibilidade e tempo de resposta. Entretanto, longe de eliminar a relevância da escalabilidade vertical, os achados reforçam que ambas as abordagens possuem espaços de aplicação bem definidos e legítimos.

Para aplicações críticas sujeitas a grandes variações de carga e com requisitos elevados de disponibilidade, a combinação de arquiteturas horizontais com *autoscaling*,

observabilidade e boas práticas de recuperação de falhas se mostra o caminho mais promissor. Para aplicações menores, com cargas estáveis ou fortemente legadas, a escalabilidade vertical continua sendo opção válida e, frequentemente, mais econômica.

Em um cenário de transformação digital contínua e crescente dependência de serviços online, compreender profundamente os *trade-offs* entre escalabilidade vertical e horizontal deixa de ser um detalhe de implementação e se torna uma competência essencial para profissionais da área. Este trabalho busca contribuir tanto para a formação de novos profissionais quanto para o aprimoramento da prática de arquitetos e desenvolvedores em atividade no mercado.

REFERÊNCIAS

AKSHLLEY, K.; CARVALHO, M.; LOPES, R. Análise de desempenho de estratégias de autoscaling vertical e horizontal: um estudo de caso com o kubernetes. In: SBC. **Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2022)**. 2022. p. 1–14. Acesso em: 18 nov. 2025. Disponível em: https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/21437/21261.

ALMEIDA, M. M. S. C.; MORAIS, F. A case study of proactive auto-scaling for an ecommerce workload. In: **arXiv preprint arXiv:2211.11928**. [s.n.], 2022. Acesso em: 18 nov. 2025. Disponível em: https://arxiv.org/pdf/2211.11928.

ARAÚJO, J. L. d. M. **Pesquisa sistemática de arquiteturas escaláveis para aplicações web**. Tese (Doutorado) — Instituto Federal da Paraíba (IFPB), João Pessoa, PB, March 2025. Acesso em: 18 nov. 2025. Disponível em: https://repositorio.ifpb.edu.br/handle/177683/4380.

ATIEWI, S. *et al.* Scalable and secure big data iot system based on cloud computing. **IEEE Access**, v. 8, p. 137397–137411, 2020. Artigo IEEE sobre escalabilidade em sistemas IoT de alta demanda.

CASALICCHIO, E.; SILVESTRI, L. Mechanisms for sla provisioning in cloud-based service providers. **Computer Networks**, Elsevier, v. 57, n. 3, p. 795–810, 2013. Citado por 115 autores. Mecanismos de provisionamento de SLA em provedores cloud.

CÂNDIDO, A. C.; SOUSA, W. J. d.; SILVA, F. G. d. Potencialidades do desenvolvimento de cloud computing como ferramenta estratégica nas organizações. **Gestão & Produção**, Scielo Brasil, v. 29, p. e5020, 2022. Acesso em: 18 nov. 2025. Disponível em: https://www.scielo.br/j/pci/a/rXjTqsQByRGZp6NQxSr8Wyw/?format=html&lang=pt.

FIDELIS, M. **System Design - Performance, Capacidade e Escalabilidade**. 2024. Acesso em: 18 nov. 2025. Disponível em: https://fidelissauro.dev/performance-capacidade-escalabilidade/.

FILHO, R. C. M.; MENDONÇA, N. C. Impacto de desempenho da granularidade de microsserviços: Uma avaliação com o arcabouço service weaver. In: **Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**. Sociedade Brasileira de Computação (SBC), 2024. p. 644–657. Acesso em: 18 nov. 2025. Disponível em: https://sol.sbc.org.br/index.php/sbrc/article/view/29825/29628.

FÉ, I. *et al.* Performance-cost trade-off in auto-scaling mechanisms for cloud computing. **Sensors**, v. 22, n. 3, p. 1221, 2022. Análise de trade-offs entre performance e custo em autoscaling. Acesso em: 18 nov. 2025. Disponível em: https://www.mdpi.com/1424-8220/22/3/1221.

GILBERT, S.; LYNCH, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. **ACM SIGACT News**, ACM, v. 33, n. 2, p. 51–59, 2002. Prova formal do Teorema CAP apresentada por pesquisadores do MIT. Acesso em: 18 nov. 2025. Disponível em: https://users.ece.cmu.edu/~adrian/731-sp04/readings/GL-cap.pdf.

GORTON, I. **Foundations of Scalable Systems: Designing Distributed Architectures**. 1. ed. Sebastopol, CA: O'Reilly Media, 2021. Livro fundamental sobre arquiteturas escaláveis e sistemas distribuídos.

GUNTHER, N. J. Performance and scalability models for a hypergrowth e-commerce site. In: **Proceedings of the CMG Conference**. [s.n.], 2000. p. 149–163. Estudo de caso clássico sobre escalabilidade em e-commerce. Acesso em: 18 nov. 2025. Disponível em: https://arxiv.org/pdf/cs/0012022.

GUPTA, S. *et al.* Identification of benefits, challenges, and pathways in e-commerce industries: An integrated two-phase decision-making model. **Sustainable Operations and Computers**, Elsevier, v. 4, p. 200–218, 2023. Análise integrada de benefícios e desafios no e-commerce incluindo escalabilidade. Acesso em: 18 nov. 2025. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666412723000156.

HASSAN, A. A. E. **Workload Characterization, Controller Design and Performance Evaluation of Elastic Cloud Systems**. Tese (Doutorado) — Umeå University, Sweden, 2015. Tese sobre autoscaling e caracterização de workloads em nuvem.

HOLMGREN, O. W. **Evaluating Cost Efficiency in Scaling Software Architectures**. Dissertação (Mestrado) — Universidade da Suécia, 2025. Tese investigando eficiência de custos entre escalabilidade vertical e horizontal.

JEONG, B.; KIM, H.; LEE, Y. Autoscaling techniques in cloud-native computing: Taxonomy, survey, and research challenges. **Journal of Cloud Computing**, v. 14, p. 15, 2025. Survey recente sobre técnicas de autoscaling. Acesso em: 18 nov. 2025. Disponível em: https://www.researchgate.net/publication/394007512_Autoscaling_techniques_in_cloud-native_computing_A_comprehensive_survey.

JOGALEKAR, P.; WOODSIDE, M. Evaluating the scalability of distributed systems. In: **IEEE Transactions on Parallel and Distributed Systems**. [s.n.], 2000. v. 11, n. 6, p. 589–603. Trabalho seminal sobre avaliação de escalabilidade. Acesso em: 18 nov. 2025. Disponível em: https://ieeexplore.ieee.org/document/649248/authors#authors.

LAVIOLA, L. F.; KREUTZ, D.; MANSILHA, R. B. Cloud autodroid: Um sistema distribuído escalável para execução de ferramentas de ia generativa. In: **Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - Salão de Ferramentas**. Sociedade Brasileira de Computação (SBC), 2025. p. 129–138. Sistema distribuído autoescalável para execução de IA em larga escala. Acesso em: 18 nov. 2025. Disponível em: https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/35867/35654.

LORIDO-BOTRÁN, T.; MIGUEL-ALONSO, J.; LOZANO, J. A. A review of auto-scaling techniques for elastic applications in cloud environments. **Journal of Grid Computing**, Springer, v. 12, p. 559–592, 2014. Survey abrangente sobre técnicas de autoscaling em ambientes cloud. Acesso em: 18 nov. 2025. Disponível em: https://www.researchgate.net/publication/265611546_A_Review_of_Auto-scaling_Techniques_for_Elastic_Applications_in_Cloud_Environments.

ROSSI, F. *et al.* Hierarchical scaling of microservices in kubernetes. In: **Proceedings of the 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)**. [s.n.], 2020. p. 116–125. Políticas hierárquicas

de escalabilidade para microsserviços em Kubernetes. Acesso em: 18 nov. 2025. Disponível em: https://ieeexplore.ieee.org/document/9196461.

SOUZA, J. F. d. *et al.* Escalabilidade de aplicações bag-of-tasks em plataformas heterogêneas. In: **Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**. Porto Alegre, RS: Sociedade Brasileira de Computação (SBC), 2019. p. 664–677. Estudo sobre escalabilidade em sistemas distribuídos heterogêneos. Acesso em: 18 nov. 2025. Disponível em: https://sol.sbc.org.br/index.php/sbrc/article/view/7394/7278.

YU, R.; LIU, X.; CHEN, W. Flexible, highly scalable and cost-effective network architectures and services. **Journal of Network and Computer Applications**, v. 213, p. 103583, 2023. Análise de arquiteturas escaláveis e custos.