# Arquitetura de Dimensionamento Adaptativo com Suporte ao Aprendizado

**André Luiz de J. Gonçalves[1], Leandro A. Freitas[2], Antonio Oliveira-Jr[1,3]**

[1]Instituto de Informática – Universidade Federal de Goiás (UFG),

[2]Instituto Federal de Goias (IFG), GO, Brasil

[3]Fraunhofer Portugal AICOS, Porto, Portugal

andre.goncalves@discente.ufg.br,

leandro.freitas@ifg.edu.br, antoniojr@ufg.br

***Abstract.*** *Meeting performance and stability demands in latency-sensitive applications is one of today's major technological challenges. This work presents an Adaptive Scaling Architecture with Learning Support, applied to the context of immersive applications and based on the combination of hardware metrics and application-level events to optimize resource allocation. The implementation uses Kubernetes and the Kubernetes Event-driven Autoscaler (KEDA), with the Hubs VR application as a case study. Experiments were conducted resulting in the construction of two structured datasets: one based solely on hardware metrics and another also integrating application events. These datasets represent a relevant outcome of the research, serving as a foundation for analyses and the development of predictive strategies. The results indicate that combining metrics can lead to more agile and stable responses to load variations, contributing to the advancement of adaptive solutions in dynamic environments.*

***Resumo.*** *Atender às demandas de desempenho e estabilidade em aplicações sensíveis à latência é um dos principais desafios tecnológicos atuais. Este trabalho apresenta uma Arquitetura de Dimensionamento Adaptativo com Suporte ao Aprendizado, aplicada ao contexto de aplicações imersivas, baseada na combinação de métricas de hardware e eventos da aplicação para otimizar a alocação de recursos. A implementação utiliza Kubernetes (K8s) e o Kubernetes Event-driven Autoscaler (KEDA), tendo a aplicação Hubs VR como estudo de caso. Foram conduzidos experimentos que resultaram na construção de conjuntos de dados estruturados distintos: um baseado apenas em métricas de hardware e outro que integra também eventos da aplicação. Esses datasets representam um produto relevante da pesquisa, servindo como base para análises e desenvolvimento de estratégias preditivas. Os resultados indicam que a combinação de métricas pode promover respostas mais ágeis e estáveis frente às variações de carga, contribuindo para o avanço de soluções adaptativas em ambientes dinâmicos.*

## 1. Introdução

Com o avanço das tecnologias pós-5G (*Beyond 5G - B5G*), as aplicações sensíveis à latência se consolidam como um dos maiores desafios tecnológicos

contemporâneos [Mehta et al. 2023, Han et al. 2022]. Essas aplicações abrangem desde plataformas de *streaming* em tempo real e sistemas industriais críticos até soluções imersivas, como aplicações de Realidade Virtual (*Virtual Reality* - VR), Realidade Aumentada (*Augmented Reality* - AR) e Realidade Mista (*Mixed Reality* - MR) [Pelle et al. 2019], que convergem para o termo Realidade Estendida (*Extended Reality* - XR). Para atender aos requisitos de baixa latência e alta responsividade exigidos por essas tecnologias, especialmente em ambientes distribuídos e baseados em nuvem, são necessárias estratégias eficazes e inovadoras [Han et al. 2022].

As soluções de XR são particularmente relevantes para a análise das exigências de aplicações sensíveis à latência, uma vez que a integração sincronizada entre o ambiente físico e os elementos virtuais é indispensável para proporcionar experiências imersivas com percepção contínua e responsiva aos usuários [Han et al. 2022]. Contudo, desafios relacionados à transmissão de dados, atraso, qualidade gráfica e desconforto físico tornam-se ainda mais críticos em cenários de carga variável e picos de demanda [Wu et al. 2023].

O avanço das comunicações imersivas reforça o impacto dessas aplicações no paradigma atual das telecomunicações. Tecnologias emergentes, como redes de alta velocidade, mostram-se promissoras para enfrentar tais desafios ao oferecer suporte para baixa latência e maior confiabilidade [Siriwardhana et al. 2021]. Em particular, o uso de arquiteturas modernas baseadas em orquestração de serviços, como as fornecidas pelo Kubernetes (K8s), viabiliza a adaptação automática de recursos frente às variações dinâmicas de demanda [Santos et al. 2023].

A evolução das infraestruturas em nuvem tem permitido a orquestração eficiente de serviços distribuídos, embora a crescente complexidade exija soluções avançadas de automação para garantir desempenho e disponibilidade [Nguyen et al. 2020, Taleb et al. 2022]. Nesse cenário, o K8s destaca-se como plataforma amplamente adotada na gestão de cargas em contêineres, oferecendo escalabilidade e resiliência [Kubernetes 2022]. Entre suas funcionalidades, o *Horizontal Pod Autoscaler* (HPA) ajusta dinamicamente o número de pods com base em métricas como uso de CPU e memória, permitindo respostas mais rápidas a variações de carga [Nguyen et al. 2020]. Ferramentas como o *Prometheus* [Prometheus 2024] também contribuem para o monitoramento e a definição de gatilhos de escalonamento.

Para ampliar a eficiência desses mecanismos, o *Kubernetes-based Event Driven Autoscaler* (KEDA) estende o HPA ao incorporar eventos da aplicação como acionadores de escalonamento, permitindo decisões mais alinhadas às necessidades operacionais [KEDA Project 2024]. Essa abordagem tem se mostrado especialmente relevante em aplicações sensíveis à latência, como as de XR, que requerem alta disponibilidade e respostas em tempo real [Gonçalves et al. 2023].

Grande parte das soluções atuais, no entanto, baseia-se exclusivamente em métricas de hardware, o que pode comprometer a precisão do dimensionamento em contextos com alta variabilidade de carga. Ao desconsiderar a dinâmica comportamental das aplicações, tais abordagens tendem a resultar em ajustes menos eficazes e, por consequência, impactar negativamente a qualidade do serviço (*Quality of Service* - QoS). A integração de métricas de hardware com eventos da aplicação surge, assim, como alternativa promissora para

um dimensionamento mais contextualizado e adaptativo às exigências dos sistemas modernos [Dogani et al. 2023].

Diversos estudos têm explorado estratégias de dimensionamento em ambientes distribuídos, baseando-se em métricas de infraestrutura, técnicas de previsão de carga e métodos de aprendizado de máquina. Neste contexto, este trabalho apresenta uma arquitetura adaptativa com suporte ao aprendizado, que integra métricas de hardware e eventos da aplicação como base para o escalonamento de recursos em aplicações imersivas. No escopo desta pesquisa, vêm sendo conduzidas avaliações experimentais com a aplicação *Hubs* VR, focadas na análise da escalabilidade de microsserviços sob diferentes condições de carga. Tais experimentos buscam examinar em que medida a correlação entre métricas de monitoramento e indicadores comportamentais da aplicação pode subsidiar decisões mais eficientes no ajuste dinâmico de recursos.

Entre os resultados obtidos até o momento, destaca-se a constituição de dois conjuntos de dados distintos e estruturados: um baseado exclusivamente em métricas de hardware e outro que incorpora eventos da aplicação à análise. Tais conjuntos subsidiaram experimentos voltados à escalabilidade de réplicas dos microsserviços, considerando a combinação de métricas como parâmetro-chave para o dimensionamento adaptativo. A abordagem mostrou vantagens frente a métodos baseados apenas em hardware, com agilidade, estabilidade e potencial para ambientes sensíveis à latência.

As principais contribuições deste artigo incluem: (i) a proposição de uma arquitetura de dimensionamento adaptativo com suporte ao aprendizado voltada a aplicações imersivas; (ii) a construção e disponibilização de dois conjuntos de dados experimentais que integram métricas de hardware e eventos da aplicação, juntamente com o código-fonte, experimentos e demais artefatos da pesquisa; (iii) a aplicação de técnicas de Estatística Descritiva e Exploratória na análise de desempenho do sistema em diferentes cenários de carga, assegurando transparência, reprodutibilidade e replicabilidade dos resultados[1].

Para além desta seção introdutória, este artigo está estruturado da seguinte forma: inicialmente, discute-se a revisão dos trabalhos relacionados na Seção 2. Na Seção 3, descreve-se a arquitetura proposta para o dimensionamento adaptativo inteligente de aplicações imersivas. A Seção 4 aborda a metodologia e os procedimentos da avaliação experimental, cujos resultados são analisados e discutidos na Seção 5. Por fim, na Seção 6, apresentam-se as conclusões e perspectivas futuras do trabalho.

## 2. Trabalhos Relacionados

O dimensionamento automático de aplicações distribuídas tem sido objeto de estudo em diferentes contextos, com abordagens que vão desde métodos reativos baseados em métricas de infraestrutura até modelos preditivos apoiados por aprendizado de máquina [Dogani et al. 2023]. Nesta seção, são analisados trabalhos relevantes da literatura, com base em critérios como: Dimensionamento Automático de Aplicações (DAA), Monitoramento no Dimensionamento Automático (MDA), Orquestração de Contêineres (OC), Análise de Dados Históricos (ADH), Dimensionamento Baseado em Eventos da Aplicação (DBE), suporte a Aplicações Sensíveis à Latência (DSL), Abordagens

[1]Repositório disponível em: `https://github.com/LABORA-INF-UFG/AdaptScale`

Adaptativas ao Dimensionamento (AAD) e Correlação entre Métricas de Hardware e Aplicação (CMHA). A Tabela 1 apresenta a síntese dos trabalhos analisados.

**Tabela 1. Trabalhos Relacionados**

| Trabalhos Relacionados | DAA | MDA | OC | ADH | DBE | DSL | AAD | CMHA |
|---|---|---|---|---|---|---|---|---|
| [Yuan and Liao 2024] | ✓ | ✓ | ✓ | ✓ | | | ✓ | |
| [Quattrocchi et al. 2024] | ✓ | ✓ | ✓ | ✓ | | | | |
| [Cheng et al. 2023] | ✓ | ✓ | | ✓ | ✓ | | | |
| [Dimolitsas et al. 2023] | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| [Bartolomeo et al. 2023] | ✓ | ✓ | | | | ✓ | | |
| [Benmerar et al. 2023] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Nossa proposta | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Adotando modelos de séries temporais, YUAN e LIAO [Yuan and Liao 2024] estruturam uma solução preditiva para o escalonamento em *clusters* K8s, utilizando algoritmos como Holt-Winters e Gated Recurrent Unit GRU. Embora a abordagem apresente bom desempenho preditivo, ela se baseia exclusivamente em dados históricos, sem incorporar métricas em tempo real ou eventos da aplicação, o que limita sua resposta a comportamentos emergentes, sobretudo em aplicações sensíveis à latência, como as de realidade virtual.

No trabalho de QUATTROCCHI et al. [Quattrocchi et al. 2024], os sistemas ScaleX e QN-CTRL integram técnicas de controle e previsão estatística para otimização de recursos. A abordagem demonstra potencial para garantir estabilidade e eficiência em ambientes de nuvem, porém sua ênfase no dimensionamento vertical pode restringir a escalabilidade em cenários de alta variabilidade. Além da falta de mecanismos adaptativos que limita a capacidade de ajuste dinâmico conforme as flutuações da carga de trabalho.

Com o *framework* ProScale, CHENG et al. [Cheng et al. 2023] propõem uma solução proativa baseada em média móvel simples (*Simple Moving Average* - SMA) para microserviços em ambientes de borda. Contudo, a dependência exclusiva do SMA como modelo preditivo restringe a capacidade de adaptação do sistema, uma vez que este método pressupõe padrões de carga relativamente estáveis. Esta limitação, combinada à ausência de estratégias adaptativas de dimensionamento, pode comprometer o desempenho em ambientes sujeitos a demandas imprevisíveis.

No contexto de *clusters* heterogêneos, DIMOLITSAS et al. [Dimolitsas et al. 2023] propõem um modelo hierárquico de escalonamento para múltiplas aplicações, combinando critérios de QoS e eficiência energética por meio do método *Analytic Hierarchy Process* (AHP). No entanto, o modelo adota uma lógica de alocação agregada, ou seja, distribui os recursos com base em metas globais de desempenho, sem considerar em profundidade o comportamento individual de cada aplicação frente às flutuações de carga. Essa ausência de correlação direta entre padrões de uso e decisões de escalonamento pode limitar sua adaptabilidade em cenários com alta variabilidade.

Com foco na orquestração de aplicações imersivas, BARTOLOMEO et al. [Bartolomeo et al. 2023] investigam uma arquitetura distribuída em ambientes de *edge computing*, utilizando o *framework Oakestra*. Embora o estudo traga avanços na escalabilidade de soluções em Realidade Aumentada (AR), a abordagem ainda depende de regras estáticas de dimensionamento, sem contemplar adequadamente variações súbitas

de carga. Além disso, a alocação de recursos fundamenta-se exclusivamente em métricas de hardware, desconsiderando indicadores diretamente relacionados ao desempenho da aplicação.

BENMERAR et al. [Benmerar et al. 2023], por sua vez, propõem o *Application Management Framework* (AMF), que aplica técnicas de Inteligência Artificial à orquestração de ambientes distribuídos voltados à Realidade Virtual (VR) e AR. O *framework* apresenta mecanismos de adaptação para garantir QoS em cenários altamente distribuídos, mas foca na gestão multi-domínio e no ciclo de vida das aplicações, sem incorporar um processo contínuo de escalonamento automático sensível às variações de carga.

Diferenciando-se dessas propostas, este trabalho propõe uma arquitetura modular orientada ao modelo MAPE-K, que integra métricas de hardware e eventos da aplicação para compor decisões de escalonamento mais contextuais e responsivas. A contribuição destaca-se não apenas pela combinação dessas métricas, mas também pela estruturação de conjuntos de dados experimentais, oferecendo subsídios para análises futuras com modelos de aprendizado de máquina. Além disso, a proposta aborda a evolução da arquitetura com base no conhecimento extraído do sistema, atuando tanto no nível do serviço quanto da plataforma, aspecto ainda incipiente nos trabalhos anteriores.

## 3. Arquitetura de Dimensionamento Adaptativo com Suporte ao Aprendizado

O dimensionamento adaptativo com suporte ao aprendizado baseia-se em uma proposta de arquitetura que adota o modelo MAPE-K (*Monitor, Analyze, Plan, Execute, Knowledge*) [Malburg et al. 2023]. Essa abordagem foi projetada para oferecer ajustes proativos e reativos em ambientes de microsserviços, ampliando a responsividade e a estabilidade do sistema mesmo sob alta variabilidade de carga.

Nesse contexto, a arquitetura proposta integra métricas de hardware e eventos da aplicação como estratégia para aprimorar a eficiência do dimensionamento adaptativo, essencial para aplicações imersivas com baixa latência e controle de recursos. A Figura 1 apresenta os fluxos operacionais e os componentes integrados responsáveis pelo monitoramento, análise e ajuste dinâmico da capacidade computacional.

A arquitetura foi implementada em ambiente controlado, de modo a possibilitar sua experimentação em cenários de carga variável. Os detalhes da infraestrutura e da configuração dos testes são apresentados na Seção 4.

No fluxo operacional, a aplicação *Hubs* atua como núcleo das interações com os usuários e do gerenciamento das sessões virtuais. As requisições simuladas pelo *K6* [k6.io 2024] chegam ao *HAProxy*, responsável pelo balanceamento de carga entre os microsserviços que compõem a aplicação. Esses microsserviços desempenham funções específicas e complementares. O *Reticulum* gerencia conexões WebSocket e roteamento de mensagens entre clientes e servidores, enquanto o *Dialog* viabiliza a comunicação *peer-to-peer* utilizando *Web Real-Time Communications* (WebRTC). O *Coturn* atua como servidor *Traversal Using Relays around NAT* (TURN), retransmitindo dados entre clientes para garantir a estabilidade das conexões WebRTC, mesmo em redes restritivas com NATs e *firewalls*. O *Nearspark* realiza cálculos para otimização de latência e o *Photomnemonic* processa e gerencia mídias, incluindo armazenamento e recuperação de imagens e vídeos.

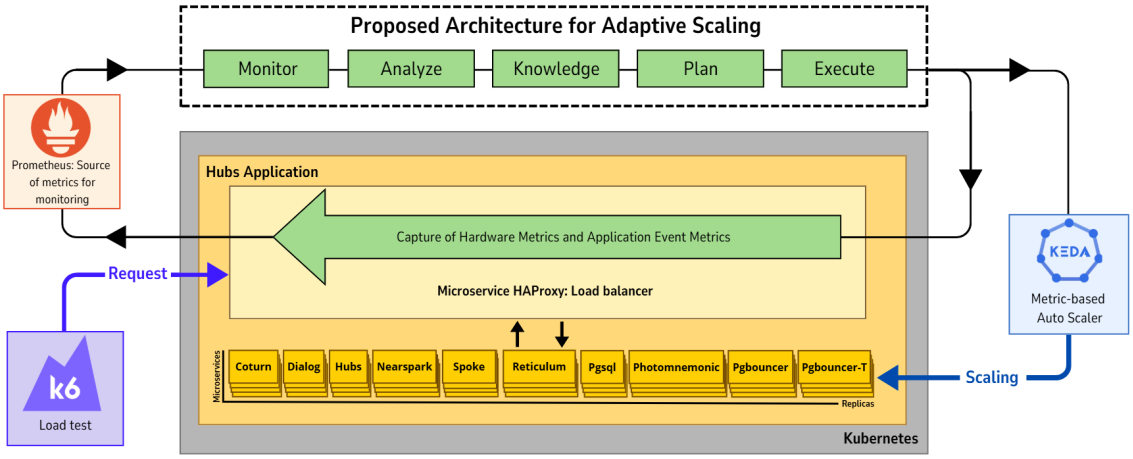


**Figura 1. Arquitetura para o Dimensionamento Adaptativo de Aplicações**

O microsserviço *Hubs*, homônimo da aplicação, centraliza a lógica das sessões virtuais, integrando a interação entre os usuários e a sincronização das atividades no ambiente. Por fim, o *Spoke* cria ambientes 3D e permite a personalização dos espaços virtuais. O banco de dados utilizado é o PgSQL, com suporte dos serviços *PgBouncer* e *PgBouncer-T* para *pooling* de conexões [Hubs Foundation 2024].

A Figura 1 ilustra como, no contexto da arquitetura proposta, o módulo *Monitor* coleta e organiza métricas por meio do *Prometheus*, padronizando e sincronizando dados como uso de CPU, volume de dados transmitidos, taxa de requisições por segundo e número de conexões simultâneas. O módulo *Analyze* processa essas informações por meio de consultas e análises gráficas com o objetivo de identificar padrões sob diferentes condições de carga e detectar variações relevantes no desempenho do sistema. As métricas de hardware, como uso de CPU e tráfego de rede, complementam-se aos eventos da aplicação, como taxa de requisições e conexões simultâneas, oferecendo múltiplas perspectivas sobre o comportamento dinâmico do ambiente monitorado. Os dados históricos alimentam o repositório *Knowledge*, que oferece suporte ao módulo *Plan* para definição de estratégias de escalonamento, incluindo a parametrização de *thresholds* com base nas informações previamente analisadas. O módulo *Execute*, por sua vez, aplica os ajustes de recursos de forma alinhada às variações de demanda observadas.

Embora a arquitetura integre ferramentas amplamente utilizadas no contexto de orquestração de contêineres, sua contribuição reside na forma como esses componentes são articulados segundo o ciclo MAPE-K. A proposta combina métricas de infraestrutura e eventos da aplicação para decisões adaptativas de escalonamento em aplicações imersivas, uma configuração que, segundo a revisão realizada, não é explorada em propostas anteriores.

No âmbito do KEDA, o processo é acionado a partir da configuração dos *ScaledObjects*, que especificam as métricas monitoradas e seus respectivos valores de referência. O KEDA atua como intermediário entre o *Prometheus* e o K8s, convertendo os valores das métricas em sinais que instruem o ajuste automático de réplicas. Quando os limites são ultrapassados, o KEDA aciona o controlador do K8s para ajustar instâncias, proporcionando uma resposta mais ágil e adequada à carga do sistema.

Com isso, a arquitetura estabelece um fluxo integrado de monitoramento, análise e

ajuste dinâmico de recursos, direcionado a aplicações imersivas com requisitos de baixa latência e alta estabilidade. No escopo desta pesquisa, a implementação desse modelo tem possibilitado a construção de um ambiente controlado para experimentação e coleta estruturada de dados. A próxima seção detalha os procedimentos adotados na avaliação experimental, com foco na geração e organização dos conjuntos de dados utilizados nas análises exploratórias.

A arquitetura proposta encontra-se em desenvolvimento progressivo, com os módulos *Monitor* e *Analyze* consolidados, e o módulo *Plan* em estruturação inicial. Já os módulos *Execute* e *Knowledge* foram projetados conceitualmente para compor versões futuras da arquitetura. O módulo *Monitor* utiliza *scripts* que padronizam e sincronizam temporalmente as métricas coletadas pelo Prometheus, garantindo consistência para análise. O módulo *Analyze* aplica rotinas estatísticas com o *Pandas* para identificar padrões sob diferentes cargas. Em sua versão atual, o módulo *Plan* opera com *thresholds* definidos a partir de análise gráfica e observações empíricas, conforme a Seção 4, e será futuramente aprimorado com modelos preditivos. A expectativa é que, com o amadurecimento dos módulos restantes, o ciclo MAPE-K seja operacionalizado de forma contínua. O módulo *Execute* aplicará os ajustes via KEDA e Kubernetes, enquanto o *Knowledge* organizará o conhecimento extraído do sistema. A arquitetura foi concebida com estrutura modular e escalável, permitindo a integração incremental de novas funcionalidades, como a previsão de carga baseada em aprendizado de máquina. Essa composição amplia a adaptabilidade da solução e sua resposta a variações dinâmicas em aplicações imersivas.

## 4. Avaliação Experimental da Arquitetura Proposta

O processo experimental foi estruturado para analisar o comportamento da arquitetura proposta em dois cenários distintos, voltados ao dimensionamento de réplicas dos microsserviços da aplicação: um baseado exclusivamente em métricas de hardware, e outro que combina métricas de hardware e eventos da aplicação. Ambos os cenários foram projetados para observar aspectos relacionados à estabilidade, previsibilidade e eficiência do sistema em condições de carga variável.

Para viabilizar os experimentos, a infraestrutura foi provisionada na plataforma *Digital Ocean*, utilizando um *cluster* K8s configurado para suportar cenários de carga variável e testes controlados. O *cluster* foi estruturado em dois *node pools*: o primeiro, com uma instância de 2 vCPUs e 4 GB de memória RAM; e o segundo, com duas instâncias de 1 vCPU e 2 GB de memória RAM cada. A configuração favorece a resiliência e distribuição da carga.

As requisições simuladas pelos usuários virtuais seguiram um mesmo padrão, envolvendo interações com a aplicação *Hubs*, como o acesso a salas virtuais e a manutenção da sessão ativa. A injeção de carga foi conduzida com a ferramenta *K6*, alternando entre uma carga regular de 55 a 65 usuários virtuais e uma sobrecarga que atingiu até 700 conexões simultâneas. Ao longo de duas horas de teste, foram simulados dois eventos de sobrecarga aplicados de forma progressiva, permitindo a análise do comportamento do sistema em relação à resiliência, tempos de resposta e capacidade de ajuste dinâmico. Para representar, ainda que de forma controlada, a imprevisibilidade típica das cargas de trabalho em aplicações reais, essa variação buscou simular cenários de oscilação de demanda.

Com base na configuração proposta, os testes foram organizados em dois cenários. No Cenário 1, foram utilizadas métricas de hardware, como uso de CPU e transmissão de dados, para orientar o dimensionamento. Essa configuração buscou observar o comportamento do sistema com base em indicadores exclusivamente reativos. No Cenário 2, as métricas combinadas incorporaram, além dos indicadores de hardware, a taxa de requisições por segundo como métrica de evento. Essa abordagem permitiu avaliar o potencial da associação entre dados de infraestrutura e eventos da aplicação como suporte para ajustes mais contextualizados. A Tabela 2 apresenta as configurações adotadas em cada cenário. Cabe destacar que a configuração do Cenário 1, ao utilizar exclusivamente métricas de infraestrutura como uso de CPU e transmissão de dados, aproxima-se do comportamento do HPA, uma vez que esse mecanismo se baseia em métricas de hardware para orientar o escalonamento. A utilização do KEDA, por sua vez, estende as capacidades do HPA ao incorporar também eventos da aplicação como gatilhos de escalonamento. Assim, os experimentos conduzidos permitem não apenas avaliar a proposta com métricas combinadas, mas também contrastá-la com um cenário de referência inspirado em estratégias já consolidadas na literatura.

**Tabela 2. Configurações e Cenários do Experimento de Dimensionamento**

| Experimentos | Duração (h) | Métricas Configuradas | Cenário de Carga |
|---|---|---|---|
| Cenário 1: Métricas de hardware | 2 | Uso de CPU ou transmissão de dados | Alternância entre: Carga Regular (55-65 usuários) Sobrecarga (até 700 usuários) |
| Cenário 2: Métricas combinadas | 2 | Uso de CPU ou transmissão de Dados e taxa de requisições por segundo | Alternância entre: Carga Regular (55-65 usuários) Sobrecarga (até 700 usuários) |

A definição dos *thresholds* foi um aspecto crítico da configuração experimental, exigindo a identificação de limites adequados para acionar ou desativar réplicas de forma oportuna [Dogani et al. 2023]. Para os microsserviços da aplicação *Hubs*, esses valores foram definidos com base em análise gráfica e observações empíricas das métricas coletadas. Durante os testes, verificou-se que, sob carga regular, o sistema mantinha estabilidade, enquanto o escalonamento era acionado a partir de 135 requisições por segundo, conforme identificado com o auxílio das ferramentas *Prometheus* e *Grafana* [Grafana, 2024 ]. A calibragem dos valores foi realizada manualmente, com ajustes baseados em testes sucessivos, buscando evitar tanto a criação de réplicas desnecessárias quanto respostas tardias. Os thresholds finais foram incorporados ao arquivo ScaledObject, componente do KEDA responsável por definir e gerenciar as regras de dimensionamento automático, como apresentado na Listagem 1.

Com o suporte da biblioteca *Pandas* [Pandas 2025], foram calculadas estatísticas descritivas para cada conjunto de dados, selecionando experimentos equivalentes para garantir consistência e minimizar vieses. Esses dados estruturados representam a principal entrega desta etapa, servindo de base às análises exploratórias. Os experimentos também consolidaram os processos dos módulos *Monitor* e *Analyze* do modelo MAPE-K. No módulo *Monitor*, *scripts* integrados ao *Prometheus* asseguraram rastreabilidade, padronização e reprodutibilidade dos dados, com documentação para replicação [LABORA-INF-UFG 2025]. No módulo *Analyze*, os *scripts* processaram os dados coletados e geraram análises estatísticas voltadas à identificação de padrões e tendências do sistema.

**Listagem 1. Configuração do arquivo *ScaledObject***

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: reticulum-scaledobject
  namespace: hcce
spec:
  scaleTargetRef:
    name: reticulum
  minReplicaCount: 1
  maxReplicaCount: 3
  cooldownPeriod: 60
  pollingInterval: 30
  triggers:
   - type: prometheus
     metadata:
       serverAddress: http://prometheus-server.monitoramento.svc.cluster.local
       metricName: rate_k6_http_reqs_total
       query: rate(k6_http_reqs_total[1m])
       threshold: "135"
   - type: prometheus
     metadata:
       serverAddress: http://prometheus-server.monitoramento.svc.cluster.local
       metricName: reticulum_cpu_usage_seconds_total
       query: >
  sum(rate(container_cpu_usage_seconds_total{namespace='hcce',
  pod=~'reticulum.*'}[5m])) 100 / count(node_cpu_seconds_total{mode='idle'})
       threshold: "8.5"
```

## 5. Resultados

Os experimentos geraram uma quantidade significativa de dados, organizados em dois conjuntos distintos, possibilitando a análise de desempenho do sistema em dois cenários: um baseado exclusivamente em métricas de hardware e outro que combinou métricas de hardware e eventos da aplicação, conforme descrito na Seção 4. Essas abordagens foram avaliadas a partir da estabilidade do sistema durante o dimensionamento e na eficiência da alocação de recursos para atender às demandas de forma ágil e proporcional.

A análise dos dados coletados nos experimentos foi conduzida por meio de técnicas de Estatística Descritiva e Exploratória, que permitem identificar padrões, variabilidade e tendências [López-Ramírez et al. 2024]. As métricas avaliadas, listadas na Tabela 3, incluem taxa de requisições por segundo (RPS), latência (L), tempo de resposta (TR) e percentual de sucesso das requisições (SR), representando aspectos importantes para a validação do sistema.

**Tabela 3. Estatísticas Descritivas dos Experimentos.**

| Experimento | Nº de registros | Métrica | Média | Desvio Padrão | Mínimo | 25% | Mediana | 75% | Máximo |
|---|---|---|---|---|---|---|---|---|---|
| Métricas de hardware | 23295 | RPS | 66.37 | 102.92 | 0.03 | 12.86 | 24.04 | 39.73 | 361.58 |
| | 23295 | L (s) | 0.67 | 0.31 | 0.17 | 0.45 | 0.47 | 0.96 | 2.00 |
| | 23295 | TR(s) | 0.50 | 0.30 | 0.17 | 0.28 | 0.44 | 0.52 | 2.00 |
| | 23295 | SR (%) | 98.49 | 0.02 | 93.33 | 98.56 | 99.85 | 99.99 | 100.00 |
| Métricas combinadas | 23077 | RPS | 61.85 | 90.48 | 0.03 | 13.28 | 27.65 | 53.31 | 364.54 |
| | 23077 | L (s) | 0.42 | 0.08 | 0.17 | 0.41 | 0.43 | 0.49 | 0.57 |
| | 23077 | TR (s) | 0.38 | 0.12 | 0.16 | 0.29 | 0.41 | 0.49 | 0.61 |
| | 23077 | SR (%) | 99.35 | 0.01 | 97.69 | 98.91 | 99.90 | 99.99 | 100.00 |

Requisições por segundo - RPS; Latência - L; Tempo de resposta - TR; Sucesso das requisições - SR.

A Figura 2 apresenta diferenças relevantes observadas entre os experimentos. O cenário de métricas combinadas apresentou mediana superior na taxa de requisições por segundo (27,65 RPS contra 24,04 RPS), conforme apresentado na Figura 2a, além de

melhores resultados em latência (Figura 2b) e tempo de resposta (Figura 2c), com 0,43 segundos e 0,41 segundos, respectivamente, frente a 0,47 segundos e 0,44 segundos no cenário de métricas de hardware.

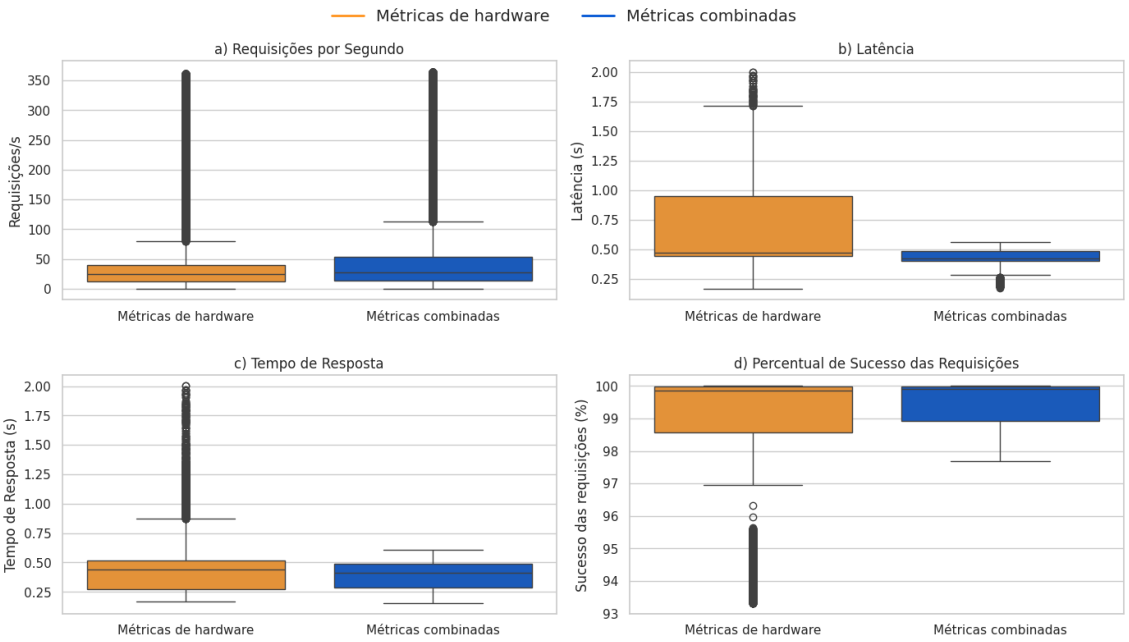


**Figura 2. Boxplots comparativo entre os indicadores de desempenho**

No percentual de sucesso das requisições, o cenário combinado foi ligeiramente superior, alcançando 99,9% contra 99,85%. Verifica-se também, que o cenário de hardware mostrou maior dispersão nas métricas, como latência e tempo de resposta, com amplitudes interquartis mais amplas e presença de outliers, especialmente no percentual de sucesso das requisições. O cenário combinado, por outro lado, apresentou distribuições mais compactas e consistentes, indicando maior estabilidade.

Os mapas de calor da Figura 3 evidenciam as correlações entre métricas nos dois cenários analisados. No cenário baseado em hardware (Figura 3a), a correlação entre latência e uso de memória do *Dialog* foi fraca, com valor de 0,1. Já no cenário de métricas combinadas (Figura 3b), essa correlação subiu para 0,8, evidenciando forte relação quando eventos da aplicação são incorporados. Alterações semelhantes ocorreram na correlação entre latência e uso de CPU no *Spoke*, que passou de 0,2 para 0,7, e entre latência e transmissão de dados no *Nearspark*, que aumentou de 0,3 para 0,8. A inclusão de eventos revela relações ocultas quando apenas métricas de hardware são utilizadas.

As relações entre número de réplicas e latência são analisadas sob a perspectiva de três microsserviços: *Reticulum*, *Hubs* e *Spoke*. O cenário baseado exclusivamente em métricas de hardware apresentou latências superiores em comparação ao cenário de métricas combinadas, como mostra a Figura 4. A análise considera médias e respectivos intervalos de confiança de 95%, permitindo avaliar a consistência dos resultados entre diferentes configurações. A abordagem com métricas combinadas revelou impacto positivo em todos os microsserviços, resultando em latências menores, independentemente do número de réplicas.

De forma semelhante ao comportamento observado na latência, a relação entre o

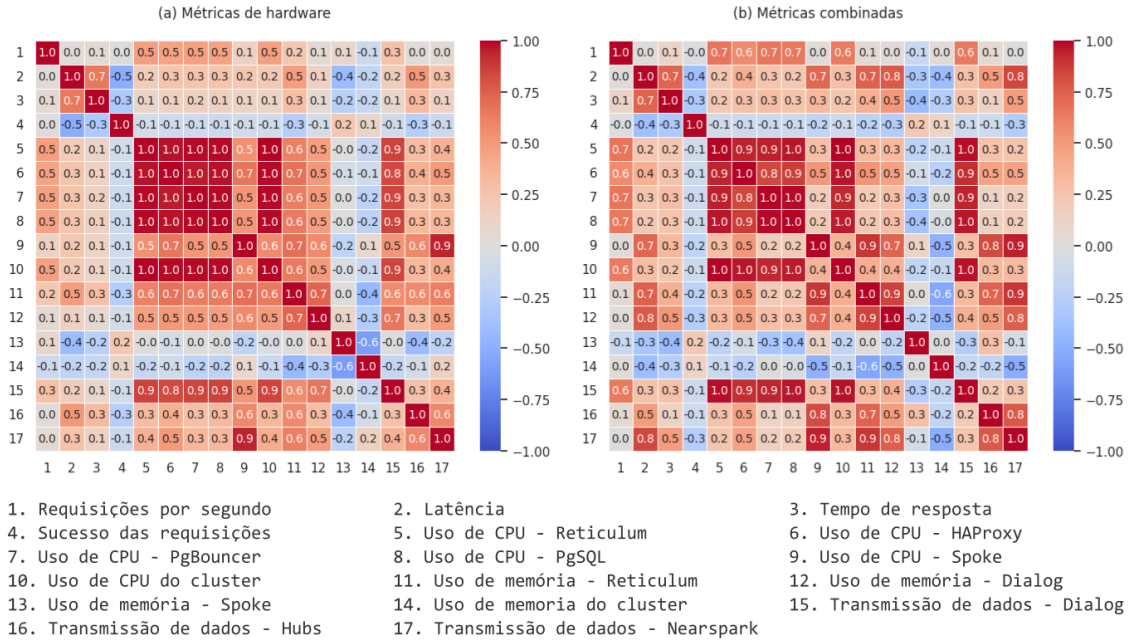


**Figura 3. Mapas de calor dos conjuntos de dados**

número de réplicas e o tempo de resposta evidencia a vantagem do cenário baseado em métricas combinadas. Conforme ilustrado na Figura 5, o tempo de resposta nos microsserviços *Reticulum*, *Hubs* e *Spoke* foi sistematicamente menor quando o dimensionamento considerou tanto métricas de hardware quanto eventos da aplicação. No microsserviço *Hubs*, por exemplo, a abordagem combinada manteve um tempo de resposta mais estável mesmo com o aumento do número de réplicas, enquanto no cenário baseado apenas em métricas de hardware observou-se um crescimento mais acentuado. A análise com intervalos de confiança reforça a evidência de que o modelo combinado contribui para mitigar impactos negativos em situações de dimensionamento intensivo.

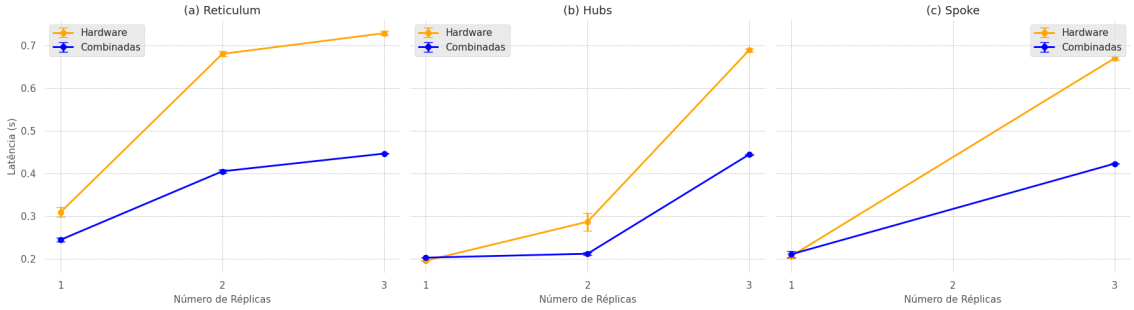


**Figura 4. Relação entre o número de réplicas e a latência nos microsserviços analisados (com intervalo de confiança de 95%)**

Por fim, a Figura 6 apresenta a média de uso de recursos computacionais, como CPU e memória, nos dois cenários analisados. Em relação à CPU, o consumo médio foi semelhante em ambas as abordagens, com o cenário baseado exclusivamente em métricas de hardware registrando 39,14%, um valor ligeiramente superior aos 37,34% observados no cenário combinado. Já no uso de memória, a diferença foi mais expressiva, com o cenário

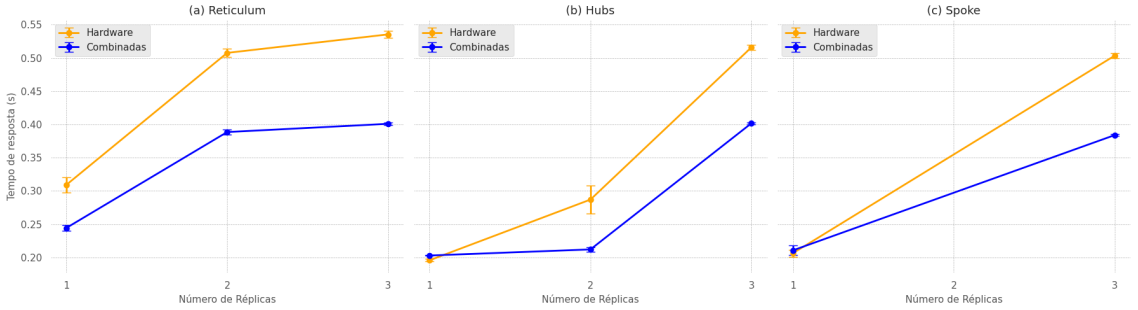


**Figura 5. Relação entre o número de réplicas e o tempo de resposta nos microsserviços analisados (com intervalo de confiança de 95%)**

de hardware atingindo uma média de 85,45%, enquanto o modelo combinado demonstrou maior eficiência ao reduzir esse consumo para 81,15%. Esses resultados sugerem vantagens potenciais da abordagem combinada, que, ao integrar eventos da aplicação, possibilitou uma melhor alocação de memória, mesmo diante de cargas variáveis e dimensionamento ativo.

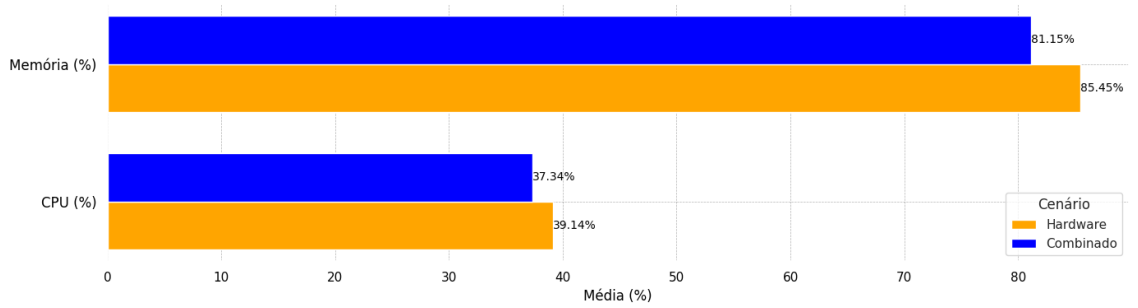


**Figura 6. Comparação do uso de CPU e memória entre os cenários analisados**

De forma geral, os resultados obtidos a partir da análise descritiva sugerem que a abordagem baseada em métricas combinadas apresenta vantagens em relação ao cenário que considera apenas métricas de hardware. Essa estratégia demonstrou maior eficiência na redução da latência e do tempo de resposta, além de maior consistência durante o processo de dimensionamento automático. Também se destacou no uso de recursos computacionais, especialmente no consumo de memória, que foi, em média, inferior ao observado no cenário exclusivamente baseado em hardware. A integração de eventos da aplicação às métricas de infraestrutura possibilitou a identificação de correlações mais representativas, contribuindo para uma gestão mais eficiente da escalabilidade do sistema. A combinação de métricas favorece a alocação de recursos sob alta variabilidade.

## 6. Considerações Finais

Esta pesquisa propôs uma arquitetura de dimensionamento adaptativo com suporte ao aprendizado, voltada à escalabilidade da aplicação *Hubs*, a partir de métricas de CPU, transmissão de dados e taxa de requisições. Conjuntos de dados estruturados, oriundos de experimentos com cenários distintos de monitoramento, compõem a principal entrega deste estágio da investigação, base para análises descritivas, exploratórias e preditivas.

Os resultados obtidos indicam que a abordagem baseada em métricas combinadas apresenta potencial para superar limitações observadas em estratégias que utilizam

exclusivamente métricas de hardware. Observou-se, por exemplo, maior agilidade no ajuste de réplicas em momentos de sobrecarga e comportamento mais estável sob condições críticas. Embora promissora, a abordagem ainda enfrenta desafios operacionais, como a definição precisa de *thresholds* para os gatilhos de escalonamento, que impactam diretamente o equilíbrio entre consumo de recursos e tempo de resposta.

Como trabalhos futuros, destaca-se a integração de modelos preditivos ao ciclo de dimensionamento, com vistas à consolidação dos módulos *Plan*, *Execute* e *Knowledge*, projetados na arquitetura. Esses módulos atuarão na definição dinâmica de *thresholds*, na aplicação automatizada das decisões e na consolidação do conhecimento extraído de execuções anteriores, resultando no aprimoramento da capacidade da arquitetura de adaptar-se às variações de carga com maior precisão.

## Referências


Bartolomeo, G., Cao, J., Su, X., and Mohan, N. (2023). Characterizing distributed mobile augmented reality applications at the edge. In *Companion of the 19th International Conference on emerging Networking EXperiments and Technologies*, pages 9–18.

Benmerar, T. Z., Theodoropoulos, T., Fevereiro, D., Rosa, L., Rodrigues, J., Taleb, T., Barone, P., Tserpes, K., and Cordeiro, L. (2023). Intelligent multi-domain edge orchestration for highly distributed immersive services: an immersive virtual touring use case. In *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, pages 381–392. IEEE.

Cheng, K., Zhang, S., Tu, C., Shi, X., Yin, Z., Lu, S., Liang, Y., and Gu, Q. (2023). Proscale: Proactive autoscaling for microservice with time-varying workload at the edge. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1294–1312.

Dimolitsas, I., Spatharakis, D., Dechouniotis, D., Zafeiropoulos, A., and Papavassiliou, S. (2023). Multi-application hierarchical autoscaling for kubernetes edge clusters. In *2023 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 291–296. IEEE.

Dogani, J., Namvar, R., and Khunjush, F. (2023). Auto-scaling techniques in container-based cloud and edge/fog computing: Taxonomy and survey. *Computer Communications*, 209:120–150.

Gonçalves, A. L. d. J., Oliveira-Jr, A., and Freitas, L. A. (2023). Revisao sistemática das aplicaçoes imersivas com base nas tecnologias habilitadoras b5g/6g, mec e ia. *Anais da XI Escola Regional de Informática de Goiás*.

Grafana, 2024. Grafana: The open and composable observability platform. `https://grafana.com/`. Acesso em 21 de janeiro de 2024.

Han, B., Pathak, P., Chen, S., and Yu, L.-F. C. (2022). Comic: A collaborative mobile immersive computing infrastructure for conducting multi-user xr research. *IEEE Network*.

Hubs Foundation (2024). Hubs foundation - open source social vr platform. Accessed: 2024-07-12.

k6.io (2024). k6 Open Source: A modern load testing tool for developers. Acessado em 20 de dezembro de 2024.

KEDA Project (2024). Keda: Kubernetes-based event driven autoscaler. Acessado em: 10-07-2024.

Kubernetes (2022). Kubernetes documentation.

LABORA-INF-UFG (2025). Repositório do projeto adaptscale. `https://github.com/LABORA-INF-UFG/AdaptScale`.

López-Ramírez, G. A., Aragón-Zavala, A., and Vargas-Rosales, C. (2024). Exploratory data analysis for path loss measurements: Unveiling patterns and insights before machine learning. *IEEE Access*.

Malburg, L., Hoffmann, M., and Bergmann, R. (2023). Applying mape-k control loops for adaptive workflow management in smart factories. *Journal of Intelligent Information Systems*, 61(1):83–111.

Mehta, R., Sahni, J., and Khanna, K. (2023). Task scheduling for improved response time of latency sensitive applications in fog integrated cloud environment. *Multimedia Tools and Applications*, 82(21):32305–32328.

Nguyen, T.-T., Yeom, Y.-J., Kim, T., Park, D.-H., and Kim, S. (2020). Horizontal pod autoscaling in kubernetes for elastic container orchestration. *Sensors*, 20(16):4621.

Pandas (2025). Pandas: Python data analysis library. `https://pandas.pydata.org/`. Accessed: 2025-01-13.

Pelle, I., Czentye, J., Dóka, J., and Sonkoly, B. (2019). Towards latency sensitive cloud native applications: A performance study on aws. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 272–280. IEEE.

Prometheus, T. (Acesso em 21 de janeiro de 2024). Prometheus: The Comprehensive Monitoring and Alerting Toolkit. `https://prometheus.io/`.

Quattrocchi, G., Incerto, E., Pinciroli, R., Trubiani, C., and Baresi, L. (2024). Autoscaling solutions for cloud applications under dynamic workloads. *IEEE Transactions on Services Computing*.

Santos, J., Wauters, T., Volckaert, B., and De Turck, F. (2023). gym-hpa: Efficient auto-scaling via reinforcement learning for complex microservice-based applications in kubernetes. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE.

Siriwardhana, Y., Porambage, P., Liyanage, M., and Ylianttila, M. (2021). A survey on mobile augmented reality with 5g mobile edge computing: Architectures, applications, and technical aspects. *IEEE Communications Surveys & Tutorials*, 23(2):1160–1192.

Taleb, T., Boudi, A., Rosa, L., Cordeiro, L., Theodoropoulos, T., Tserpes, K., Dazzi, P., Protopsaltis, A. I., and Li, R. (2022). Toward supporting xr services: Architecture and enablers. *IEEE Internet of Things Journal*, 10(4):3567–3586.

Wu, D., Yang, Z., Zhang, P., Wang, R., Yang, B., and Ma, X. (2023). Virtual-reality inter-promotion technology for metaverse: A survey. *IEEE Internet of Things Journal*.

Yuan, H. and Liao, S. (2024). A time series-based approach to elastic kubernetes scaling. *Electronics*, 13(2):285.