UNIVERSIDADE FEDERAL DE ITAJUBÁ

Programa de Pós-Graduação em Ciência e Tecnologia da Computação

Orquestração de Contêineres em Computação de Borda: Uma Avaliação de Desempenho Comparativa entre K3s e K0s

VICTOR LUNARTI VALADÃO

Itajubá

2026

UNIVERSIDADE FEDERAL DE ITAJUBÁ

VICTOR LUNARTI VALADÃO

Orquestração de Contêineres em Computação de Borda: Uma Avaliação de Desempenho Comparativa entre K3s e K0s

Dissertação apresentada ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação da Universidade Federal de Itajubá como requisito parcial para a obtenção do título de Mestre em Computação.

Orientador: Rodrigo Maximiano Antunes de Almeida

Itajubá

2026

À minha esposa, Bárbara, sem sua compreensão, apoio diário e incentivo constante, esta tese não existiria. Aos meus pais, Sirlei e Claiton, por serem a fundação da minha vida, por investirem na minha educação e por nunca me deixarem esquecer do meu potencial. Sem vocês, nada disso faria sentido.

Agradecimentos

Agradeço especialmente ao meu orientador, Rodrigo. Sua paciência, seu incentivo constante e sua capacidade de apontar novos caminhos foram fundamentais. Suas lições sobre perspectiva, resiliência e sobre o valor de uma pergunta bem feita foram tão importantes quanto a própria dissertação. Obrigado por me ensinar a ver além do óbvio e por transformar os desafios desta jornada em valiosas lições de vida. Levo seus conselhos para muito além da academia.

À UNIFEI, e em especial ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação, pela oportunidade. Agradeço também ao corpo docente do departamento, pelos ensinamentos compartilhados ao longo das disciplinas.

Resumo

A crescente adoção de contêineres como uma tecnologia de virtualização leve tem acompanhado a popularidade das arquiteturas de microsserviços, dada a escalabilidade, isolamento e natureza efêmera inerentes aos contêineres. Embora os *frameworks* de orquestração de contêineres tenham sido tradicionalmente utilizados em *data centers*, há um interesse crescente em implantar esses *frameworks* em dispositivos com recursos restritos, como dispositivos de Internet das Coisas (IoT), *gateways* de borda e estações de trabalho de desenvolvedores. Essa mudança é impulsionada pela necessidade de provisionamento, implantação, dimensionamento, rede e balanceamento de carga automatizados em ambientes diversos. No entanto, com uma variedade de soluções de orquestração de contêineres disponíveis, selecionar a plataforma apropriada permanece um desafio para os desenvolvedores. Esta dissertação apresenta uma análise comparativa de plataformas de orquestração de contêineres, começando com uma revisão de literatura para examinar várias plataformas e sua aplicabilidade em diferentes ambientes. Em seguida, K3s e KOs são selecionados para uma avaliação aprofundada sob um cenário específico, focando em seus requisitos mínimos de recursos, bem como no desempenho do *control plane*. Através desta abordagem, identificamos fatores-chave e casos de uso ideais, oferecendo insights sobre a adequação dessas plataformas para ambientes de hardware com recursos restritos.

Palavras-chave:Orquestração de Contêineres, Computação de Borda, *Internet of Things*, IoT, Análise de Desempenho, Dispositivos com Recursos Restritos, K3s, K0s.

Abstract

The rising adoption of containers as a lightweight virtualization technology has paralleled the popularity of microservice architectures, given containers' inherent scalability, isolation, and ephemeral nature. While container orchestration frameworks have traditionally been utilized in data centers, there is growing interest in deploying these frameworks on resource-constrained devices, such as Internet of Things (IoT) devices, edge gateways, and developer workstations. This shift is driven by the need for automated provisioning, deployment, scaling, networking, and load balancing across diverse environments. However, with a range of container orchestration solutions available, selecting the appropriate platform remains a challenge for developers. This paper presents a comparative analysis of container orchestration platforms, beginning with a literature review to examine various platforms and their applicability across different environments. Following this, K3s and KOs are selected for an in-depth evaluation under a specified scenario, focusing on their minimal resource requirements, as well as *control plane*. Through this approach, we identify key factors and optimal use cases, offering insights into the suitability of these platforms for resource-constrained hardware environments.

Keywords: Container Orchestration Edge Computing. Internet of Things, IoT, Performance Analysis, Resource-Constrained Environments, K3s, K0s.

Lista de abreviaturas e siglas

IoT	Internet of Things - Internet das Coisas
COPs	Plataformas de Orquestração de Contêineres
K8s	Kubernetes
CPU	Unidade Central de Processamento
QPs	Questões de Pesquisa
GQM	Goal-Question-Metrics - Objetivo-Pergunta-Métrica
CI	Critérios de Inclusão
CE	Critérios de Exclusão
I/O	Input/Outpu - Entrada/Saída
IOPS	Input/Output Operations Per Second - Operações de Entrada/Saída por Segundo
VMs	Máquinas Virtuais
SBC	Computador de Placa Única
QoS	Qualidade de Serviço
CNCF	Cloud Native Computing Foundation
MSA	Arquitetura de Microserviços
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
TNs	Redes Táticas
VPN	Virtual Private Network - Rede Privada Virtual
VNFs	Funções de Rede Virtualizadas
HPA	Horizontal Pod Autoscaler
DACS	Delay-Aware Container Scheduling
RAP	Resource Adaptive Proxy
ARM	Advanced RISC Machine

SDN	Software-Defined Networking
CNI	Interface de Rede de Contêiner
RAM	Random Access Memory
NFV	Virtualização de Funções de Rede
IA/ML	Inteligência Artificial e Aprendizado de Máquina

Lista de ilustrações

Lista de tabelas

Sumário

1

Introdução

A rápida proliferação de dispositivos IoT em diversos domínios, como cidades inteligentes, saúde e manufatura, criou uma demanda crítica por processamento de dados em tempo real com requisitos de latência mínima (PALLEWATTA; KOSTAKOS; BUYYA, 2024; TUSA et al., 2024; KAUR et al., 2020). Historicamente, o envio desses dados para processamento na nuvem tem sido a abordagem padrão, mas isso frequentemente resulta em desafios de latência e congestionamento de rede(KRISTIANI et al., 2021; PALLEWATTA; KOSTAKOS; BUYYA, 2024; RISCO et al., 2024).

Para mitigar esses desafios, o paradigma de *Edge Computing* (computação de borda) surgiu como uma solução promissora, pois posiciona o processamento e a análise de dados mais próximos à fonte de geração. Essa proximidade não só reduz significativamente a latência e aprimora a responsividade, mas também diminui a dependência de conectividade contínua com a nuvem, permitindo que grandes volumes de dados brutos sejam pré-processados localmente.

No contexto de *Edge Computing*, os containers são fundamentais devido às severas restrições de hardware e energia dos dispositivos de borda. Eles permitem que a inteligência do processamento seja movida da nuvem centralizada para a proximidade das fontes de dados, o que é essencial para aplicações que exigem baixa latência e tempo real. A natureza etereal e modular dos containers facilita a migração dinâmica e a escalabilidade de serviços; por exemplo, um container pode "seguir"um usuário móvel (como um veículo autônomo) mudando-se de um nó de borda para outro quase instantaneamente para manter o desempenho.

Para coordenar e gerenciar esses contêineres em larga escala, as plataformas de orquestração de contêineres (*Container Orchestration Platform* - COP), como o Kubernetes (K8s), consolidaram-se como o padrão da indústria. O Kubernetes automatiza a implantação, o gerenciamento e o escalonamento de aplicações conteinerizadas em ambientes distribuídos. Contudo, o Kubernetes tradicional foi concebido para operar em ambientes de nuvem de grande escala, com requisitos substanciais de Unidade Central de Processamento (CPU) e memória. Isso o torna inadequado para a natureza heterogênea e com restrições de recursos de muitos dispositivos de borda e IoT. A busca por soluções de orquestração mais leves é impulsionada pela necessidade de lidar com a capacidade computacional limitada e a eficiência energética desses nós de borda.

1.1 Objetivo Geral

O objetivo principal deste trabalho é investigar e analisar criticamente as plataformas de orquestração de contêineres aplicáveis a ambientes com recursos restritos, como sistemas embarcados e IoT, por meio de:

1. Um Mapeamento Sistemático da Literatura para contextualizar o estado da arte e identificar desafios e ferramentas.

2. Uma Avaliação Experimental aprofundada do desempenho de plataformas leves selecionadas, especificamente K3s e K0s (alternativas viáveis para o K8s padrão, projetadas para facilitar a implantação e gerenciamento de aplicações em contêineres, especialmente em ambientes com recursos limitados, como sistemas embarcados e IoT).

1.2 Objetivos Específicos

Os objetivos específicos podem ser divididos em duas grandes etapas: a fase de Revisão Sistemática (Objetivos Fundacionais) e a fase de Avaliação Experimental (Objetivos Técnicos).

Fase 1: Revisão Sistemática e Mapeamento da Literatura (Objetivos Fundacionais)

Estes objetivos são guiados pelas Questões de Pesquisa (QPs) da revisão sistemática:

1. Mapear Desafios e Áreas de Pesquisa: Identificar as principais áreas de pesquisa e os desafios abordados na literatura sobre orquestração de contêineres em ambientes de *Edge Computing*, névoa ou contínuo (*continuum*) (QP1).

2. Categorizar Abordagens e Algoritmos: Documentar e categorizar as abordagens, métodos ou algoritmos propostos para a orquestração de contêineres na borda (QP2).

3. Identificar Limitações e Direções Futuras: Analisar as limitações, questões em aberto e direções futuras de pesquisa relacionadas à aplicação da orquestração de contêineres na borda (QP3).

4. Analisar Ferramentas: Determinar quais ferramentas de orquestração de contêineres e tecnologias subjacentes são mais investigadas ou utilizadas na pesquisa sobre borda (QP4).

5. Identificar Domínios de Aplicação: Mapear os domínios de aplicação (como Cidades Inteligentes, Saúde e IoT Industrial) que são abordados pela pesquisa focada em orquestração de contêineres na borda (QP5).

6. Avaliar a Maturidade dos Métodos: Determinar o nível de maturidade dos métodos de avaliação e ambientes de teste utilizados para as soluções de orquestração de contêineres propostas na borda (QP6).

7. Escolha de plataformas para avaliação: Utilizar os resultados da revisão sistemática para justificar a seleção das plataformas leves para avaliação experimental.

Fase 2: Avaliação Experimental (Objetivos Técnicos)

Estes objetivos focam na avaliação prática das plataformas K3s e K0s, conforme detalhado na Metodologia Experimental *Goal-Question-Metrics* (GQM) (BASILI; WEISS, 1984):

1. Avaliar o Desempenho do Plano de Controle: Analisar o desempenho das COPs ao executar o *control plane* em dispositivos com recursos restritos.

2. Medir a Latências: Quantificar o tempo necessário para a criação e destruição de *pods* (menor unidade de execução Kubernetes).

3. Quantificar a Utilização de Recursos: Determinar a utilização de recursos do dispositivo embarcado (CPU, memória e I/O (Input/Output - Entrada/Saída) de disco) sob condições de carga padrão gerenciadas pela plataforma de orquestração (Q1).

4. Analisar o Impacto do Hardware: Investigar como o tipo de armazenamento impacta o desempenho das COPs, medindo IOPS (Input/Output Operations Per Second - Operações de Entrada/Saída por Segundo) e uso de CPU/memória (Q3).

5. Analisar a Influência da Carga: Determinar como o número de *pods* afeta o desempenho e a utilização de recursos das COPs (Q4).

6. Identificar Fatores-Chave: Identificar os fatores-chave que influenciam o desempenho e a viabilidade dessas plataformas em hardware limitado, como o Raspberry Pi 3B+, oferecendo insights para ambientes de borda.

Em última análise, o estudo busca determinar a capacidade dessas soluções em suportar arquiteturas de microsserviços resilientes e de alta disponibilidade em cenários onde o Kubernetes tradicional seria impraticável. Para a coleta de métricas e análise de desempenho, o estudo emprega ferramentas de monitoramento como Prometheus e Grafana.

2

Revisão Bibliográfica

Sistemas Embarcados (*Embedded Systems*) são computadores especializados, integrados em um dispositivo ou sistema maior para executar uma ou algumas funções dedicadas, geralmente com restrições de tempo real, tamanho e consumo de energia, seu foco principal são funcionalidade e confiabilidade. Eles garantem que um dispositivo funcione corretamente (por exemplo, um airbag, um forno de micro-ondas, ou um sistema de freios ABS) (RAC; BRORSSON, 2024), tipicamente com baixa ou nenhuma conectividade externa. Se há comunicação, geralmente é por meio de protocolos de rede local. Os dados são processados localmente para realizar a função primária.

A Internet das Coisas surgiu quando os sistemas embarcados ganharam capacidade de conectividade à internet, permitindo que os dispositivos se comunicassem e gerassem dados em escala. Se trata de uma rede de objetos físicos incorporados com sensores, *software* e outras tecnologias que se conectam e trocam dados com outros dispositivos e sistemas pela internet (QIAN et al., 2019). É a evolução do sistema embarcado que agora possui capacidade de comunicação IP (Internet Protocol).

O gerenciamento de um ecossistema IoT enfrenta desafios significativos devido ao grande número de dispositivos conectados e ao volume massivo de dados gerados. A tarefa de implantar e administrar aplicativos e sistemas operacionais é otimizada pela conteinerização, uma tecnologia de virtualização leve que provê ambientes portáteis, isolados e de rápido início (KAISER et al., 2022). Embora o processamento centralizado em datacenters na nuvem (*Cloud*) tenha sido o padrão histórico, ele se mostra inadequado para as novas exigências de baixa latência e alto tráfego impostas pela evolução da IoT(KRISTIANI et al., 2021).

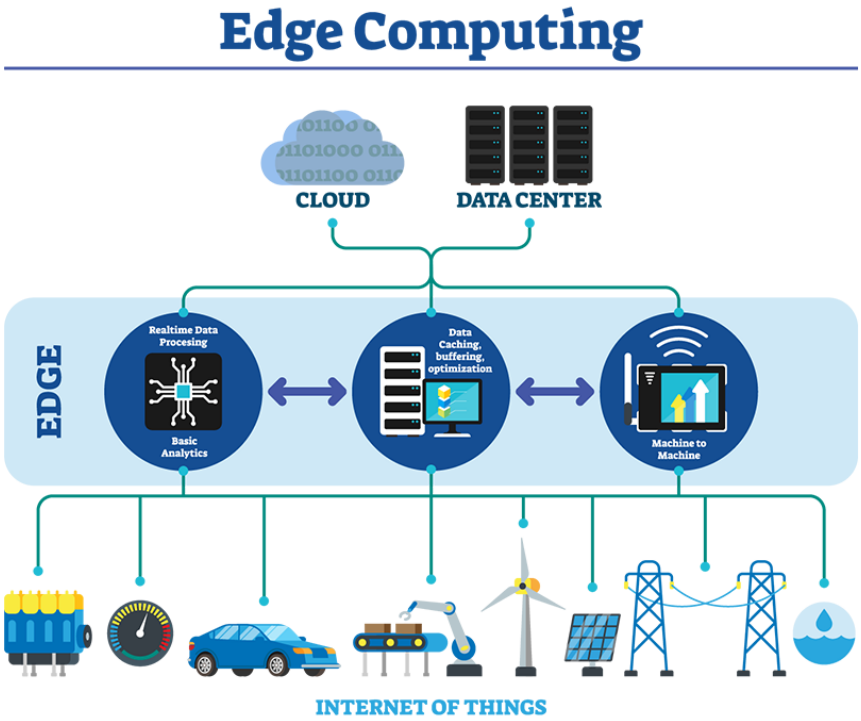


Figura 1 – IoT e *Edge Computing* (EDGE. . . , 2019)

Edge Computing é a resposta aos desafios de latência, largura de banda e soberania de dados do modelo IoT puramente baseado em nuvem (NGUYEN et al., 2020). É uma metodologia de computação que leva o processamento de dados para próximo da fonte de dados (a borda da rede), em vez de depender exclusivamente da nuvem. É a evolução do IoT onde o dispositivo conectado (ou um *gateway* próximo) ganha poder de processamento inteligente para tomar decisões localmente. Como demonstrado na Figura 1, a arquitetura de *Edge Computing* estabelece uma estrutura tripartida onde o fluxo de dados se inicia na camada de Internet das Coisas (IoT), composta por sensores e dispositivos físicos que geram volumes massivos de dados brutos em contextos diversos, como mobilidade urbana, infraestrutura energética e automação industrial. Para mitigar os gargalos de largura de banda e as latências inerentes à computação centralizada, a camada intermediária de borda assume o papel de processamento crítico, realizando análises em tempo real, otimização de cache e facilitando a comunicação direta entre máquinas.

2.1 Virtualização, Contêineires e Orquestradores

O paradigma de *Edge Computing*, depende fundamentalmente da capacidade de implantar e gerenciar aplicações de forma eficiente e isolada nesses novos locais. A virtualização é o conceito tecnológico central que permite essa flexibilidade. Em seu sentido mais amplo, a virtualização refere-se à criação de uma versão virtual de um recurso computacional, como um sistema operacional, um servidor, um dispositivo de armazenamento ou recursos de rede, permitindo que múltiplas cargas de trabalho coexistam de forma independente em um mesmo

hardware. Tradicionalmente, isso era feito por meio de máquinas virtuais (VMs), que virtualizam o hardware subjacente. Cada VM contém um sistema operacional convidado completo, uma cópia virtual do hardware, a aplicação e todas as bibliotecas e dependências associadas (KAISER et al., 2022).

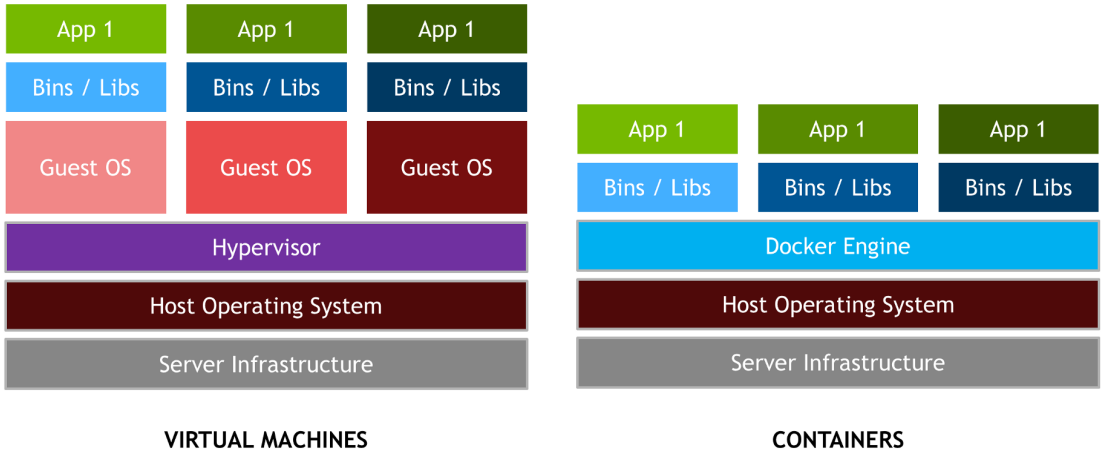


Figura 2 – Comparativo VMs e Contêineres (CONTAINERIZATION, 2020)

Contêineres são descritos como uma forma de virtualização leve no contexto de um sistema operacional, em vez de virtualizar o hardware, os contêineres virtualizam a camada de software acima do sistema operacional, rodando diretamente sobre o sistema operacional host e compartilhando o mesmo kernel e serviços como podemos ver na Figura 2. Isso os torna significativamente mais eficientes e leves em comparação com as VMs (FAYOS-JORDAN et al., 2020). Além disso, Contêineres facilitam o empacotamento de aplicações com todo o código-fonte, bibliotecas e dependências necessárias, tornando a construção, implantação, operação e atualização de aplicações mais fáceis e seguras (CHAN et al., 2022).

Os contêineres são projetados para executar uma aplicação por vez e quando pensamos em escala, existem desafios a se considerar: implantar centenas ou milhares de contêineres está fora do escopo dos motores de gerenciamento, que não consegue lidar eficazmente com questões como controle de limite de recursos, agendamento, balanceamento de carga, verificações de saúde (*health checks*), tolerância a falhas e autoescalonamento (*autoscaling*). É necessária uma camada de abstração superior que controle um ou vários motores de gerenciamento de contêineres, e é aí que entra o orquestrador de contêineres (CASALICCHIO, 2018).

Plataformas de Orquestração de Contêineres são ferramentas de software projetadas para automatizar o gerenciamento programático de contêineres em larga escala (RAHMAN et al., 2023). Elas são responsáveis por facilitar a implantação, o gerenciamento e o escalonamento de aplicações conteinerizadas em ambientes distribuídos (PHUC; PHAN; KIM, 2022; CAI; BUYYA, 2022). Em essência, uma COP atua como um sistema de controle, orquestrando as operações de contêineres (NGUYEN et al., 2020).

A crescente adoção das COPs é impulsionada por diversas necessidades no desenvol-

vimento e operação de software, especialmente no contexto de arquiteturas de microsserviços (KAISER et al., 2022). As principais motivações incluem:

- Escalabilidade e Eficiência de Recursos: COPs são fundamentais para gerenciar um grande número de dispositivos e contêineres, oferecendo capacidade de escalonamento e otimização do uso de recursos computacionais. Elas fornecem mecanismos para o provisionamento dinâmico de recursos e autoescalonamento (KAUR et al., 2020).
- Automação e Agilidade: As plataformas automatizam o provisionamento e a implantação, reduzindo a intervenção manual e acelerando o ciclo de vida do desenvolvimento(PHUC; PHAN; KIM, 2022).
- Gerenciamento em Ambientes Distribuídos e Heterogêneos: Originalmente projetadas para datacenters em nuvem de grande escala, as COPs adaptam-se a ambientes mais complexos e com restrições de recursos, como *Edge Computing* e Internet das Coisas (ROBLES-ENCISO; SKARMETA, 2024). Essa capacidade de aproximar o processamento de dados da fonte é crucial para reduzir a latência e o congestionamento da rede (RAC; BRORSSON, 2024).
- Qualidade de Serviço (QoS) e Desempenho: O objetivo principal é aprimorar o desempenho e a entrega de serviços (SOFIA et al., 2024), minimizando a latência percebida pelo usuário e a latência de ponta a ponta.
- Gerenciamento de Redes e Balanceamento de Carga: As COPs oferecem funcionalidades essenciais para gerenciar a comunicação entre contêineres dentro do cluster e distribuir a carga de trabalho de forma eficiente (LAI; WANG; WEI, 2023).

O conceito de um *continuum* de recursos transparente, que hoje vemos nas COPs, tem suas raízes em sistemas operacionais distribuídos dos anos 1980, que buscavam apresentar múltiplos CPUs como um único computador (IORIO et al., 2023). Essencialmente, o *Continuum* Computing visa unificar conceitualmente as camadas de Edge e Cloud em um ambiente de computação contínuo, proporcionando flexibilidade para que desenvolvedores implementem suas aplicações de forma distribuída em um ecossistema de vários tipos de nós (IORIO et al., 2023).

A evolução da tecnologia de contêineres marcou um ponto de virada, oferecendo uma solução leve e eficiente para empacotar e distribuir aplicações. Isso pavimentou o caminho para o surgimento das COPs. O Kubernetes, em particular, emergiu como a plataforma de orquestração de contêineres mais proeminente e é amplamente considerado o padrão da indústria (PHUC; PHAN; KIM, 2022). Sua arquitetura foi inspirada no sistema Borg do Google (GOETHALS; TURCK; VOLCKAERT, 2022). Em 2015, o Google doou o Kubernetes 1.0 para a *Cloud Native Computing Foundation* (CNCF) (CONTINUING. . . , 2020), solidificando seu papel no ecossistema de código aberto.

2.1.1 Arquitetura Kubernetes

O K8s, amplamente considerado o padrão da indústria para orquestração, implementa uma arquitetura de cluster que separa a lógica de gerenciamento (o Plano de Controle ou *Control Plane*) das cargas de trabalho (os Nós de trabalho ou *Worker Nodes*). Conforme ilustrado na Figura 3, essa arquitetura é composta pelos seguintes componentes principais:

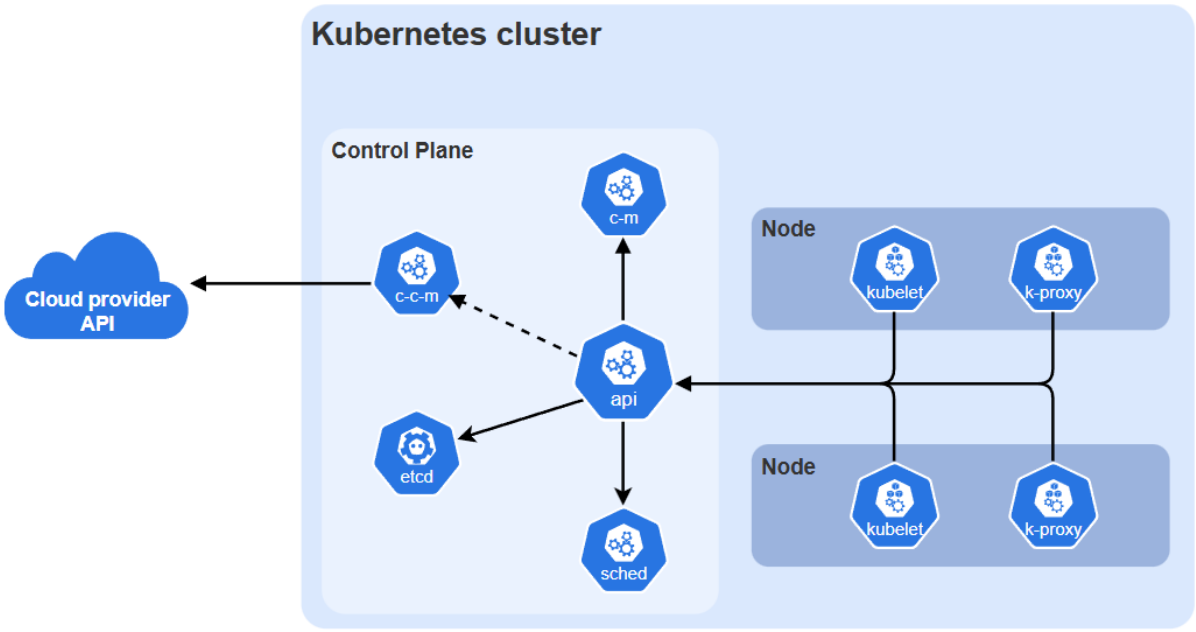


Figura 3 – Arquitetura de um cluster Kubernetes (KUBERNETES. . . , 2025)

O *control plane* é o "cérebro"do cluster, responsável por tomar decisões globais e manter o estado desejado do sistema. Seus componentes centrais incluem:

- *API Server* (api): Atua como o frontend para o *control plane* e o ponto central de comunicação. Todos os outros componentes interagem através dele.
- etcd: Um banco de dados chave-valor consistente e de alta disponibilidade usado como o armazenamento principal do Kubernetes para todos os dados do cluster, garantindo a persistência do estado.
- *Scheduler* (sched): Este componente monitora novos *pods* que ainda não foram atribuídos a um nó e seleciona o nó ideal para eles executarem.
- Controller Manager (c-m): Executa processos de controle que regulam o estado do cluster. Ele executa os "loops de controle"que trabalham para fazer com que o estado atual do cluster corresponda ao estado desejado.

- *Cloud Controller Manager* (c-c-m) Este componente é responsável por integrar o cluster com a infraestrutura de um provedor de nuvem específico (como AWS, Azure ou GCP). Ele gerencia controladores que dependem da nuvem, como balanceadores de carga ou volumes de armazenamento.

A replicação e a distribuição das funcionalidades do *control plane* são requisitos para alcançar a alta disponibilidade em um sistema distribuído. Em cenários de multi-rede, o uso de múltiplos planos de controle é implementado para melhorar a resiliência da implantação (RISCO et al., 2024).

Os Nós (*Nodes*) são as máquinas (virtuais ou físicas) onde as aplicações são efetivamente executadas. Cada nó é gerenciado pelo *control plane* e contém os serviços necessários para rodar os *pods*, que são a unidade atômica do Kubernetes. No Kubernetes, o contêiner não é a menor unidade manipulável; esse papel cabe ao *Pod*. Um *Pod* é um invólucro lógico que encapsula um ou mais contêineres que compartilham armazenamento, rede (IP único) e especificações de execução. Eles são efêmeros por natureza: se um nó falha, os *pods* não são "consertados", mas sim substituídos por novas réplicas em outros nós saudáveis. Para gerenciar esses *pods*, cada nó executa:

- kubelet: Um agente que é executado em cada nó. Ele se comunica com o API Server para garantir que os contêineres descritos nos *pods* estejam sendo executados e estejam saudáveis.
- kube-proxy (k-proxy): Um proxy de rede executado em cada nó que mantém as regras de rede. Ele permite a comunicação de rede para seus *pods*, tanto interna quanto externamente, e gerencia o balanceamento de carga.

Em clusters com nós geograficamente distribuídos, o *control plane* visa garantir a alta disponibilidade e a tolerância a falhas. Em arquiteturas baseadas em contêineres, o loop de controle garante que falhas sejam corrigidas automaticamente, alcançando o estado desejado e promovendo a resiliência (CENTOFANTI et al., 2024).

Se a infraestrutura for composta por nós espalhados geograficamente, o *control plane* rodando em um cluster remoto pode fornecer resiliência de serviço aprimorada em caso de desconexão da rede principal, garantindo que a falha potencial de um nó local não leve à disrupção do serviço graças ao reagendamento automático de *pods* (IORIO et al., 2023).

Em arquiteturas modernas que desagregam o *control plane* em microsserviços (uma abordagem comum na borda), é possível eliminar o ponto único de falha. Em comparação com arquiteturas monolíticas, os controladores baseados em microsserviços demonstraram uma redução de 53,41% no tempo de recuperação durante falhas (PéREZ et al., 2023).

2.2 *Edge Computing*

O conceito de *Edge Computing* representa uma extensão estratégica e um paradigma em rápido crescimento da computação em nuvem tradicional, que posiciona recursos computacionais, de tratamento de dados, armazenamento e rede mais próximos das fontes de dados e dos dispositivos finais (RAC; BRORSSON, 2024). Esta descentralização tem como objetivo principal melhorar o desempenho e a entrega de serviços, pois reduz a latência da rede e o consumo de largura de banda ao processar dados localmente na borda, em vez de enviá-los para a nuvem centralizada (TUSA et al., 2024).

O número e a variedade de dispositivos IoT conectados estão crescendo rapidamente, gerando um volume massivo de dados em domínios de aplicação inteligentes (PALLEWATTA; KOSTAKOS; BUYYA, 2024). Em um modelo puramente centrado na nuvem, o envio desses grandes volumes de dados para datacenters centrais incorre em alta latência e congestionamento de largura de banda, o que não é ideal para aplicações sensíveis ao tempo. A *Edge Computing* resolve isso ao permitir que o processamento ocorra o mais próximo possível da geração de dados (TUSA et al., 2024).

Essa proximidade permite que os dispositivos de borda gerenciem tarefas de forma autônoma e respondam rapidamente a eventos, o que é essencial para otimizar a eficiência operacional e garantir a confiabilidade do serviço (GOETHALS; TURCK; VOLCKAERT, 2022). A arquitetura de microserviços (MSA), por exemplo, é popular para o desenvolvimento de aplicações IoT em ambientes de *Edge Computing*, devido à sua natureza desacoplada, independentemente implementável e escalável, facilitando a colocação distribuída e a composição dinâmica (PALLEWATTA; KOSTAKOS; BUYYA, 2024). Microsserviços são comumente conteinerizados com tecnologias como Docker e orquestrados por plataformas como Kubernetes, garantindo conectividade contínua em recursos distribuídos.

2.3 Arquitetura de Referência para o *Continuum* Borda-Nuvem

A integração entre as capacidades praticamente ilimitadas da nuvem e a proximidade física da borda consolidou o paradigma conhecido como *Edge-Cloud Continuum*. Este modelo arquitetural, ilustrado na Figura 4, propõe que os recursos computacionais não devem ser tratados como silos isolados, mas sim como um espectro contínuo de processamento gerenciado de forma unificada(IORIO et al., 2023).

Nesta arquitetura hierárquica, o Kubernetes atua como a camada de abstração transversal, permitindo que cargas de trabalho sejam orquestradas dinamicamente entre três níveis lógicos distintos. Cada nível possui responsabilidades específicas para mitigar os desafios de latência e largura de banda identificados na literatura (TUSA et al., 2024).

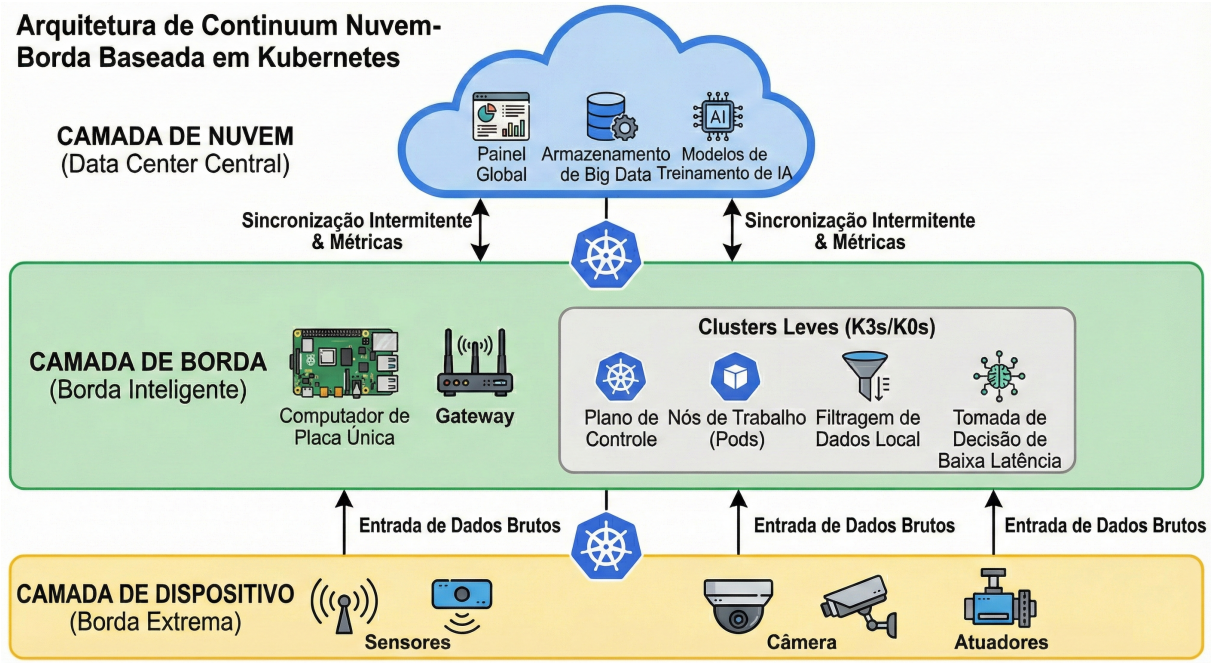


Figura 4 – Arquitetura de Referência para o *Continuum* Borda-Nuvem baseada em Kubernetes. (Fonte: Próprio Autor)

2.3.1 Camada de Nuvem (*Core Cloud*)

No topo da hierarquia encontra-se a camada de Nuvem ou *Data Center* Central. Esta camada caracteriza-se pela alta disponibilidade de recursos computacionais e de armazenamento, mas sofre com latências de rede mais elevadas em relação ao usuário final (PALLEWATTA; KOSTAKOS; BUYYA, 2024).

Neste modelo de referência, a Nuvem não é utilizada para o processamento de tempo real crítico, mas sim para tarefas de "caminho frio", tais como:

- **Armazenamento Histórico:** Retenção de longo prazo dos dados agregados e filtrados enviados pela borda.
- **Treinamento de Modelos de IA:** Execução de algoritmos complexos de aprendizado de máquina que exigem GPUs potentes, gerando modelos que serão posteriormente implantados na borda para inferência.
- **Observabilidade Global:** Centralização de painéis de controle (*dashboards*) para monitoramento da saúde de todo o parque de dispositivos distribuídos.

A convergência entre a Internet das Coisas (IoT) e a Inteligência Artificial (IA) tem impulsionado a transformação digital na indústria, frequentemente referida como Indústria 4.0. Neste cenário, algoritmos de Machine Learning (ML) são empregados para tarefas críticas, como manutenção preditiva, detecção de falhas em tempo real e controle de qualidade automatizado através de visão computacional (KAUR et al., 2020; REHAN et al., 2023).

Para aplicações industriais sensíveis ao tempo, a latência de rede introduzida pelo envio de dados brutos para a nuvem é inaceitável. Portanto, a fase de inferência — a execução do modelo já treinado para tomada de decisão — deve ocorrer na Camada de Borda Inteligente (*Smart Edge*) (LIANG et al., 2023; TUSA et al., 2024).

A orquestração de contêineres desempenha um papel fundamental neste processo, permitindo que modelos atualizados sejam implantados de forma atômica e padronizada em diversos dispositivos fabris heterogêneos. Por outro lado, a fase de treinamento dos modelos exige processamento massivo de dados históricos e capacidades de hardware (como clusters de GPUs de alto desempenho) que são inviáveis técnica e economicamente na borda. Conforme estabelecido na arquitetura de referência, a nuvem assume este papel, processando grandes volumes de dados para refinar os modelos, que são posteriormente sincronizados com os nós de borda (RAC; BRORSSON, 2024).

Além do treinamento, a nuvem atua como um mecanismo de segurança e elasticidade através do conceito de descarregamento de tarefas (*task offloading*). Em cenários onde a capacidade de processamento dos nós de borda é saturada — por exemplo, durante um pico de análise de vídeo em alta resolução ou fusão de dados de múltiplos sensores — o sistema de orquestração pode redirecionar dinamicamente o excedente de carga para a nuvem (DALGKITSIS et al., 2021; NGUYEN et al., 2020).

Este modelo híbrido permite que a indústria mantenha uma infraestrutura local enxuta e eficiente, recorrendo à capacidade elástica da nuvem apenas sob demanda. A eficácia dessa estratégia, contudo, depende diretamente da capacidade do orquestrador em gerenciar recursos limitados e decidir, em tempo real, onde cada contêiner deve ser executado para respeitar os Acordos de Nível de Serviço (SLA) de latência e consumo energético (TRAN; KIM, 2024).

2.3.2 Camada de Borda Inteligente (*Smart Edge*)

A camada intermediária é o foco central desta pesquisa. Ela é composta por nós de computação localizados fisicamente próximos à fonte de dados, como *gateways* IoT e computadores de placa única (SBCs) (NGUYEN; PHAN; KIM, 2022).

O objetivo fundamental desta camada é garantir a soberania dos dados e a baixa latência. Ao invés de enviar todo o tráfego bruto para a nuvem — o que consumiria largura de banda excessiva — a Borda processa os dados localmente. É neste cenário que se justifica o uso de distribuições leves de Kubernetes. Visto que o hardware nesta camada possui restrições severas de memória e CPU, o orquestrador deve possuir um consumo mínimo para permitir que os recursos sejam dedicados aos microsserviços de aplicação (KJORVEZIROSKI; FILIPOSKA, 2022; KAISER et al., 2022).

2.3.3 Camada de Dispositivos (*Far Edge* / *Device Layer*)

Na base da arquitetura encontram-se os dispositivos finais, como sensores, atuadores e câmeras. Nesta proposta, estes dispositivos são considerados produtores de dados. Eles não executam a orquestração de contêineres diretamente, mas estão conectados via rede local aos nós da Camada de Borda. O fluxo de dados segue destes dispositivos para os *pods* em execução na Borda, onde ocorre a filtragem e a tomada de decisão imediata (ex.: parar uma máquina industrial ao detectar uma falha), garantindo a resiliência do sistema mesmo em casos de interrupção da conectividade com a Internet.

2.3.4 Síntese e Direcionamento para a Pesquisa Experimental

A arquitetura de referência para o *Continuum* Borda-Nuvem apresentada evidencia que a eficiência do sistema como um todo é dependente da robustez da Camada de Borda Inteligente. Embora a Nuvem ofereça poder computacional ilimitado e os Dispositivos Finais garantam a capilaridade da coleta de dados, é na Borda que reside o gargalo crítico de desempenho: a necessidade de orquestrar contêineres complexos em hardware severamente limitado.

Se o *control plane* do orquestrador consumir recursos excessivos nesta camada intermediária, todo o fluxo de processamento local é comprometido, invalidando os benefícios de baixa latência propostos pelo modelo. Conclui-se, portanto, que a viabilidade prática desta arquitetura não depende apenas do design lógico, mas da eficiência do software de orquestração escolhido.

Desta forma, a avaliação experimental conduzida neste trabalho (detalhada nos Capítulos 3 e 4) não se limita a um *benchmark* isolado, mas busca validar empiricamente a sustentabilidade desta arquitetura de referência. Ao submeter as plataformas K3s e K0s a testes de estresse no Raspberry Pi 3B+, objetiva-se determinar se estas distribuições leves são capazes de suportar a carga operacional exigida pela Camada de Borda, preenchendo a lacuna entre o modelo teórico do *Continuum* e sua implementação física real.

3

Metodologia

Para alcançar os objetivos geral e específicos delineados no Capítulo 1, este capítulo apresenta a metodologia de pesquisa adotada. A abordagem metodológica é estruturada em duas fases principais e complementares, projetadas para fornecer tanto uma visão ampla do campo quanto uma análise técnica aprofundada. A primeira fase consiste em um Mapeamento Sistemático da Literatura, empregando a estrutura PICOC - uma estrutura metodológica utilizada para formular questões de pesquisa claras, composta por cinco elementos: População, Intervenção, Comparação, Outcome (Resultado) e Contexto - e critérios definidos de inclusão e exclusão, com o objetivo de contextualizar o estado da arte, identificar desafios e mapear as ferramentas de orquestração aplicáveis a ambientes de borda.

A segunda fase compreende uma Avaliação Experimental, que utiliza os insights da revisão para justificar a seleção e analisar o desempenho prático das plataformas leves K3s e K0s. Esta avaliação é guiada pela abordagem GQM e executada em hardware com recursos restritos (Raspberry Pi 3B+), permitindo uma análise quantitativa dos fatores que impactam a viabilidade dessas soluções.

3.1 Revisão Sistemática

Para identificar e categorizar as melhores práticas e aplicações de orquestração de contêineres em ambientes de *Edge Computing*, será conduzido mapeamento sistemático. A metodologia empregada basea-se em Petersen (PETERSEN et al., 2008), incorporando a estrutura PICOC e utilizando o software Parsifal (SEGATTO W.; FREITAS, 2025) para o gerenciamento da revisão sistemática.

3.1.1 Aplicação da Estrutura PICOC

A estrutura PICOC é aplicada da seguinte forma:

- População: O estudo foca em *edge computing*, sistemas embarcados e IoT.

- Intervenção: As intervenções estudadas incluem K8s, Docker Swarm, Apache Mesos e outras tecnologias de orquestração de contêineres.
- Comparação: Os critérios de avaliação abrangem confiabilidade do sistema, escalabilidade, desempenho, segurança, custo-benefício e facilidade de uso.
- Resultados: Os resultados investigados incluem levantamentos (*surveys*), revisões, benchmarking, comparações, melhores práticas e avaliações de técnicas de orquestração de contêineres.
- Contexto: A pesquisa é conduzida no contexto de práticas DevOps, administração de sistemas e organizações que utilizam tecnologias de conteinerização.

3.1.2 Definição das Questões de Pesquisa

As QPs que guiam este estudo são:

- QP1: Quais são as principais áreas de pesquisa e desafios abordados na literatura sobre orquestração de contêineres especificamente em ambientes de borda, névoa ou contínuo?
- QP2: Quais abordagens, métodos ou algoritmos são propostos para a orquestração de contêineres na borda?
- QP3: Quais são as limitações, questões em aberto ou direções futuras de pesquisa identificadas em relação à orquestração de contêineres e sua aplicação na borda ou em todo o contínuo da computação?
- QP4: Quais ferramentas de orquestração de contêineres e tecnologias de contêineres subjacentes são investigadas ou utilizadas na pesquisa sobre borda?
- QP5: Quais domínios de aplicação são abordados ou aproveitados pela pesquisa focada na orquestração de contêineres em borda?
- QP6: Qual o nível de maturidade dos métodos de avaliação e ambientes de teste para as soluções de orquestração de contêineres propostas na borda?

Estas questões visam compreender de forma abrangente o cenário atual da orquestração de contêineres na borda.

3.1.3 Protocolo de Busca

Para garantir uma cobertura abrangente da literatura relevante, foi desenvolvido um protocolo de busca:

("*Edge Computing*"OR "*Embedded Systems*"OR "*Fog Computing*"OR "*Internet of Things*")
AND
("Apache Mesos"OR "*Container Orchestration*"OR "Docker Swarm"OR "Kubernetes")
AND
("*Cost-Effectiveness*"OR "*Ease of Use*"OR "*Performance*"OR "*Scalability*"OR "*Security*"OR "*System reliability*")

Este protocolo orienta a busca em múltiplas bases de dados acadêmicas para identificar estudos pertinentes.

3.1.4 Critérios de Inclusão e Exclusão

Critérios de inclusão e exclusão foram definidos para selecionar os estudos relevantes, conforme apresentado nas Tabelas 1 e 2.

Tabela 1 – Critérios de Inclusão (CI)

Código	**Descrição do Critério**
CI-01	Compara ferramentas e métodos de orquestração.
CI-02	Avalia uma ferramenta ou método de orquestração.
CI-03	Avalia o impacto da orquestração em cenários do mundo real.
CI-04	Explora conceitos relacionados à orquestração de contêineres.

Tabela 2 – Critérios de Exclusão (CE)

Código	**Descrição do Critério**
CE-01	Abaixo do percentil 80 em relevância.
CE-02	Não é sobre orquestração de contêineres.
CE-03	Não está diretamente relacionado a edge, fog ou IoT.

O uso do Percentil do Scopus como critério de exclusão fornece um método mais objetivo para filtrar artigos, reduzindo assim o tempo necessário para revisões de texto completo.

3.1.5 Verificação de Qualidade

Uma breve verificação de qualidade foi realizada nos artigos selecionados. Devido ao uso de critérios de exclusão baseados em percentil, assumimos que os artigos incluídos possuem qualidade adequada.

- As ferramentas ou metodologias de orquestração de contêineres foram avaliadas de forma clara, incluindo seus detalhes de implementação e implantação?
- Os resultados incluíram alguma análise estatística ou achados qualitativos?

As respostas possíveis e a pontuação correspondente são:

- Sim: 1 ponto
- Parcialmente: 0,5 ponto
- Não: 0 pontos

3.1.6 Visão Geral dos Estudos Selecionados

Realizamos nossa busca utilizando dois repositórios: IEEE e Scopus. No Scopus, o prefixo "*TITLE-ABS-KEY*", que significa Título, Resumo e Palavras-chave, foi adicionado ao protocolo de busca para recuperar resultados com base em palavras-chave encontradas nos títulos, resumos e palavras-chave dos artigos. Para obter um comportamento semelhante no IEEE, utilizamos a chave "*All Metadata*"para cada termo de busca. Adicionalmente, aplicamos um filtro em ambos os repositórios para limitar os resultados apenas a artigos de periódicos. A coleta de dados foi realizada em setembro de 2025.

A Figura 5 fornece um resumo visual da abordagem dos autores para a seleção, seguindo o diagrama de fluxo PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*).

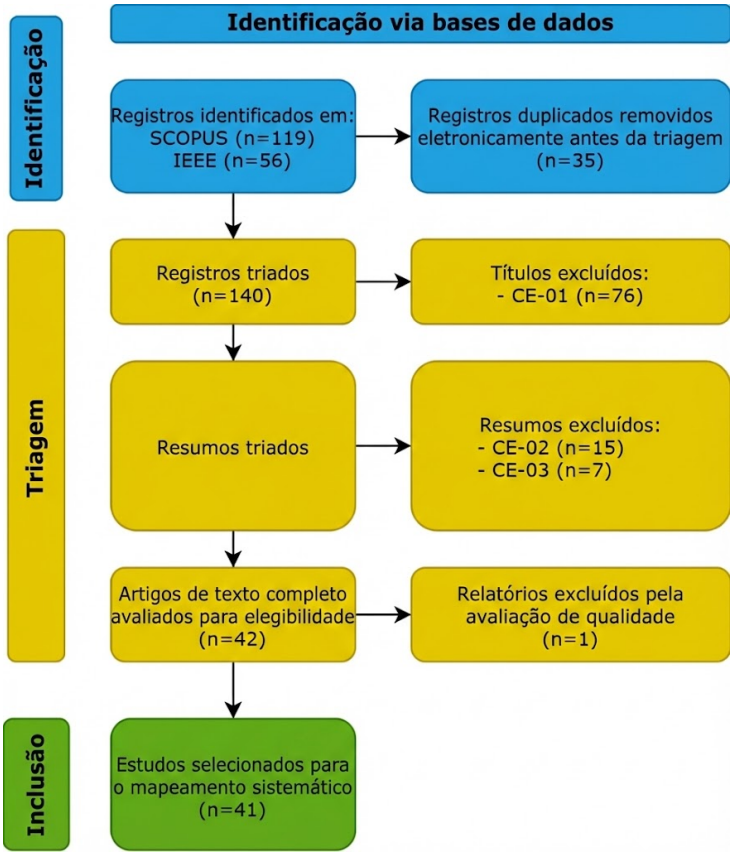


Figura 5 – Diagrama de fluxo PRISMA mostrando a metodologia usada para o mapeamento sistemático. (Fonte: Próprio Autor)

A fase inicial foca em encontrar e coletar artigos potenciais em bases de dados. A busca começou consultando duas bases de dados acadêmicas: SCOPUS, que retornou 119 registros, e IEEE, que retornou 56 registros, totalizando 175 artigos. Desse conjunto inicial, 35 registros duplicados foram identificados e removidos eletronicamente. Isso resultou em 140 artigos únicos para serem rastreados.

Na fase de rastreamento (*screening*), os 140 artigos foram avaliados em relação a critérios específicos para filtrar estudos irrelevantes. Os títulos dos 140 artigos foram rastreados com base em percentis de relevância, resultando na exclusão de 76 artigos (com base no critério de exclusão CE-01).

Os resumos dos artigos restantes foram então revisados, e 22 artigos adicionais foram excluídos: 15 foram removidos por não serem sobre orquestração de contêineres (CE-02) e 7 foram removidos por não estarem diretamente relacionados a edge, fog ou IoT (CE-03). Isso deixou 42 artigos que foram lidos na íntegra para avaliar sua elegibilidade. Durante esta verificação final, 1 relatório foi excluído com base em uma avaliação de qualidade, onde o mesmo não detalhes de implementação, implantação e resultados de forma satisfatória.

Após a conclusão de todos os rastreamentos e verificações de qualidade, um conjunto final de 41 estudos foi selecionado para ser incluído no mapeamento sistemático e na extração de

dados. A lista final de artigos selecionados é apresentada na Tabela 3.

Tabela 3 – Artigos selecionados

Ref	**Title**	**Journal**	**Year**	**Scopus Percentile**
(SOFIA et al., 2024)	A Framework for Cognitive, Decentralized Container Orchestration	IEEE Access	2024	90
(LOBATO et al., 2023)	A Monitoring System for Electric Vehicle Charging Stations: A Prototype in the Amazon	Energies	2023	80
(BLANCO et al., 2024)	A Novel Approach for Scalable and Sustainable 6G Networks	IEEE Open Journal of the Communications Society	2024	89
(ROBLES-ENCISO; SKARMETA, 2024)	Adapting Containerized Workloads for the *Continuum* Computing	IEEE Access	2024	82
(GHASEMI; SCHRANZ, 2024)	Bottom-Up Resource Orchestration in *Edge Computing*: An Agent-Based Modeling Approach	International IEEE Conference proceedings, IS	2024	84
(SUBRAMANYA; RIGGIO, 2021)	Centralized and Federated Learning for Predictive VNF Autoscaling in Multi-Domain 5G Networks and Beyond	IEEE Transactions on Network and Service Management	2021	97
(IORIO et al., 2023)	Computing Without Borders: The Way Towards Liquid Computing	IEEE Transactions on Cloud Computing	2023	92
(KAISER et al., 2022)	Container Technologies for ARM Architecture: A Comprehensive Survey of the State-of-the-Art	IEEE Access	2022	94
(RAC; BRORSSON, 2024)	Cost-Aware Service Placement and Scheduling in the Edge-Cloud Continuum	ACM Transactions on Architecture and Code Optimization	2024	85
(DALGKITSIS et al., 2021)	Data Driven Service Orchestration for Vehicular Networks	IEEE Transactions on Intelligent Transportation Systems	2021	86
(LAI; WANG; WEI, 2023)	Delay-Aware Container Scheduling in Kubernetes	IEEE *Internet of Things* Journal	2023	91
(DENG et al., 2022)	Dependent Function Embedding for Distributed Serverless *Edge Computing*	IEEE Transactions on Parallel and Distributed Systems	2022	96
(SLAMNIK-KRIJEŠTORAC et al., 2024)	Edge-Aware Cloud-Native Service for Enhancing Back Situation Awareness in 5G-Based Vehicular Systems	IEEE Transactions on Vehicular Technology	2024	83
(QIAO et al., 2024)	EdgeOptimizer: A programmable containerized scheduler of time-critical tasks in Kubernetes-based edge-cloud clusters	Future Generation Computer Systems	2024	85
(NGUYEN et al., 2020)	ElasticFog: Elastic Resource Provisioning in Container-Based Fog Computing	IEEE Access	2020	92
(QUAN; KUNDROO; KIM, 2023)	Experimental Evaluation and Analysis of Federated Learning in *Edge Computing* Environments	IEEE Access	2023	86
(GOETHALS; TURCK; VOLCKAERT, 2022)	Extending Kubernetes Clusters to Low-Resource Edge Devices Using Virtual Kubelets	IEEE Transactions on Cloud Computing	2022	93
(CHAN et al., 2022)	Implementation of a Cluster-Based Heterogeneous *Edge Computing* System for Resource Monitoring and Performance Evaluation	IEEE Access	2022	82
(CHAUDHRY et al., 2020)	Improved QoS at the Edge Using Serverless Computing to Deploy Virtual Network Functions	IEEE *Internet of Things* Journal	2020	86

Ref	Title	Journal	Year	Scopus Percentile
(CAI; BUYYA, 2022)	Inverse Queuing Model-Based Feedback Control for Elastic Container Provisioning of Web Systems in Kubernetes	IEEE Transactions on Computers	2022	83
(KAUR et al., 2020)	KEIDS: Kubernetes-Based Energy and Interference Driven Scheduler for Industrial IoT in Edge-Cloud Ecosystem	IEEE *Internet of Things* Journal	2020	97
(BORSATTI et al., 2024)	KubeTwin: A Digital Twin Framework for Kubernetes Deployments at Scale	IEEE Transactions on Network and Service Management	2024	90
(KJORVEZIROSKI; FILIPOSKA, 2022)	Kubernetes distributions for the edge: serverless performance evaluation	Journal of Supercomputing	2022	93
(NGUYEN; PHAN; KIM, 2022)	Load-Balancing of Kubernetes-Based *Edge Computing* Infrastructure Using Resource Adaptive Proxy	Sensors	2022	91
(KIM; KIM, 2023)	Local Scheduling in KubeEdge-Based *Edge Computing* Environment	Sensors	2023	94
(PALLEWATTA; KOSTAKOS; BUYYA, 2024)	MicroFog: A framework for scalable placement of microservices-based IoT applications in federated Fog environments	Journal of Systems and Software	2024	94
(LV et al., 2022)	Microservice Deployment in *Edge Computing* Based on Deep Q Learning	IEEE Transactions on Parallel and Distributed Systems	2022	97
(TUSA et al., 2024)	Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge-based real-time IoT analytics	Future Generation Computer Systems	2024	80
(LIANG et al., 2023)	Model-driven Cluster Resource Management for AI Workloads in Edge Clouds	ACM Transactions on Autonomous and Adaptive Systems	2023	95
(CHAHOUD et al., 2023)	On-Demand-FL: A Dynamic and Efficient Multicriteria Federated Learning Client Deployment Scheme	IEEE *Internet of Things* Journal	2023	93
(TRAN; KIM, 2024)	Optimized resource usage with hybrid auto-scaling system for knative serverless edge computing	Future Generation Computer Systems	2024	92
(FAYOS-JORDAN et al., 2020)	Performance comparison of container orchestration platforms with low-cost devices in the fog, assisting *Internet of Things* applications	Journal of Network and Computer Applications	2020	80
(ČILIć et al., 2023)	Performance Evaluation of Container Orchestration Tools in *Edge Computing* Environments	Sensors	2023	93
(ALMEIDA et al., 2024)	RIC-O: Efficient Placement of a Disaggregated and Distributed RAN Intelligent Controller With Dynamic Clustering of Radio Nodes	IEEE Journal on Selected Areas in Communications	2024	87
(ALASMARY, 2024)	ScalableDigitalHealth (SDH): An IoT-Based Scalable Framework for Remote Patient Monitoring	Sensors	2024	92
(BOTEZ et al., 2021)	Sdn-based network slicing mechanism for a scalable 4g/5g core network: A kubernetes approach	Sensors	2021	84
(RAHMAN et al., 2023)	Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study	ACM Transactions on Software Engineering and Methodology	2023	95
(REHAN et al., 2023)	Supply Chain Management Using an Industrial *Internet of Things* Hyperledger Fabric Network	Human-centric Computing and Information Sciences	2023	92

Ref	**Title**	**Journal**	**Year**	**Scopus Percentile**
(CENTOFANTI et al., 2024)	Taming latency at the edge: A user-aware service placement approach	Computer Networks	2024	89
(KRISTIANI et al., 2021)	The Implementation of a Cloud-*Edge Computing* Architecture Using OpenStack and Kubernetes for Air Quality Monitoring Application	Mobile Networks and Applications	2021	91
(PHUC; PHAN; KIM, 2022)	Traffic-Aware Horizontal Pod Autoscaler in Kubernetes-Based *Edge Computing* Infrastructure	IEEE Access	2022	95

3.1.7 Extração de Dados

A extração de dados será realizada para coletar informações-chave de cada estudo selecionado. Isso inclui detalhes sobre as técnicas, ferramentas, cenários de aplicação, desafios e métricas de desempenho relacionadas à orquestração de contêineres em *Edge Computing*. Os dados extraídos serão utilizados para responder às questões de pesquisa e para realizar uma análise abrangente do estado da arte neste campo.

Após a seleção dos artigos que irão compor este trabalho, foi realizada uma extração de dados para responder às questões de pesquisa previamente elaboradas. As principais informações extraídas foram:

- Plataformas de orquestração de contêineres utilizadas no estudo
- Softwares e ferramentas utilizados no estudo
- Métricas de desempenho utilizadas no estudo
- Cenário de aplicação final sugerido ou atendido
- Hardware de MEC/Borda/Dispositivo utilizado no estudo
- Principais descobertas relacionadas à orquestração de contêineres
- Onde os testes foram realizados? laboratório ou indústria?
- Percentil Scopus

3.1.8 Mapeamento Sistemático

Alinhada às diretrizes de Petersen (PETERSEN et al., 2008) para estudos de mapeamento sistemático, a análise dos resultados foca-se principalmente na apresentação das frequências de publicações para cada categoria definida. Esta abordagem é crucial para determinar a cobertura do campo de pesquisa e identificar lacunas e oportunidades para pesquisas futuras. Para este estudo, os artigos identificados foram classificados em várias facetas (categorias) predefinidas, incluindo:

- Foco da Pesquisa: abrangendo áreas como Orquestração (Orchestration), Melhorias Específicas para Kubernetes (K8s Enhance), Inteligência Artificial e Aprendizado de Máquina para Gerenciamento de Recursos (IA/ML Mgmt), Sistemas Veiculares (Vehicular), Aplicações e Sistemas IoT (IoT Apps), Virtualização de Funções de Rede/5G+ (NFV), Segurança e Confiabilidade (Security), Cadeia de Suprimentos e Industrial (Supply Chain), e Levantamentos e Metodologias (Surveys).
- Contribuição: categorizando a natureza do resultado da pesquisa, como Projeto de Framework/Arquitetura (Frm), Proposta de Algoritmo/Política (Alg), Avaliação Empírica/Estudo de Caso (Eval), Implementação de Ferramenta/Sistema (Tool), Modelagem Teórica/Formalização (Thr), Levantamento/Revisão da Literatura (Surv) e Análise Arquitetural (Arch).
- Objetivo de Otimização: destacando os objetivos das soluções propostas, incluindo Latência/Tempo de Resposta (Lat), Eficiência/Utilização de Recursos (Res), Otimização de Custo (Cos), Escalabilidade/Adaptabilidade (Sca), Aderência a QoS/SLA (QoS), Eficiência Energética/Sustentabilidade (Ene), Segurança/Privacidade (Sec), Gerenciamento de Dados / Observabilidade (Obs) e Confiabilidade/Tolerância a Falhas (Rel).

Ao analisar a distribuição e as frequências dos artigos nessas categorias, de forma semelhante ao uso de facetas de tópico, tipo de contribuição e abordagem de pesquisa por Petersen, este mapa sistemático identifica quais áreas foram enfatizadas em pesquisas passadas e onde pode haver escassez de publicações.

Enquanto a Tabela 4 classifica cada estudo com base nas facetas previamente estabelecidas, a Figura 6 relata visualmente a frequência cruzada das facetas empregando gráficos de bolhas, o que apoia melhor a análise ao permitir que diferentes facetas sejam consideradas simultaneamente. Este método de visualização pode fornecer uma visão geral mais rápida do campo, mostrando a interseção das categorias e o número de artigos correspondentes a essas interseções. A discussão subsequente na Seção IV aprofunda-se nesses achados categorizados, abordando diretamente as questões de pesquisa do estudo.

A análise do gráfico de bolhas, visualizada em dois subgráficos, revela um cenário de pesquisa predominantemente impulsionado por considerações práticas. "Latência"e "Eficiência de Recursos"emergem consistentemente como os objetivos de otimização mais frequentemente abordados em quase todas as áreas de foco de pesquisa, especialmente em Orquestração e Melhorias para K8s, ressaltando a necessidade crítica de sistemas responsivos e eficientes.

Tabela 4 – Mapeamento Sistemáticos nas facetas (categorias) definidas

Ref	Research Focus	Contribution	Optimization Goal
(PALLEWATTA; KOSTAKOS; BUYYA, 2024)	Orchestration	Frm, Alg, Eval	Lat, Sca, Res, QoS
(KRISTIANI et al., 2021)	IoT Apps	Tool, Eval	Lat, Res, Rel
(TRAN; KIM, 2024)	K8s Enhance	Frm, Alg, Eval	Lat, Res
(TUSA et al., 2024)	IoT Apps	Eval	Lat, Res
(QIAO et al., 2024)	K8s Enhance	Tool, Alg, Eval	Lat, Sca, Res
(CENTOFANTI et al., 2024)	Orchestration	Alg, Eval	Lat, Res
(RAHMAN et al., 2023)	Security	Eval, Tool	Sec
(REHAN et al., 2023)	Supply Chain	Frm	N/A
(LIANG et al., 2023)	AI/ML Mgmt	Thr, Alg, Eval	Lat, Res
(RAC; BRORSSON, 2024)	Orchestration	Alg, Eval	Cos, Lat
(SOFIA et al., 2024)	Orchestration	Frm, Alg, Thr	Sca, Ene, Lat, QoS, Obs
(BLANCO et al., 2024)	NFV	Frm, Alg, Eval	Sca, Ene, QoS
(ROBLES-ENCISO; SKARMETA, 2024)	K8s Enhance	Frm, Alg, Eval	Res, Sca
(GHASEMI; SCHRANZ, 2024)	Orchestration	Alg, Thr, Eval	Res, QoS
(SUBRAMANYA; RIGGIO, 2021)	AI/ML Mgmt, NFV	Alg, Thr, Eval	QoS, Cos, Sec
(IORIO et al., 2023)	Orchestration	Frm, Eval	Sca, Lat
(KAISER et al., 2022)	Surveys	Surv	N/A
(DALGKITSIS et al., 2021)	Vehicular	Alg, Eval	Lat, Res
(LAI; WANG; WEI, 2023)	K8s Enhance	Alg, Eval	Lat, Res
(DENG et al., 2022)	Orchestration	Alg, Thr, Eval	Lat
(SLAMNIK-KRIJEšTORAC et al., 2024)	Vehicular	Tool, Eval	Lat, Res
(NGUYEN et al., 2020)	IoT Apps	Alg	Res, QoS
(QUAN; KUNDROO; KIM, 2023)	AI/ML Mgmt	Eval	Sca, Sec
(GOETHALS; TURCK; VOLCKAERT, 2022)	K8s Enhance	Frm, Eval	Sca, Cos
(CHAN et al., 2022)	IoT Apps	Tool, Eval	Res, Obs
(CHAUDHRY et al., 2020)	NFV	Frm, Eval	QoS, Res
(CAI; BUYYA, 2022)	AI/ML Mgmt	Alg, Thr, Eval	QoS, Cos
(KAUR et al., 2020)	Supply Chain	Alg, Eval	Ene, Res
(BORSATTI et al., 2024)	K8s Enhance	Tool, Thr	Sca
(LV et al., 2022)	AI/ML Mgmt	Alg, Thr, Eval	Lat, Res, Sca
(CHAHOUD et al., 2023)	AI/ML Mgmt	Alg, Thr, Eval	Sca, Obs
(ALMEIDA et al., 2024)	NFV	Alg, Thr, Eval, Tool	Cos, Lat
(PHUC; PHAN; KIM, 2022)	K8s Enhance	Alg, Eval	Lat, Res
(LOBATO et al., 2023)	Supply Chain	Surv	Sca
(FAYOS-JORDAN et al., 2020)	Orchestration	Eval	Res
(KJORVEZIROSKI; FILIPOSKA, 2022)	K8s Enhance	Eval	Lat
(BOTEZ et al., 2021)	NFV	Alg, Eval	Res, Sca, Lat
(NGUYEN; PHAN; KIM, 2022)	K8s Enhance	Alg	Lat, Res
(KIM; KIM, 2023)	K8s Enhance	Eval	Lat, Res
(ČILIć et al., 2023)	Surveys	Surv, Arch, Eval	N/A
(ALASMARY, 2024)	IoT Apps	Frm, Eval	Lat, Sca, Cos

Correspondentemente, "Avaliação"e "Algoritmos"são os tipos de contribuição mais prevalentes, complementados por "Frameworks"e "Ferramentas", destacando uma forte abordagem empírica e orientada a soluções na literatura. Embora Escalabilidade e Qualidade de Serviço também apareçam significativamente, particularmente em domínios de sistemas dinâmicos, objetivos de otimização menos frequentes como "Eficiência Energética", "Gerenciamento de Dados/Observabilidade"e "Confiabilidade"parecem ser mais de nicho, sugerindo áreas potenciais para pesquisas futuras ou indicando que esses aspectos podem ser abordados implicitamente dentro de objetivos de desempenho mais amplos. No geral, os gráficos ilustram coletivamente uma trajetória de pesquisa robusta e orientada ao desempenho, enfatizando a validação empírica e a aplicação prática nos domínios de computação em nuvem nativa e de borda.

3.1.9 Questões de Pesquisa

Esta seção apresenta a análise detalhada dos achados do mapeamento sistemático, estruturada para responder diretamente às seis Questões de Pesquisa definidas na metodologia. Cada subseção a seguir é dedicada a uma QP específica, sintetizando os dados extraídos dos 41 artigos selecionados. O objetivo é consolidar o estado da arte e identificar os principais desafios, abordagens, ferramentas e a maturidade da pesquisa em orquestração de contêineres aplicada a ambientes de borda

QP1: Quais são as principais áreas de pesquisa e desafios abordados na literatura sobre orquestração de contêineres especificamente em ambientes de borda, névoa ou contínuo?

A pesquisa sobre orquestração de contêineres em ambientes de borda, névoa e contínuo foca-se principalmente na adaptação e otimização de plataformas existentes, como Kubernetes e Docker Swarm, para esses cenários distribuídos e com recursos restritos (ČILIć et al., 2023). Uma área central de pesquisa é a otimização de agendamento e posicionamento, com esforço significativo direcionado ao desenvolvimento de algoritmos de agendamento cientes da latência que priorizam a minimização da latência percebida pelo usuário final e a latência de ponta a ponta, o que muitas vezes não é totalmente resolvido ao focar apenas na latência entre serviços dentro de um cluster (CENTOFANTI et al., 2024).

Os pesquisadores também estão desenvolvendo estratégias para gerenciamento de recursos e alocação de cargas de trabalho que levam em conta a natureza heterogênea dos nós de borda (ROBLES-ENCISO; SKARMETA, 2024), e explorando a orquestração de cargas de trabalho específicas, como microsserviços, funções serverless, IA/ML distribuída (ex: Aprendizado Federado) e Funções de Rede Virtualizadas (VNFs) nesses ambientes.

Apesar do trabalho fundamental, vários desafios persistem. As restrições de recursos e a heterogeneidade dos dispositivos de borda em termos de CPU, memória, largura de banda e capacidades de hardware representam obstáculos significativos, já que as ferramentas de orquestração tradicionais podem consumir muitos recursos (FAYOS-JORDAN et al., 2020; GOETHALS; TURCK; VOLCKAERT, 2022). A complexidade do agendamento aumenta devido à necessidade de otimização dinâmica, ciente do contexto e, muitas vezes, multiobjetivo, considerando fatores além da mera disponibilidade de recursos, como condições de rede, mobilidade, confiabilidade e eficiência energética (SOFIA et al., 2024; KAUR et al., 2020).

Minimizar a latência do usuário final e lidar com a variabilidade da percepção da rede e da mobilidade dos nós continuam sendo questões em aberto (RAC; BRORSSON, 2024). A escalabilidade e a elasticidade são desafiadoras em ambientes de borda altamente distribuídos e dinâmicos, questionando a eficácia de métodos de autoescalonamento padrão baseados na nuvem, como o *Horizontal Pod Autoscaler* (HPA) (FAYOS-JORDAN et al., 2020; ALASMARY, 2024). Garantir a confiabilidade e a robustez contra falhas de nós e mobilidade inesperada é crucial (LV

et al., 2022; NGUYEN et al., 2022). A eficiência energética e as más configurações de segurança também são desafios observados (KAUR et al., 2020; RAHMAN et al., 2023). Além disso, percebe-se uma falta de avaliação e testes rigorosos das soluções propostas usando ferramentas de orquestração reais em ambientes de borda do mundo real, com muitos estudos dependendo de simulações. Desafios técnicos específicos relacionados à otimização do desempenho em arquiteturas ARM e outro hardware específico de borda também necessitam de mais atenção (ČILIć et al., 2023).

QP2: Quais abordagens, métodos ou algoritmos são propostos para a orquestração de contêineres na borda

A orquestração de contêineres gerencia a implantação, a rede e o ciclo de vida de microsserviços, garantindo escalabilidade e confiabilidade. Um componente crítico é o agendamento, que mapeia serviços para os nós ideais, maximizando a utilização de recursos e o desempenho. Na borda, a heterogeneidade dos nós (variações de computação, memória e largura de banda) desafia os agendadores padrão do Kubernetes, projetados para nuvens homogêneas (LAI; WANG; WEI, 2023). Soluções como o algoritmo *Delay-Aware Container Scheduling* (DACS) levam em conta os recursos residuais e os atrasos de processamento/rede para melhorar o desempenho (LAI; WANG; WEI, 2023).

Os balanceadores de carga padrão frequentemente ignoram a localização ou o status dos recursos, aumentando a latência (PHUC; PHAN; KIM, 2022; NGUYEN; PHAN; KIM, 2022). Abordagens adaptativas como o *ElasticFog* otimizam a alocação de *pods* com base na distribuição de tráfego (NGUYEN et al., 2020), enquanto o *Resource Adaptive Proxy* (RAP) processa requisições localmente ou as encaminha para nós ideais (NGUYEN; PHAN; KIM, 2022). O agendamento local (por exemplo, no KubeEdge) reduz a latência ao priorizar o processamento no próprio nó (KIM; KIM, 2023).

O posicionamento de serviços deve atender aos SLAs, mapeando as demandas de recursos para os nós disponíveis, garantindo ao mesmo tempo a QoS (QIAO et al., 2024). A latência de ponta a ponta depende fortemente da localização dos microsserviços. Agendadores Cientes da Latência priorizam nós com o mínimo de atraso percebido pelo usuário (CENTOFANTI et al., 2024), enquanto soluções da borda para a nuvem combinam um posicionamento inicial (otimizado para custo/latência) com reagendamento (por exemplo, para mobilidade do usuário) (RAC; BRORSSON, 2024). A orquestração ajusta periodicamente os serviços com base na distribuição de requisições e nas restrições do cluster (QIAO et al., 2024).

Ambientes de borda exigem o uso eficiente de recursos devido à sua escassez (GHASEMI; SCHRANZ, 2024). As técnicas incluem multiplexação de hardware (LIANG et al., 2023), integração de dispositivos IoT e isolamento para mitigar interferências. Algoritmos de alocação dinâmica preveem as necessidades para minimizar o desperdício (NGUYEN et al., 2020), enquanto *frameworks* cientes de QoS aplicam políticas de largura de banda (por exemplo,

compartilhamento proporcional, limites mín/máx) (TRAN; KIM, 2024).

Algoritmos de provisionamento de contêineres (ex: *auto-scaling*) são vitais para a QoS. Embora o HPA do Kubernetes enfrente dificuldades com a heterogeneidade da borda (ALASMARY, 2024), alternativas como:

- Controle por Feedback Baseado em Modelo de Fila Inversa reduzem violações de SLA (CAI; BUYYA, 2022),
- HPA ciente do tráfego (THPA) otimiza a vazão (*throughput*) adaptando-se ao tráfego (PHUC; PHAN; KIM, 2022),
- *Auto-scaling* híbrido combina métodos verticais (ajuste de recursos) e horizontais (ajuste de instâncias) (TRAN; KIM, 2024; GHASEMI; SCHRANZ, 2024).

Várias técnicas de IA/ML são empregadas na orquestração de contêineres, agendamento, auto-scaling e gerenciamento de recursos, particularmente em ambientes de borda nativos da nuvem:

- Aprendizado por Reforço (RL/DRL/DQL): Otimiza o agendamento (SOFIA et al., 2024), a alocação de recursos (TRAN; KIM, 2024; KAUR et al., 2020), e a implantação de microsserviços (LV et al., 2022).
- Modelos Preditivos: Preveem a demanda para auto-scaling proativo (SUBRAMANYA; RIGGIO, 2021; CAI; BUYYA, 2022).
- Aprendizado Federado (FL): Permite a orquestração descentralizada e com preservação de privacidade (QUAN; KUNDROO; KIM, 2023; CHAHOUD et al., 2023).
- Meta-heurísticas (GA, ACO): Resolvem problemas NP-difíceis como o descarregamento de tarefas (*task offloading*) (DALGKITSIS et al., 2021; NGUYEN et al., 2022) ou gerenciamento de recursos inspirado em enxames (CHAN et al., 2022).
- Técnicas Emergentes: Aprendizado Dividido (*Split Learning*) e GNNs permitem decisões descentralizadas e cientes do contexto (SOFIA et al., 2024).

Esses métodos visam otimizar a latência, o uso de recursos, a eficiência energética e a confiabilidade em sistemas distribuídos (NGUYEN; PHAN; KIM, 2022; PéREZ et al., 2023).

QP3: Quais são as limitações, questões em aberto ou direções futuras de pesquisa identificadas em relação à orquestração de contêineres e sua aplicação na borda ou em todo o contínuo da computação?

Com base nos artigos selecionados, os *frameworks* de orquestração de contêineres existentes enfrentam limitações significativas quando aplicados à borda ou em todo o contínuo da

computação, principalmente porque foram projetados para arquiteturas de nuvem homogêneas, em vez da natureza heterogênea, com recursos restritos e geograficamente distribuída dos ambientes de borda (ROBLES-ENCISO; SKARMETA, 2024). Uma limitação fundamental é sua inadequação para dispositivos de borda de baixos recursos devido aos altos requisitos de recursos (ČILIć et al., 2023). As abordagens de agendamento padrão, como espalhar contêineres ou selecionar o servidor menos alocado, muitas vezes não são aplicáveis ou eficientes para nós de borda, onde fatores como localização geográfica e características de rede são críticos para alcançar baixa latência (RAC; BRORSSON, 2024; CAI; BUYYA, 2022; NGUYEN et al., 2022). Além disso, o processo de alocação um-para-um de pod para nó em ferramentas como o Kubernetes limita a capacidade de otimizar o conjunto completo de implantações em todo o contínuo. Os *frameworks* existentes também têm limitações em áreas como o posicionamento multi-névoa e multi-nuvem, implantação totalmente automatizada e composição dinâmica entre clusters (PALLEWATTA; KOSTAKOS; BUYYA, 2024). Há uma necessidade de consciência de contexto e capacidades cognitivas para lidar com desafios de infraestrutura, como conectividade intermitente e falha de nós (SOFIA et al., 2024). Outras questões em aberto incluem a complexidade da implantação e gerenciamento nesses ambientes, tornando as simulações muitas vezes mais convenientes do que as avaliações do mundo real, e a dificuldade em integrar lógicas de agendamento personalizadas e rastrear parâmetros dinâmicos de QoS externamente (ČILIć et al., 2023). Abordar esses desafios é essencial para construir um *framework* de borda resiliente (ALASMARY, 2024).

Consequentemente, as direções futuras de pesquisa focam em superar essas limitações e adaptar melhor a orquestração para o contínuo borda-nuvem. Isso inclui o desenvolvimento de novos *frameworks* projetados especificamente para este ambiente (PALLEWATTA; KOSTAKOS; BUYYA, 2024) e a proposição de soluções para adaptar orquestradores existentes como o Kubernetes, modificando ou estendendo seus componentes e funcionalidades centrais (ROBLES-ENCISO; SKARMETA, 2024). Uma área importante de pesquisa é o projeto e a implementação de algoritmos complexos de agendamento e alocação para o Problema de Atribuição de Tarefas, que precisa considerar uma vasta gama de características heterogêneas dos nós, como poder de computação, aceleradores de hardware, localização, latência, largura de banda, consumo de energia e custo (LAI; WANG; WEI, 2023; ROBLES-ENCISO; SKARMETA, 2024). Técnicas algorítmicas avançadas, incluindo várias meta-heurísticas (Algoritmos Genéticos, PSO, Otimização por Colônia de Formigas), Aprendizado por Reforço, Redes Neurais e abordagens multiobjetivo, estão sendo exploradas para esses problemas complexos (LV et al., 2022; CHAHOUD et al., 2023; KIM et al., 2023; ALMEIDA et al., 2024). Trabalhos futuros também envolvem a otimização de algoritmos de *auto-scaling* de contêineres, potencialmente incorporando a distribuição geográfica, e a pesquisa sobre otimização de contêineres para arquiteturas específicas como ARM (*Advanced RISC Machine*) é uma arquitetura de processador baseada no conceito de RISC (*Reduced Instruction Set Computing*), que visa eficiência e menor consumo de energia. Investigar alternativas aos contêineres, como Unikernels em ARM, também é um problema de pesquisa em aberto (KAISER et al., 2022). Esforços estão sendo direcionados para o aproveitamento de

abordagens de IA/ML distribuídas para gerenciamento e orquestração automatizados (SOFIA et al., 2024; ALMEIDA et al., 2024; BLANCO et al., 2024), e a implementação de serviços externos para coletar dados de QoS em tempo real para permitir o reagendamento dinâmico (ČILIć et al., 2023). No geral, há um forte impulso para a realização de mais avaliações experimentais em ambientes de borda dinâmicos e do mundo real para validar as soluções propostas para além das simulações. A segurança também é uma área que requer mais pesquisa, particularmente em relação a más configurações em plataformas como o Kubernetes (RAHMAN et al., 2023).

QP4: Quais ferramentas de orquestração de contêineres e tecnologias de contêineres subjacentes são investigadas ou utilizadas na pesquisa sobre borda?

O Kubernetes surge como a ferramenta mais proeminente, reconhecida como um padrão de fato para a orquestração de contêineres (ROBLES-ENCISO; SKARMETA, 2024; IORIO et al., 2023; PHUC; PHAN; KIM, 2022; NGUYEN; PHAN; KIM, 2022). É amplamente explorado para gerenciar aplicações conteinerizadas e a implantação de microsserviços em ambientes de borda, névoa e multi-nuvem. A Figura 8 representa visualmente a distribuição das plataformas de orquestração de contêineres identificadas e utilizadas nos estudos selecionados, ressaltando a prevalência de várias ferramentas, particularmente o Kubernetes e suas distribuições, em ambientes de borda.

A Figura 7 fornece um resumo visual que organiza as diferentes abordagens e ferramentas de orquestração de contêineres identificadas na pesquisa sobre borda. O diagrama ilustra uma divisão principal no campo: de um lado, as "Ferramentas de Cloud Existentes - Adaptadas", que inclui plataformas robustas de nuvem como o Kubernetes (K8s) e o Docker Swarm, que são adaptadas para o cenário de borda. Do outro lado, estão as "Soluções com Foco em Borda", que são projetadas especificamente para os desafios de ambientes com recursos restritos. Esta segunda categoria se subdivide em "Distribuições Leves"— que são versões otimizadas e enxutas do Kubernetes, como K3s, K0s, MicroK8s e KubeEdge — e "Soluções e *Frameworks*"personalizados, que englobam ferramentas customizadas e propostas acadêmicas como MicroFog, EdgeOptimizer e KubeTwin.

Distribuições leves de Kubernetes, como K3s, MicroK8s, K0s e KubeEdge, são versões do K8s que foram modificadas ou otimizadas para operar eficientemente em ambientes com recursos restritos, como dispositivos de borda, nós de IoT e clusters de pequena escala (KJORVEZIROSKI; FILIPOSKA, 2022). Essas distribuições são projetadas para reduzir as altas demandas de recursos de uma instalação completa do Kubernetes, tornando a orquestração de contêineres mais viável em ambientes com CPU, memória e armazenamento limitados.

O Docker Swarm também é discutido e avaliado, particularmente no contexto de computação em névoa e SBCs de baixo custo, e observa-se que supera o Kubernetes em certos cenários (FAYOS-JORDAN et al., 2020). No entanto, o Docker Swarm também é identificado como carente de recursos nativos como *auto-scaling* e balanceamento de carga

(QUAN; KUNDROO; KIM, 2023) e não está mais sendo desenvolvido ou suportado ativamente. O Eclipse ioFog é considerado uma ferramenta de orquestração orientada para a borda, mas uma avaliação concluiu que ele é inadequado para ambientes de borda dinâmicos devido ao alto tempo de implantação e ao consumo de memória (ČILIć et al., 2023). Além destes, os pesquisadores propõem ou implementam *frameworks* de orquestração e soluções de agendamento personalizados para o contínuo borda-nuvem, a fim de abordar as limitações das ferramentas genéricas (GOETHALS; TURCK; VOLCKAERT, 2022; PALLEWATTA; KOSTAKOS; BUYYA, 2024; QIAO et al., 2024; LIANG et al., 2023; NGUYEN et al., 2020; NGUYEN et al., 2022; MOON et al., 2024). As tecnologias de contêineres subjacentes frequentemente mencionadas incluem Docker, amplamente utilizado para empacotar aplicações, Linux Containers, Containerd e CRI-O como runtimes de contêiner que implementam a Container Runtime Interface (CRI), e Podman como uma alternativa sem daemon. Os Unikernels também são explorados como potenciais alternativas aos contêineres por seus benefícios de segurança e eficiência (KAISER et al., 2022).

Além das principais plataformas de orquestração, as fontes mencionam várias outras ferramentas e *frameworks* que desempenham um papel no gerenciamento de aplicações conteinerizadas na borda ou que fazem parte do ecossistema mais amplo da borda. Estes incluem:

- Ferramentas de Monitoramento e Observabilidade:
  - Prometheus e Grafana: Frequentemente usados em conjunto para monitoramento e visualização de recursos em vários estudos envolvendo implantações de borda baseadas em Kubernetes (PALLEWATTA; KOSTAKOS; BUYYA, 2024; SUBRAMANYA; RIGGIO, 2021; BLANCO et al., 2024; LAI; WANG; WEI, 2023; KAUR et al., 2020; CHAN et al., 2022; ALMEIDA et al., 2024; KJORVEZIROSKI; FILIPOSKA, 2022; BOTEZ et al., 2021; PéREZ et al., 2023; ALASMARY, 2024).
  - Swarmprom e o *framework* de monitoramento Lattice: Ferramentas de monitoramento específicas mencionadas no contexto do Docker Swarm (LV et al., 2022).
  - Ganglia Monitoring System: Usado em uma arquitetura de nuvem-borda para monitoramento da qualidade do ar (KRISTIANI et al., 2021).
- Plataformas de Virtualização e Emulação:
  - OpenStack: Empregado em arquiteturas de nuvem-borda, muitas vezes ao lado do Kubernetes, para gerenciar recursos virtualizados (PALLEWATTA; KOSTAKOS; BUYYA, 2024; NGUYEN et al., 2020; KRISTIANI et al., 2021; BOTEZ et al., 2021; KAUR et al., 2020).
  - VMware Workstation e Proxmox: Usados para criar ambientes virtualizados para avaliações experimentais de agendamento e orquestração de contêineres (LAI; WANG; WEI, 2023; QIAO et al., 2024).

- Ferramentas de Rede e Conectividade:
  - Wireguard: Uma solução de VPN utilizada para dar suporte a nós em redes privadas (IORIO et al., 2023; GOETHALS; TURCK; VOLCKAERT, 2022).
  - OpenFlow: Um protocolo Software-Defined Networking (SDN) usado em *frameworks* para distribuição confiável de energia em microrredes e fatiamento de rede (network slicing) (KRISTIANI et al., 2021; BOTEZ et al., 2021; PéREZ et al., 2023).
  - Calico: Um plugin de Interface de Rede de Contêiner (CNI) usado para redes em clusters Kubernetes (PéREZ et al., 2023; SOFIA et al., 2024).
  - Multus: Um plugin CNI para habilitar múltiplas interfaces de rede em *pods*, usado em fatiamento de rede baseado em SDN (BOTEZ et al., 2021).
  - Open vSwitch (OvS): Um componente de rede definida por software usado em mecanismos de fatiamento de rede (KRISTIANI et al., 2021; BOTEZ et al., 2021; SUBRAMANYA; RIGGIO, 2021; PéREZ et al., 2023).

QP5: Quais domínios de aplicação são abordados ou aproveitados pela pesquisa focada na orquestração de contêineres em borda?

A pesquisa focada na orquestração de contêineres na borda e em todo o contínuo da computação aborda e aproveita uma vasta gama de domínios de aplicação, impulsionados principalmente pelo número crescente de dispositivos de IoT e pela necessidade de processamento de baixa latência e em tempo real (PALLEWATTA; KOSTAKOS; BUYYA, 2024; TUSA et al., 2024; QIAO et al., 2024). Proeminentes entre estes estão as aplicações em cidades inteligentes, que envolvem o gerenciamento de enormes fluxos de dados para planejamento urbano, otimização de tráfego e vigilância (GHASEMI; SCHRANZ, 2024; NGUYEN; PHAN; KIM, 2022; SOFIA et al., 2024).

A Figura 9 representa visualmente a vasta gama de domínios de aplicação que são abordados ou aproveitados pela pesquisa focada na orquestração de contêineres na borda. Esses domínios são impulsionados principalmente pelo número crescente de dispositivos de IoT e pela necessidade crítica de processamento de baixa latência e em tempo real. O domínio da saúde, abrangendo saúde inteligente, monitoramento remoto de pacientes, telemedicina e a Internet das Coisas Médicas, também é amplamente abordado por suas necessidades de processamento crítico de latência e privacidade de dados (KRISTIANI et al., 2021; CHAUDHRY et al., 2020; KIM et al., 2023; QUAN; KUNDROO; KIM, 2023). A IoT Industrial e a manufatura inteligente (Indústria 4.0), incluindo fábricas inteligentes e detecção de falhas de equipamentos, beneficiam-se significativamente da orquestração de contêineres na borda para análises em tempo real, minimização de energia e manuseio eficiente de interferências (CHAN et al., 2022; CHAUDHRY et al., 2020; PHUC; PHAN; KIM, 2022; KIM et al., 2023).

Além dessas áreas centrais, a pesquisa explora o papel da orquestração de contêineres em diversas aplicações, como transporte inteligente e redes veiculares, incluindo gêmeos digitais veiculares, comunicação *Vehicle-to-Everything* (V2X), veículos autônomos, carros sem motorista e o aproveitamento de veículos estacionados como recursos computacionais (SLAMNIK-KRIJEŠTORAC et al., 2024; BORSATTI et al., 2024; CHAHOUD et al., 2023; NGUYEN et al., 2022; MOON et al., 2024). Aplicações de IA e aprendizado de máquina, muitas vezes integradas a outros domínios para tarefas como análise visual, detecção de objetos e reconhecimento de imagem, são um foco principal, com a orquestração de contêineres facilitando sua implantação e gerenciamento em todo o contínuo (TRAN; KIM, 2024; LIANG et al., 2023; BLANCO et al., 2024; SUBRAMANYA; RIGGIO, 2021; KAISER et al., 2022; DENG et al., 2022; QUAN; KUNDROO; KIM, 2023; ALMEIDA et al., 2024). Outros domínios abordados incluem realidade aumentada e virtual (KRISTIANI et al., 2021), robótica, particularmente robôs autônomos e robótica em nuvem (BORSATTI et al., 2024), energia e redes elétricas inteligentes, edifícios inteligentes, agricultura inteligente, vigilância inteligente e construção inteligente. Esses diversos domínios de aplicação destacam a ampla aplicabilidade e a necessidade crítica de estratégias eficazes de orquestração de contêineres adaptadas às características únicas dos ambientes de borda.

QP6: Qual o nível de maturidade dos métodos de avaliação e ambientes de teste para as soluções de orquestração de contêineres propostas na borda?

Os métodos de avaliação e a maturidade dos ambientes de teste podem ser categorizados da seguinte forma (ver Figura 10): Prevalência de Simulação e Laboratórios (Baixa Maturidade): Um número significativo de estudos depende de ambientes simulados (mais de 60%) ou laboratórios controlados com placas de desenvolvimento como Raspberry Pi e Jetson Nano. Isso indica que grande parte da pesquisa é exploratória, focando no projeto algorítmico e na validação conceitual, em vez da prontidão para produção.

Uma porção notável do trabalho progrediu para o nível de maturidade Médio. Esses estudos utilizam bancadas de teste (*testbeds*) estruturadas, muitas vezes com uma mistura de hardware (incluindo equipamentos de rede comerciais ou servidores), ou emulam cenários complexos e distribuídos. Isso mostra uma clara tendência em direção a uma prototipagem mais realista e robusta.

Apenas alguns estudos (menos de 10%) alcançaram um nível de maturidade Alto, com testes conduzidos em ambientes do mundo real, industriais ou semelhantes aos de produção (por exemplo, uma rodovia inteligente, uma rede física de carregamento de veículos elétricos). Esses estudos são cruciais, pois fornecem validação da aplicabilidade prática e do desempenho das tecnologias fora de um ambiente controlado.

Em conclusão, embora haja uma enorme quantidade de atividade acadêmica e de pesquisa neste campo, a transição de conceitos de laboratório para soluções endurecidas industrialmente e

implantadas ainda está em seus estágios iniciais. Uma visão detalhada da análise realizada para chegar a esta avaliação pode ser vista na Tabela 5.

Tabela 5 – Maturidade Avaliada

Ref	**Ambiente de Teste**	**Hardware Utilizado**	**Maturidade Tecnológica Avaliada**	**Justificativa**
(ALASMARY, 2024)	Laboratório controlado	Arduino, ESP32, vários sensores	Baixa	Ambiente de laboratório com hardware de prototipagem/desenvolvimento.
(ČILIć et al., 2023)	Laboratório/pesquisa	Raspberry Pi 4	Baixa	Ambiente de laboratório com um único tipo de placa de desenvolvimento.
(QIAO et al., 2024)	Laboratório	Simulado	Baixa	Estudo puramente baseado em simulação.
(RAC; BRORSSON, 2024)	Cluster Kubernetes real (nuvem pública)	Simulado (VMs emulando a borda)	Média	Usa um cluster real, mas emula a topologia de borda; um passo além da simulação pura.
(LIANG et al., 2023)	Laboratório/pesquisa	Jetson Nano, USB edgeTPU	Baixa	Ambiente de laboratório usando placas de desenvolvimento especializadas para IA.
(KIM; KIM, 2023)	Laboratório/pesquisa	Simulado	Baixa	Estudo puramente baseado em simulação.
(QUAN; KUNDROO; KIM, 2023)	Plataforma de testes (testbed) experimental	Nós abstratos de nuvem/borda	Média	Uma "plataforma de testes experimental"estruturada implica uma configuração mais avançada do que um laboratório básico.
(LV et al., 2022)	Controlado/simulado (virtualização)	RK3399 (SoC)	Baixa	Ambiente simulado, mesmo com um SoC específico mencionado.
(NGUYEN; PHAN; KIM, 2022)	Ambiente de borda experimental	Especificações de servidor genéricas (provavelmente VMs)	Média	Ambiente experimental usando recursos de classe de servidor (VMs), não placas de desenvolvimento.
(PHUC; PHAN; KIM, 2022)	Laboratório	Simulado	Baixa	Estudo puramente baseado em simulação.
(KRISTIANI et al., 2021)	Plataforma de testes (testbed) acadêmica	Arduino, Raspberry Pi 3	Baixa	Plataforma de testes acadêmica usando placas de desenvolvimento.
(FAYOS-JORDAN et al., 2020)	Plataforma de testes (testbed) customizada	RPi, Tinker, Udoo, Jetson, Cisco Switch	Média	Plataforma de testes construída especificamente com hardware diverso, incluindo equipamento de rede comercial.
(KAUR et al., 2020)	Ambiente simulado	Simulado	Baixa	Estudo puramente baseado em simulação.
(NGUYEN et al., 2020)	Laboratório controlado (Névoa emulada)	Especificações de servidor genéricas (provavelmente VMs)	Média	Ambiente de laboratório projetado para emular um cenário distribuído com nós de classe de servidor.

Ref	Ambiente de Teste	Hardware Utilizado	Maturidade Tecnológica Avaliada	Justificativa
(CHAUDHRY et al., 2020)	Laboratório acadêmico	NetFPGA-10G, Raspberry Pi, Ixia XM2	Média	Uso de hardware especializado (NetFPGA) e de teste profissional (Ixia) eleva o nível.
(CHAHOUD et al., 2023)	Simulações	Simulado	Baixa	Estudo puramente baseado em simulação.
(KAISER et al., 2022)	Laboratório (para testes dos autores)	Raspberry Pi 4	Baixa	Artigo de revisão (survey); seus próprios experimentos são baseados em laboratório com uma placa de desenvolvimento.
(LAI; WANG; WEI, 2023)	Laboratório simulado	Máquinas Virtuais (VMware)	Baixa	Um ambiente simulado executado em VMs em um laboratório.
(GOETHALS; TURCK; VOLCKAERT, 2022)	Plataforma de testes (testbed) experimental	Servidor (x64) e Raspberry Pi 3	Média	Plataforma de testes projetada para validar entre arquiteturas de servidor e de dispositivos de baixos recursos.
(BORSATTI et al., 2024)	Controlado e simulado	Simulado	Baixa	Estudo puramente baseado em simulação.
(DALGKITSIS et al., 2021)	Laboratório	Servidores MEC Simulados	Baixa	Simulação em ambiente de laboratório.
(SUBRAMANYA; RIGGIO, 2021)	Pesquisa (simulação + protótipo)	Simulado	Baixa	Primariamente simulação, apesar de um componente de protótipo.
(ALMEIDA et al., 2024)	Laboratório	Simulado	Baixa	Simulação em ambiente de laboratório.
(ROBLES-ENCISO; SKARMETA, 2024)	Laboratório	Simulado	Baixa	Simulação em ambiente de laboratório.
(SLAMNIK-KRIJEšTORAC et al., 2024)	Real/industrial (Rodovia Inteligente)	Unidades de Beira de Estrada (RSUs)	Alta	Testado em um ambiente industrial do mundo real (rodovia E313, Bélgica).
(BLANCO et al., 2024)	Laboratório	Amarisoft Callbox gNBs	Média	Ambiente de laboratório usando hardware de teste 5G/6G especializado, quase comercial.
(RAHMAN et al., 2023)	Indústria de software de código aberto	Não Aplicável	N/A	Estudo empírico de código existente, não um teste de implementação de tecnologia.
(CAI; BUYYA, 2022)	Laboratório de pesquisa	Quatro máquinas físicas	Média	Laboratório de pesquisa usando um cluster de máquinas físicas, não apenas VMs ou placas de desenvolvimento.
(CHAN et al., 2022)	Laboratório	Jetson Nano/TX2, Raspberry Pi 3/4	Baixa	Configuração de laboratório usando uma variedade de placas de desenvolvimento comuns.

Ref	Ambiente de Teste	Hardware Utilizado	Maturidade Tecnológica Avaliada	Justificativa
(KJORVEZIROSKI; FILIPOSKA, 2022)	Laboratório/pesquisa	Simulado	Baixa	Simulação em ambiente de laboratório.
(DENG et al., 2022)	Laboratório/simulação	Computador macOS	Baixa	Simulação executada em uma máquina de desenvolvedor padrão.
(PALLEWATTA; KOSTAKOS; BUYYA, 2024)	Laboratório (implícito)	Recursos virtualizados	Baixa	Uso de recursos virtualizados aponta para uma configuração simulada/emulada.
(IORIO et al., 2023)	Lab/simulado e produção real	Simulado	Alta	Uma mistura de simulação para benchmarks e um caso de uso em um ambiente industrial real.
(REHAN et al., 2023)	Laboratório controlado	Não Aplicável	Baixa	Ambiente de laboratório controlado; falta de hardware sugere foco no *framework* de software.
(GHASEMI; SCHRANZ, 2024)	Simulação	Simulado	Baixa	Puramente baseado em simulação.
(CENTOFANTI et al., 2024)	Laboratório	Simulado	Baixa	Simulação em ambiente de laboratório.
(LOBATO et al., 2023)	Plataforma de testes (testbed) real (infraestrutura física)	(Hardware real implícito)	Alta	Uso de uma plataforma de testes real com infraestrutura física de carregamento de VEs (Veículos Elétricos) em uma universidade.
(TUSA et al., 2024)	Laboratório/pesquisa	Simulado	Baixa	Simulação em ambiente de laboratório.
(SOFIA et al., 2024)	Plataformas de teste (testbeds) de pesquisa e alinhadas à indústria	LiDARs, AGVs, Câmeras térmicas	Alta	Testes em ambientes alinhados à indústria com hardware de nível industrial (AGVs, LiDARs).
(TRAN; KIM, 2024)	Plataforma de testes (testbed)/configuração experimental	Servidores Intel Xeon	Média	Uma plataforma de testes (testbed) estruturada usando hardware de servidor potente.
(BOTEZ et al., 2021)	Laboratório/pesquisa	Raspberry Pi, Nokia FZM eNBs	Média	Ambiente de laboratório que combina placas de desenvolvimento com hardware de telecomunicações comercial.

3.2 Avaliação Experimental

Esta seção detalha a metodologia empregada para atingir os objetivos propostos na fase de avaliação experimental, que se divide em duas partes principais: a Seleção das Plataformas de COPs para Ambientes de Borda e o Delineamento do Experimento. Anteriormente apresentamos como foi conduzida a busca da Revisão Sistemática e agora, o foco é em explicar como

identificamos e escolhemos as plataformas leves K3s e K0s para a avaliação prática.

Em seguida, é detalhado o planejamento do experimento, que, antes de prosseguir para a discussão dos resultados, foi organizado sistematicamente utilizando a estrutura GQM para definir claramente as questões de pesquisa, as métricas de desempenho e o hardware a ser utilizado. O objetivo central dessa metodologia é avaliar o desempenho dessas plataformas em dispositivos com recursos restritos, como o Raspberry Pi 3B+, e analisar a viabilidade de sua aplicação em cenários de *Edge Computing*.

Conduzimos uma busca estruturada na literatura com o objetivo de selecionar pelo menos duas COPs para avaliação comparativa. Esta busca é definida pelos três requisitos a seguir:

1. Abrangência de pelo menos uma tecnologia de orquestração: A busca deve incluir referências a pelo menos uma tecnologia de COP relevante.
2. Foco em borda e sistemas com recursos restritos: Os artigos devem visar especificamente sistemas de borda ou ambientes com restrições de recursos, como IoT, sistemas embarcados ou baseados em borda.
3. Comparação entre COPs: A literatura selecionada deve incluir análises comparativas ou avaliações entre COPs para facilitar o entendimento de suas vantagens e desvantagens.

A string de busca final foi formulada da seguinte maneira:

(kubernetes OR K3s OR nomad OR k8s OR microk8s OR "*container orchestration*")
AND
(iot OR *embedded* OR edge OR "*Resource Constrained*"OR "*Low Resource*")
AND
(*comparison* OR *comparative* OR *evaluation*)

A busca foi aplicada em dois repositórios, IEEE e Scopus, resultando em um total de 314 resultados (159 do Scopus e 155 do IEEE), publicados entre 2019 e 2024. Após a remoção de 73 duplicatas, conduzimos um processo de filtragem. Inicialmente, aplicamos o critério de inclusão:

- Critério de Inclusão: Estudos que comparam diversas plataformas de orquestração de contêineres. Esta etapa reduziu os resultados para 13 artigos. Em seguida, aplicamos o critério de exclusão.
- Critério de Exclusão: Estudos não focados em plataformas leves de orquestração de contêineres adequadas para ambientes com recursos restritos.

Após aplicar esses critérios, identificamos 9 artigos relevantes para nosso estudo, que são apresentados na seção seguinte.

3.2.1 Análises de Desempenho Anteriores

A literatura revisada abrange uma variedade de análises, focando em desempenho, segurança e capacidades de rede em plataformas de orquestração de contêineres. O Kubernetes é comumente identificado como a solução líder, embora COPs alternativas como Docker Swarm, Apache Mesos e HashiCorp Nomad também sejam mencionadas. No entanto, a maioria das opções não é otimizada para ambientes com recursos restritos ou de borda, e a direção predominante da pesquisa é em abordagens de Continuum Borda-Nuvem (Edge-to-Cloud Continuum).

- Ascensão et al. (2024) investiga distribuições do Kubernetes quanto à sua adequação em vários cenários, com foco em segurança e agilidade para lidar com ameaças emergentes, visando identificar as distribuições mais robustas para diferentes casos de uso.
- Jing, Qiao e Sinnott (2022) avalia tecnologias de contêineres em ambientes de IoT, considerando as limitações de memória, armazenamento e capacidade de processamento dos dispositivos de IoT. O artigo compara runtimes de contêiner, concluindo que o Containerd é ligeiramente mais eficiente que o Docker. O KubeEdge é destacado por suas modificações, que substituem alguns componentes do Kubernetes por alternativas autodesenvolvidas e eficientes em termos de recursos para um melhor controle em ambientes de borda.
- Valantasis, Makris e Korakis (2022) estuda a orquestração de redes na borda distante (far-edge), visando dispositivos com recursos restritos, como gateways de IoT portáteis. O trabalho enfatiza a importância da rápida instanciação, desativação (teardown), autoescalonamento e monitoramento de serviços nesses ambientes. Ele sugere o Nomad e o K3s como *frameworks* leves adequados para orquestrar microsserviços em dispositivos com recursos limitados.
- Koziolek e Eskandani (2023) contribui com um estudo empírico sobre as características e o desempenho de distribuições leves do Kubernetes, utilizando uma ferramenta de benchmarking para comparar sua eficiência.
- Böhm e Wirtz (2021) foca no consumo de recursos e tempo em distribuições leves (MicroK8s e K3s) em comparação com o Kubernetes nativo, avaliando eventos do ciclo de vida para fornecer insights detalhados sobre as demandas de recursos da plataforma.
- Fogli et al. (2021) foca em avaliar o desempenho de distribuições do Kubernetes em ambientes de nuvem federados e adaptativos dentro de redes táticas (TNs). As TNs frequentemente enfrentam condições de rede desfavoráveis, como largura de banda limitada e flutuante, latência variável e conectividade intermitente. Os resultados oferecem insights para otimizar o Kubernetes para implantação militar e fornecem orientação estratégica para parceiros de missão no desenvolvimento de estratégias de nuvem eficazes sob tais restrições.

- Bahy et al. (2023) compara KubeEdge, K3s e Nomad, com foco na utilização de recursos para identificar a plataforma mais eficiente para borda. O estudo examina o uso de memória e armazenamento, bem como o desempenho das plataformas em vários cenários, destacando a importância da comunicação segura entre nós via VPN. Os resultados oferecem orientação para organizações na seleção de uma plataforma de borda com base nas demandas de recursos e requisitos de segurança.

- Kjorveziroski e Filiposka (2022) explora o desempenho de três distribuições do Kubernetes e seus métodos de implantação, examinando especificamente como eles impactam a execução de funções sem servidor (serverless) em infraestruturas de borda com recursos restritos. O artigo fornece insights sobre a adequação de cada distribuição para cargas de trabalho sem servidor em ambientes de borda.

- Čilić et al. (2023) visa resumir e avaliar as principais ferramentas de orquestração de contêineres para borda, a fim de apoiar a seleção da ferramenta mais adequada para este domínio. O artigo fornece uma visão abrangente do estado da arte em orquestração de borda, categorizando as ferramentas com base em estratégias de agendamento e métodos de orquestração de contêineres. Esta pesquisa destaca tendências na orquestração de borda e serve como referência para avaliar ferramentas de orquestração de contêineres, com foco particular em conteinerização e estratégias de agendamento para serviços de borda.

3.2.2 Análise Comparativa

Kubernetes: demonstra o melhor desempenho geral e segurança entre as plataformas de orquestração de contêineres (ASCENSÃO et al., 2024). Ele opera de forma confiável mesmo sob restrições de recursos e se destaca em grandes cargas de trabalho (ČILIć et al., 2023).

K3s: uma distribuição leve do Kubernetes, apresenta um consumo de recursos inferior e se sobressai em desempenho de processador, memória e I/O de disco (**??**BAHY et al., 2023). No entanto, em cenários específicos onde os papéis de controlador e trabalhador (worker) são combinados, o desempenho do K3s pode ser prejudicado (KOZIOLEK; ESKANDANI, 2023).

MicroK8s: outra variante leve do Kubernetes, obtém resultados comparáveis aos do K3s, mas tende a consumir mais recursos (BÖHM; WIRTZ, 2021). Seu desempenho é ligeiramente superior ao do K3s no gerenciamento da variabilidade de cargas de trabalho (KJORVEZIROSKI; FILIPOSKA, 2022).

K0s: outra distribuição leve, mostra uma ligeira vantagem de desempenho sobre o K3s e possui maior segurança (ASCENSÃO et al., 2024). Embora seja uma distribuição promissora, sua adoção pela comunidade ainda é menor em comparação com outras alternativas.

Nomad: demonstra consistentemente alto desempenho em várias métricas, destacando-se na utilização de disco e no gerenciamento geral de recursos, o que o torna adequado para

aplicações de borda com recursos limitados (VALANTASIS; MAKRIS; KORAKIS, 2022; BAHY et al., 2023).

KubeEdge: projetado para ambientes de borda, tem um desempenho particularmente bom sob condições de rede degradadas, provando-se eficaz em cenários com conectividade limitada e intermitente, como redes táticas (FOGLI et al., 2021). No entanto, seu nó de controle consome mais recursos, pois utiliza o controlador original do Kubernetes.

As plataformas Docker Swarm e ioFog recebem menos ênfase nesta análise. O Docker Swarm é geralmente considerado menos ideal para ambientes com recursos restritos, enquanto o ioFog, embora frequentemente usado em configurações de IoT e borda, não apresenta vantagens claras listadas na literatura revisada.

3.2.3 Justificativa da Seleção

O presente estudo concentra-se na análise comparativa de distribuições baseadas em Kubernetes, visto que esta tecnologia se estabeleceu como o padrão de facto para orquestração de contêineres. O K3s foi selecionado como objeto primário de estudo devido à sua prevalência na literatura e ampla adoção pela comunidade (KUBERNETES. . . , 2022). Adicionalmente, o K0s foi incluído na análise para verificar a hipótese de superioridade de desempenho em relação ao K3s em cenários específicos.

Quanto aos critérios de exclusão, optou-se por não utilizar o MicroK8s nesta etapa; embora seja uma solução robusta no ecossistema Canonical, a literatura indica uma equivalência de desempenho com o K3s, o que reduziria a relevância comparativa para os fins desta pesquisa. O KubeEdge foi desconsiderado devido ao overhead de seu *control plane*, que, apesar da resiliência de rede, poderia enviesar os experimentos de consumo de recursos. Por fim, o Nomad não foi incluído por divergir do escopo de distribuições Kubernetes.

3.2.4 Delineamento do Experimento

Este estudo visa avaliar o desempenho de plataformas leves de orquestração de contêineres em dispositivos de poucos recursos, com foco nas operações do *control plane* (*control plane*). O objetivo é analisar a viabilidade de executar clusters de nó único ou nós mestres em hardware com recursos restritos e identificar os fatores-chave que impactam o desempenho das COPs selecionadas. Para atingir esse objetivo, adotamos a estrutura QGM para orientar sistematicamente nossos experimentos.

3.2.4.1 Questões e Métricas

Para nortear a coleta de dados e garantir que os resultados experimentais reflitam os objetivos técnicos delineados para esta pesquisa, adotou-se a metodologia GQM. Esta estrutura

permite decompor o objetivo principal em questões específicas que, por sua vez, são respondidas através de métricas quantificáveis coletadas durante a execução dos testes.

- Q1: Qual é a utilização de recursos do dispositivo quando gerenciado pela plataforma de orquestração?
  - M1: Utilização da CPU do dispositivo embarcado durante a operação.
  - M2: Porcentagem de uso de memória sob condições de carga padrão.
  - M3: Uso de I/O de disco.
- Q2: Qual é o desempenho da COP ao executar o *control plane* no dispositivo com recursos restritos?
  - M1: Latência de inicialização do pod (tempo para iniciar um pod).
  - M2: Latência de desativação do pod (tempo para encerrar um pod).
- Q3: Como diferentes configurações de hardware (ex: tipo de armazenamento) impactam o desempenho das COPs?
  - M1: Operações de IOPS sob cargas de trabalho variadas (4 e 8 *pods*).
  - M2: Uso de CPU e memória ao utilizar diferentes tipos de armazenamento.
- Q4: Como o número de *pods* afeta o desempenho das COPs?
  - M1: Utilização de recursos (CPU, memória, I/O de disco) com o aumento do número de *pods*.
  - M2: Latência na criação e desativação de *pods* com contagens variadas de *pods*.

3.2.4.2 Hardware

A escolha do hardware para a avaliação experimental foi guiada pela necessidade de representar um ambiente de borda com recursos severamente restritos. A seleção do Raspberry Pi 3B+, equipado com apenas 1 GB de memória RAM, foi uma decisão metodológica deliberada para estabelecer um cenário de estresse rigoroso, situando-se abaixo dos requisitos recomendados para o controlador do K3s (2 GB), conforme apresentado na Tabela 6. Esta restrição de hardware visa simular condições extremas de *Edge Computing*, onde a escassez de recursos é a norma e a atualização de hardware nem sempre é viável. Ao operar no limiar da capacidade física, o experimento amplifica as diferenças de *overhead* entre as plataformas, permitindo avaliar não apenas a conformidade teórica, mas a resiliência prática e a capacidade de sobrevivência do *control plane* quando submetido a uma competição severa por memória e processamento.

Tabela 6 – Requisitos Mínimos Oficiais: K3s vs. K0s

Recurso	**K3s**	**K0s**
CPU	2 Cores	1 vCPU
Memória RAM (Controller)	2 GB	1 GB
Memória RAM (Worker)	512 MB	512 MB
Arquitetura	x86_64, ARMv7, ARM64	x86_64, ARMv7, ARM64
Sistema Operacional	Linux Kernel 3.10+	Linux Kernel 3.10+

Fonte: Elaborado pelo autor com base na documentação oficial. (K3S. . . , 2025; K0S. . . , 2025a)

Além das restrições de recursos físicos, a compatibilidade plena com o sistema operacional Linux constituiu um critério eliminatório para a seleção do hardware. As plataformas de orquestração de contêineres analisadas (K3s e k0s) são intrinsecamente dependentes de primitivas exclusivas do kernel Linux — especificamente control groups (cgroups) para o gerenciamento de recursos e namespaces para o isolamento de processos. Portanto, o dispositivo de borda selecionado deveria ser capaz de executar distribuições Linux, diferenciando-se de microcontroladores que operam exclusivamente com sistemas de tempo real (RTOS) ou firmwares bare-metal, onde a abstração necessária para a orquestração de contêineres nativa não está disponível.

As especificações técnicas do dispositivo adotado são fundamentais para entender os limites de desempenho observados:

- **SoC:** Broadcom BCM2837B0, baseado na arquitetura Cortex-A53 (ARMv8) de 64 bits.
- **Frequência de Operação:** 1,4 GHz.
- **Memória RAM:** 1 GB LPDDR2.
- **Armazenamento:** Interface para cartão microSD.

3.2.4.3 Considerações Técnicas

A implementação de orquestradores em dispositivos de borda exige ajustes específicos no kernel e no sistema operacional para garantir a estabilidade do *control plane*:

- **Gestão de Memória Virtual:** Foi necessário a desativação do uso de *Swap* pelos cgroups para garantir o determinismo do Orquestrador. Em dispositivos com baixa memória RAM, isso torna a densidade de *pods* crítica.
- **Cgroups:** É mandatória a ativação dos grupos de controle (*cgroups*) no firmware do dispositivo (ex: `cmdline.txt` no Raspberry Pi) para permitir o isolamento de recursos de memória e CPU.

3.2.4.4 Coleta de Métricas

Para a coleta de métricas do experimento, foram utilizadas as ferramentas Prometheus e Grafana. O Prometheus (K0S. . . , 2025b) é um sistema de monitoramento que coleta dados de alvos configurados em intervalos definidos. O Grafana (K0S. . . , 2025c), por sua vez, é usado para consultar, visualizar e criar painéis a partir dos dados coletados.

Conforme ilustrado no diagrama da arquitetura de testes, o Prometheus e o Grafana foram hospedados em um computador desktop na mesma rede do Raspberry Pi, permitindo a coleta de dados por meio de requisições HTTP. Durante a execução dos testes, foram observadas falhas intermitentes, *timeouts* devido espera por disco, com a plataforma K3s. A execução dos experimentos foi conduzida iterativamente até a obtenção de 10 amostras válidas.

3.2.4.5 Design Experimental Fatorial

O estudo emprega um design fatorial completo de 2 níveis (2^k) (JAIN, 1990) para avaliar três fatores, cada um com dois níveis para avaliar sistematicamente o impacto de três fatores distintos, cada um com dois níveis (baixo e alto). Os fatores investigados foram (A) o tipo de meio de armazenamento, (B) a plataforma de orquestração (COP), e (C) a quantidade de *pods* implantados, conforme detalhado na Tabela 7.

Tabela 7 – Fatores e Níveis no Design Fatorial 2^k

Fator	**Nível 1 (-)**	**Nível 2 (+)**
Tipo de Armazenamento (A)	SanDisk Ultra	SanDisk Extreme
Distribuição COP (B)	K3s	K0s
Quantidade de *pods* (C)	4 *pods*	8 *pods*

Para o Fator A (Tipo de Armazenamento), foram avaliados dois modelos de cartão microSD SanDisk de 64GB, cujas especificações detalhadas são:

- **Nível 1 (-):** Cartão SanDisk Ultra 64GB, Classe 10, XC-I, UHS-I.
- **Nível 2 (+):** CartãoSanDisk Extreme 64GB, Classe 10, XC-I, UHS-I, V30, U3 e A2.

3.2.4.6 Script de Benchmark

Conforme estabelecido pelo GQM, foi implementado um script de benchmark A. Este script executa um processo iterativo para quantificar duas métricas fundamentais: a latência de criação, medida desde o comando de aplicação (kubectl apply) até que todos os *pods* atinjam o estado de prontidão (*Ready*), e a latência de exclusão ou destruição dos *pods*, que corresponde ao tempo necessário para a remoção completa do namespace e de todos os seus recursos. Durante toda a execução do Script, as métricas de interesses restantes são coletadas via Prometheus.

O benchmark é conduzido para cenários de cargas distintas, especificamente implantações de 4 e 8 *pods*, a fim de analisar o impacto da escalabilidade na agilidade do cluster. Para garantir a fidedignidade dos resultados, cada iteração é executada em um namespace isolado e seguida por um período de pausa, permitindo a estabilização do ambiente e evitando a interferência entre os testes consecutivos.

Para garantir o isolamento das métricas do control plane, os *Pods* implantados utilizam uma imagem Nginx padrão em estado de espera (*idle*), sem serem submetidos a requisições externas ou processamento de dados. Desta forma, assegura-se que o consumo de CPU e Memória registrado reflete exclusivamente o overhead operacional da plataforma de orquestração (agendamento, manutenção de estado e verificações de saúde), eliminando o ruído que seria introduzido por uma carga de trabalho variável na camada de aplicação.

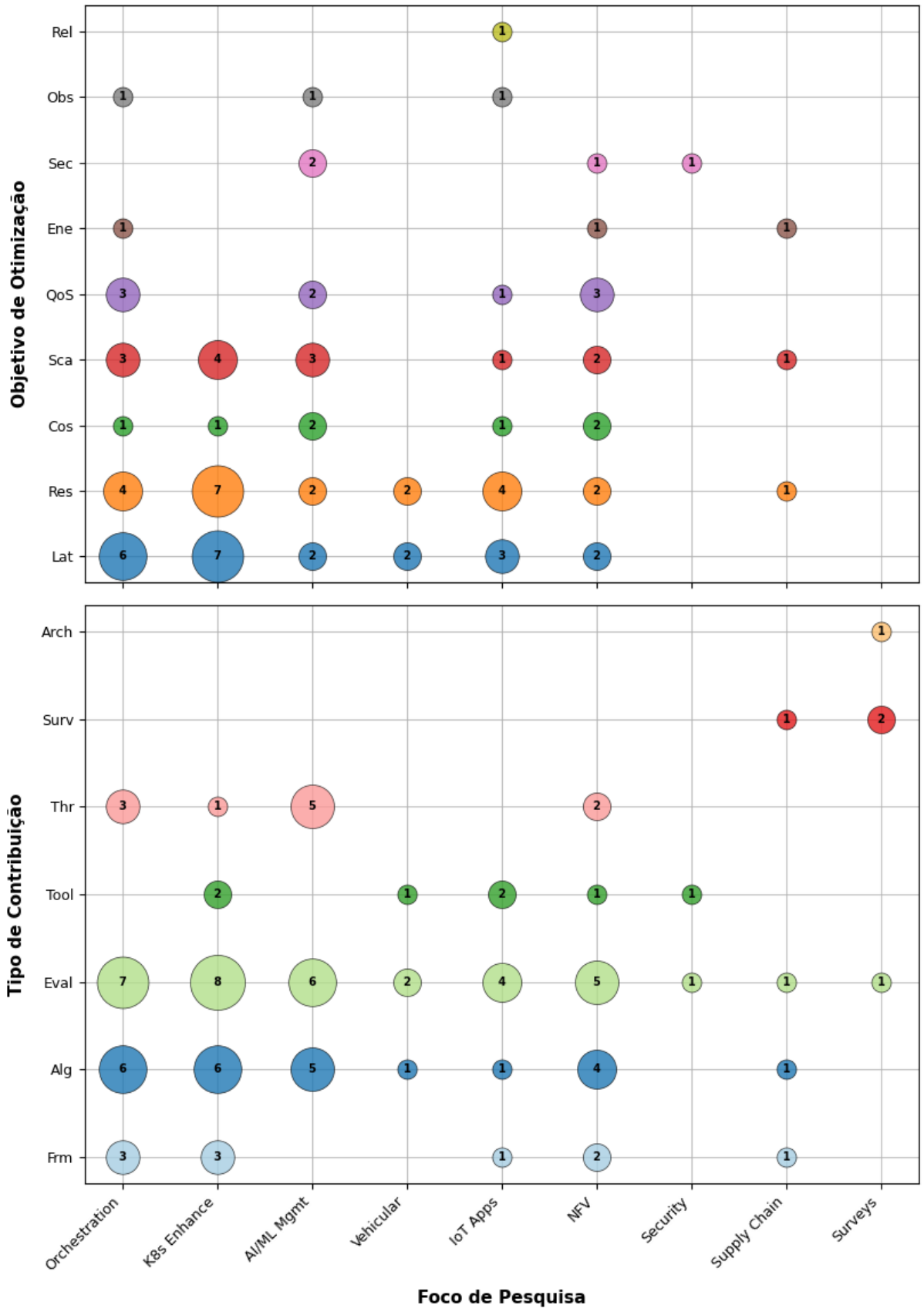


Figura 6 – Visualização de um Mapa Sistemático na forma de um gráfico de bolhas (Fonte: Próprio Autor)

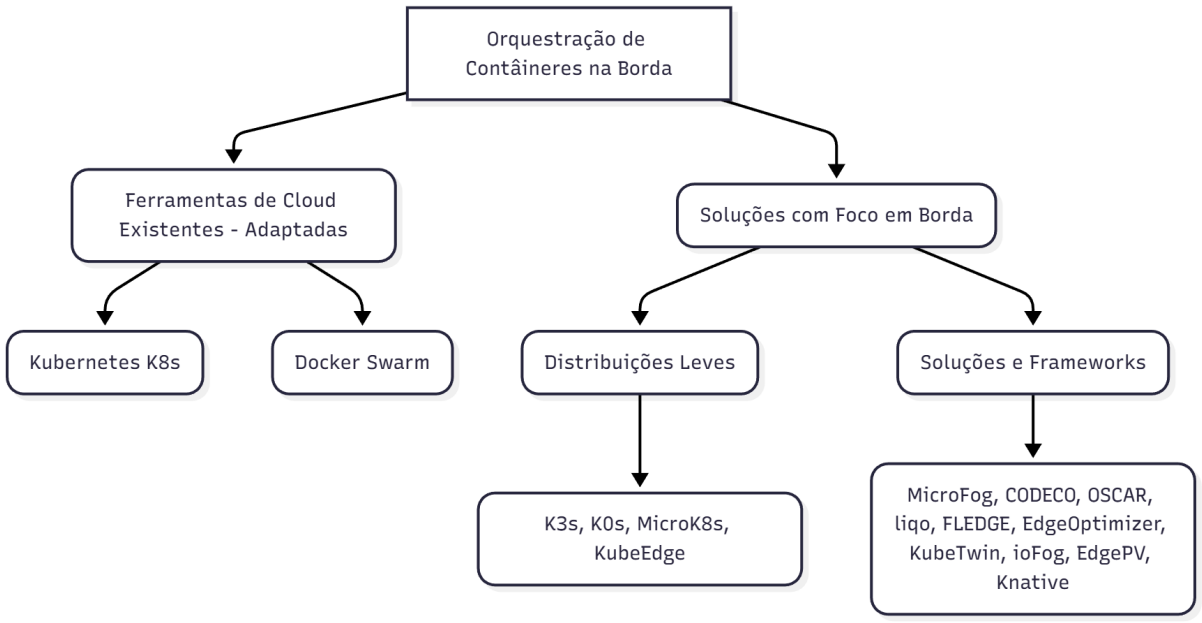


Figura 7 – Abordagens e ferramentas de orquestração de contêineres usadas ou investigadas na pesquisa sobre borda (Fonte: Próprio Autor)

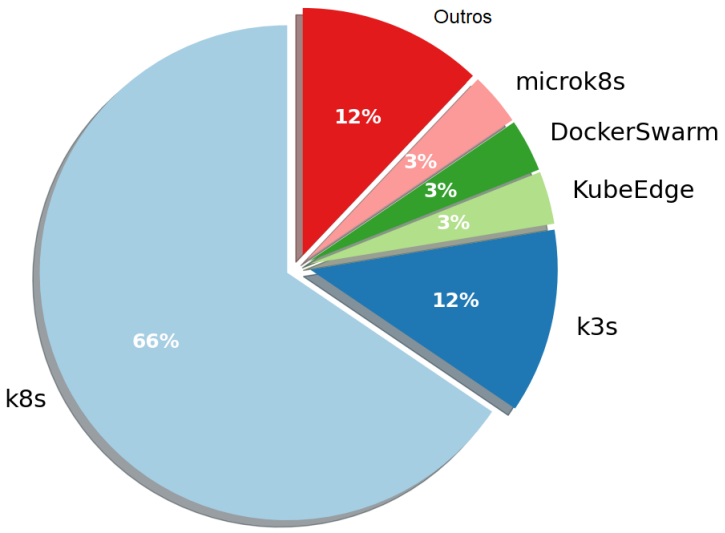


Figura 8 – Distribuição das plataformas de orquestração de contêineres utilizadas (Fonte: Próprio Autor)

Figura 9 – Domínios de Aplicação (Fonte: Próprio Autor)

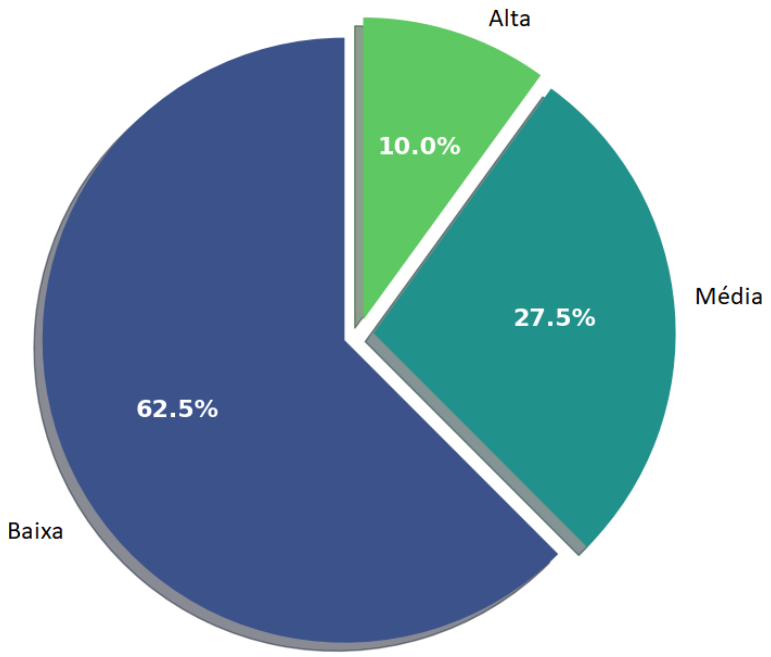


Figura 10 – Distribuição dos Estudos por Maturidade Tecnológica Avaliada (Fonte: Próprio Autor)

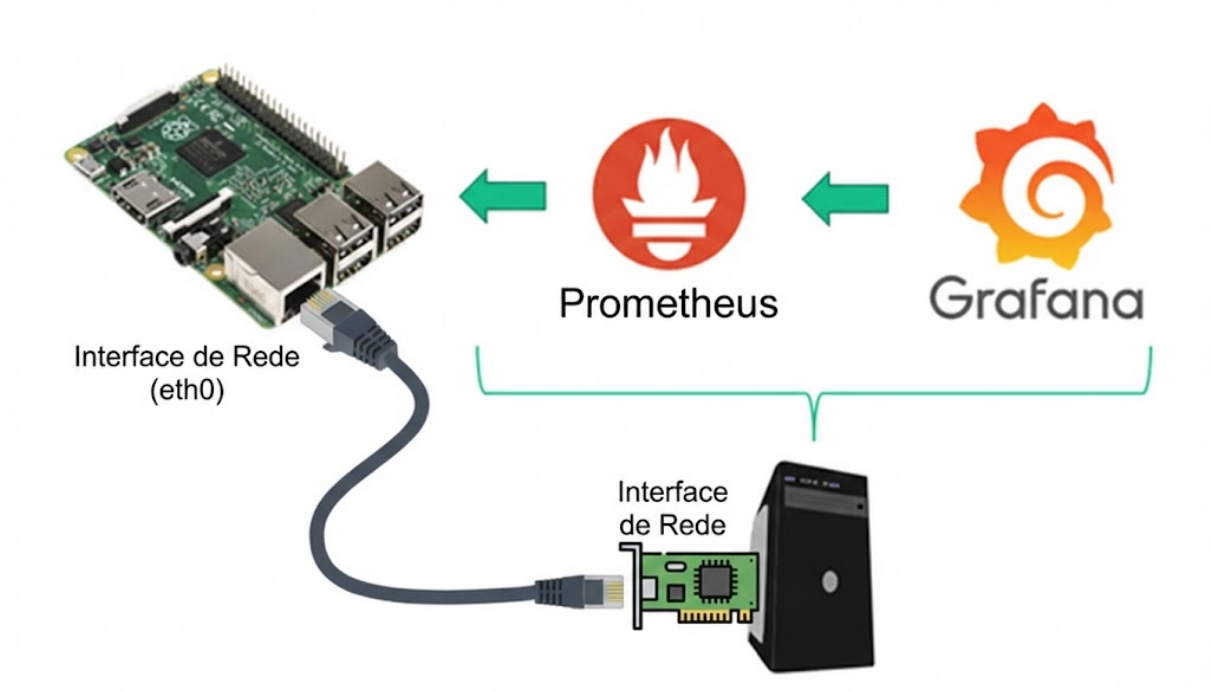
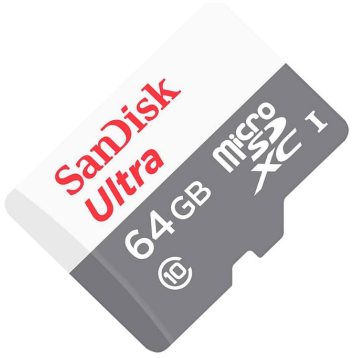


Figura 11 – Arquitetura de Teste. (Fonte: Próprio Autor)



(a) SanDisk Ultra 64GB (Nível 1).

(b) SanDisk Extreme 64GB (Nível 2).

Figura 12 – Cartões microSD utilizados para o Fator B (Tipo de Armazenamento).

4

Resultados e Discussão

Este capítulo apresenta a análise detalhada e a discussão dos dados coletados durante a avaliação experimental, conduzida com o objetivo de investigar a viabilidade e o desempenho de plataformas leves de orquestração de contêineres em dispositivos com recursos restritos. Seguindo a abordagem GQM e o desenho experimental fatorial definidos na metodologia.

Os resultados coletados não são analisados apenas com base em médias aritméticas simples, mas são submetidos a uma avaliação de rigor estatístico para validar a significância dos dados. A análise parte implicitamente da hipótese nula que pressupõe que a alteração de um fator experimental — como a troca da plataforma de orquestração (COP) de K3s para K0s ou a mudança na velocidade do cartão SD — não exerce efeito significativo sobre as métricas de desempenho do sistema (como tempo de CPU ou latência). Para determinar se a hipótese nula deve ser aceita ou rejeitada, o estudo utiliza o p-valor como métrica de decisão.

A Figura 13 nos mostra que há uma diferença considerável entre os COPs para operação de criação de *pods*. Quando houve a mudança de cartão, o COP K3s não sofreu alteração estatisticamente significativa em sua performance, já o K0s apresentou uma diminuição significativa (p valor < 0,01) no tempo de criação, para 8 *pods*, ao trocar do cartão Ultra para o Extreme. Em todos os outros casos as mudanças não são significativas. O incremento observado na média, principalmente no K3s, pode-se dar pela maior exigência de trabalho dos serviços extras quando há o aumento de carga. O alto consumo de recursos no K3s pode não ter permitido que o sistema se aproveitasse da melhoria de velocidade dos cartões.

Para o tempo de destruição dos *pods* (Figura 14) continuamos a observar que o maior impacto se dá entre a mudança do COP, enquanto o cartão SD praticamente não faz diferença. A única exceção é do K0s com 8 *pods*, mesma situação da criação de *pods*. Por não ter nenhum serviço extra, o K0s consegue otimizar os acessos ao cartão e se beneficia significativamente (p valor = 0,048) da mudança de velocidade. O mesmo não acontece em 4 *pods* possivelmente por que o efeito não é grande o suficiente para ser medido. O tempo consumido de CPU (Figura 15) não se altera com a mudança do cartão para o COP K3s. Para o COP K0s, a redução no consumo de CPU (4 *pods* = 51%; 8 *pods* = 37,2%) é significativa (p valor 4 *pods* < 0,01; p valor 8 *pods* = 0,017) com o uso de um cartão mais rápido.

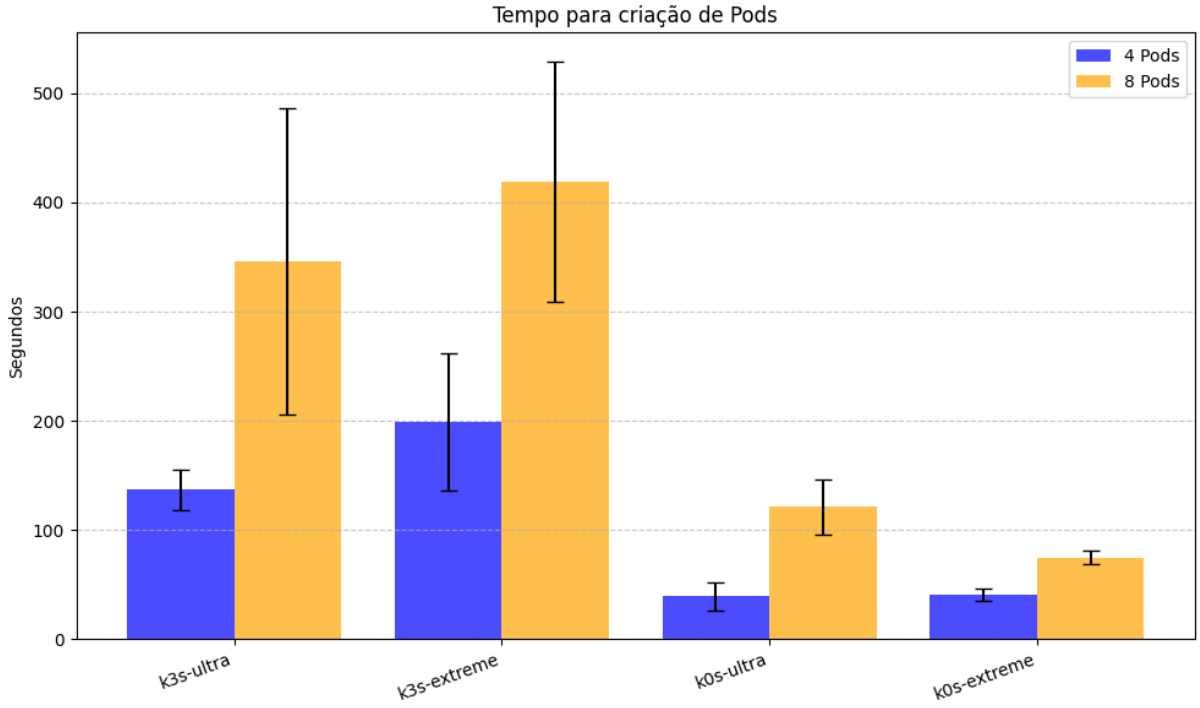


Figura 13 – Latência média para criação de *Pods* comparando K3s e K0s sob variação de carga e tipo de armazenamento. (Fonte: Próprio Autor)

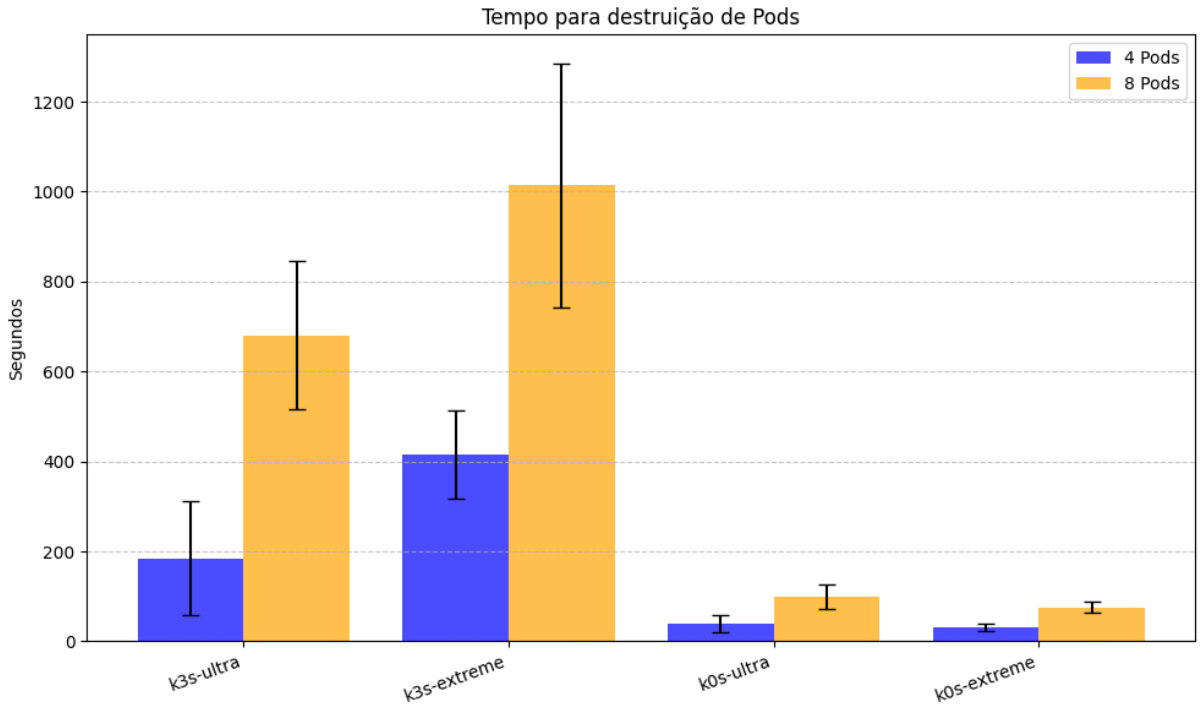


Figura 14 – Latência para destruição de Pods evidenciando o impacto da escolha do orquestrador. (Fonte: Próprio Autor)

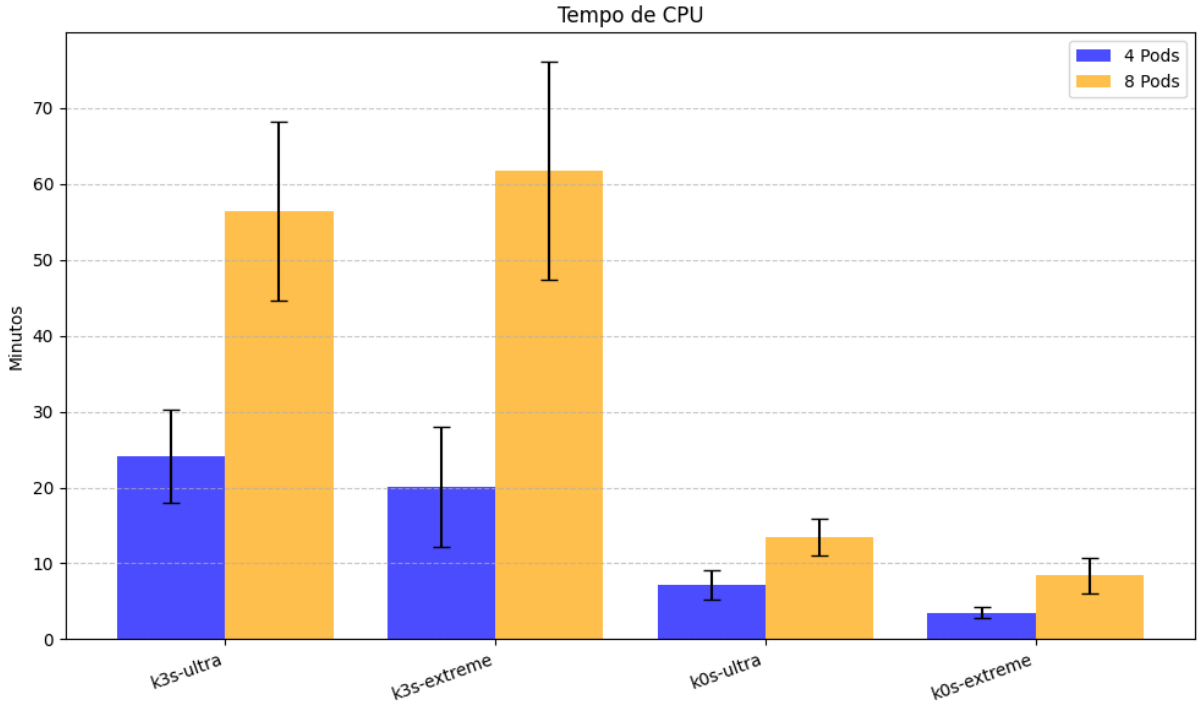


Figura 15 – Tempo total de CPU (em minutos) consumido pelos processos de controle durante a execução do *benchmark*. (Fonte: Próprio Autor)

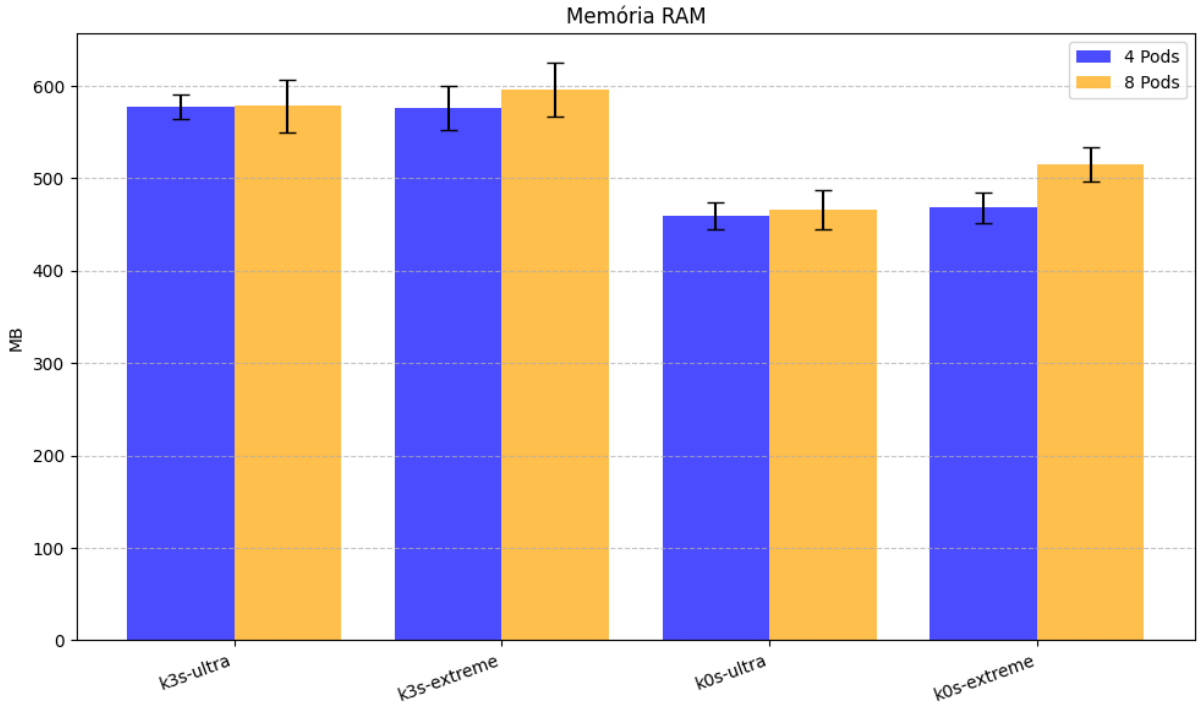


Figura 16 – Alocação média de memória RAM residente: comparativo entre o overhead do K3s e o minimalismo do K0s. (Fonte: Próprio Autor)

O consumo médio de Memória RAM (Figura 16) não se alterou com o aumento de número de *pods*, pois a aplicação utilizada tem baixo consumo de memória RAM. Com a mudança

do COP, houve alteração no consumo médio, beneficiando o K0s. Isso pode ser justificado pelos serviços extras que o K3s possui. A troca do cartão não gerou impacto na RAM, como era esperado.

O uso de *Swap* se alterou principalmente com a mudança do COP, conforme Figura 17, enquanto a mudança do cartão trouxe pequenas ou nenhuma alteração para o COP K3s, sendo significativa (p < 0,01) apenas para 8 *pods* com um baixo impacto (7,4%). O acréscimo no uso de *Swap* pode ser justificado pelos serviços extras presentes no K3s. Quando analisamos o K0s, o cartão SanDisk Extreme trouxe redução significativa (p valor 4*pods* < 0,01; p valor 8 *pods* < 0,01) no uso de *Swap*, com um impacto considerável (4 *pods* = 15,9%; 8 *pods* = 15%).

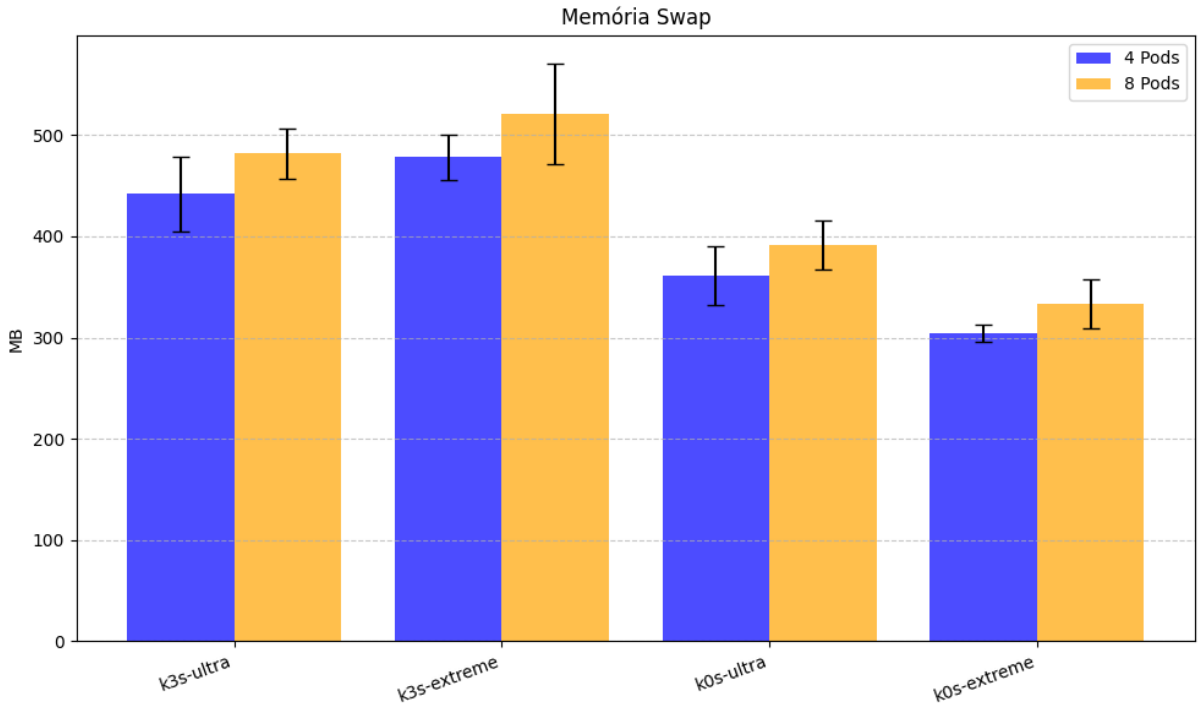


Figura 17 – Utilização de memória *Swap*, destacando a pressão exercida pelos serviços auxiliares do K3s sobre a memória virtual. (Fonte: Próprio Autor)

De acordo com a Figura 18 podemos ver que com o aumento de *pods*, é exigido mais uso do Disco. O COP K3s intensifica ainda mais o uso de disco, pela presença de um conjunto maior de ferramentas disponíveis. Para ambos os COP, a expectativa era que com o aumento da quantidade de *pods*, de 4 para 8, o uso do disco crescesse de forma igual, 100%, porém a realidade não foi essa, o K0s teve um aumento médio de 103.5% se aproximando da expectativa, enquanto o K3s 171%, uma justificativa sobre esse aumento considerável no K3s pode ser, novamente, os serviços extras que ele possui, pois alguns desses serviços armazenam parte das informações de gestão no disco. Entre cartões não era esperado que houvesse mudança de quantidade IO lido. No entanto, para o K0s, foi observada uma diminuição (4 *pods* = 51%; 8 *pods* = 34%) significativa (p valor 4 *pods* < 0,01; p valor 8 *pods* = 0,017) em ambos os casos. É possível que isto se dê porque a rapidez pode ajudar a manter os acertos de cache, sendo mais rápido há maior chance que a informação ainda esteja presente.

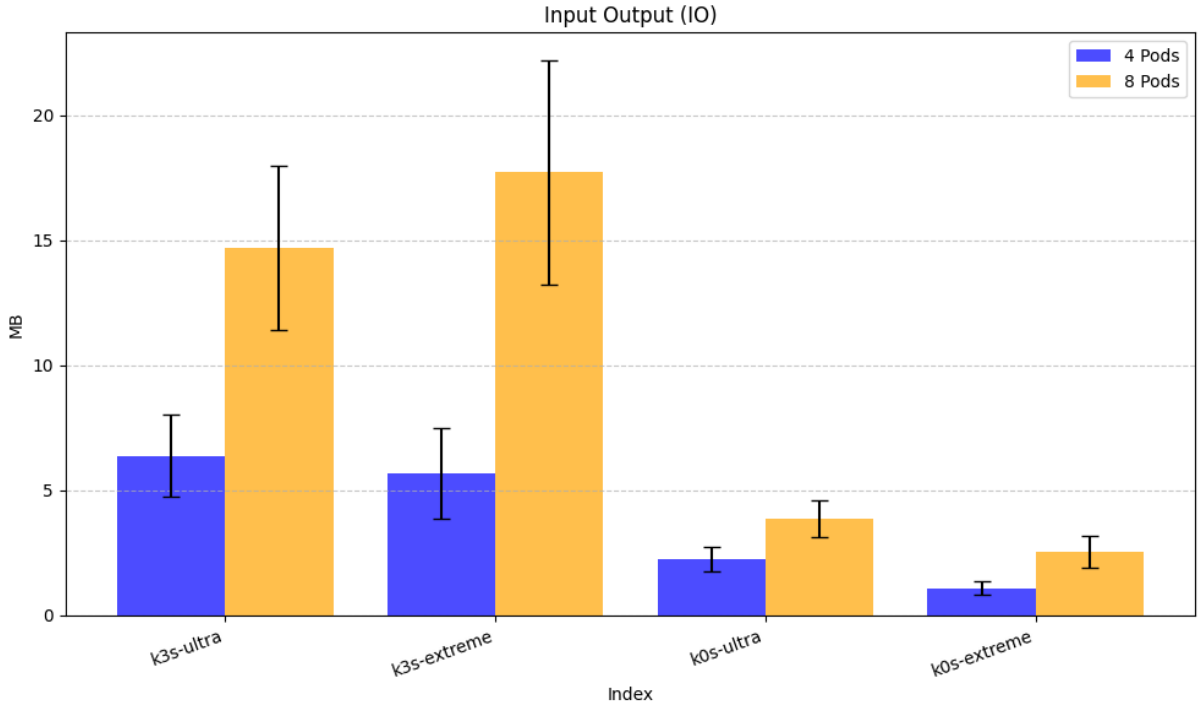


Figura 18 – Volume total de dados trafegados (*Throughput* de Disco em MB) durante o ciclo de vida dos *Pods*. (Fonte: Próprio Autor)

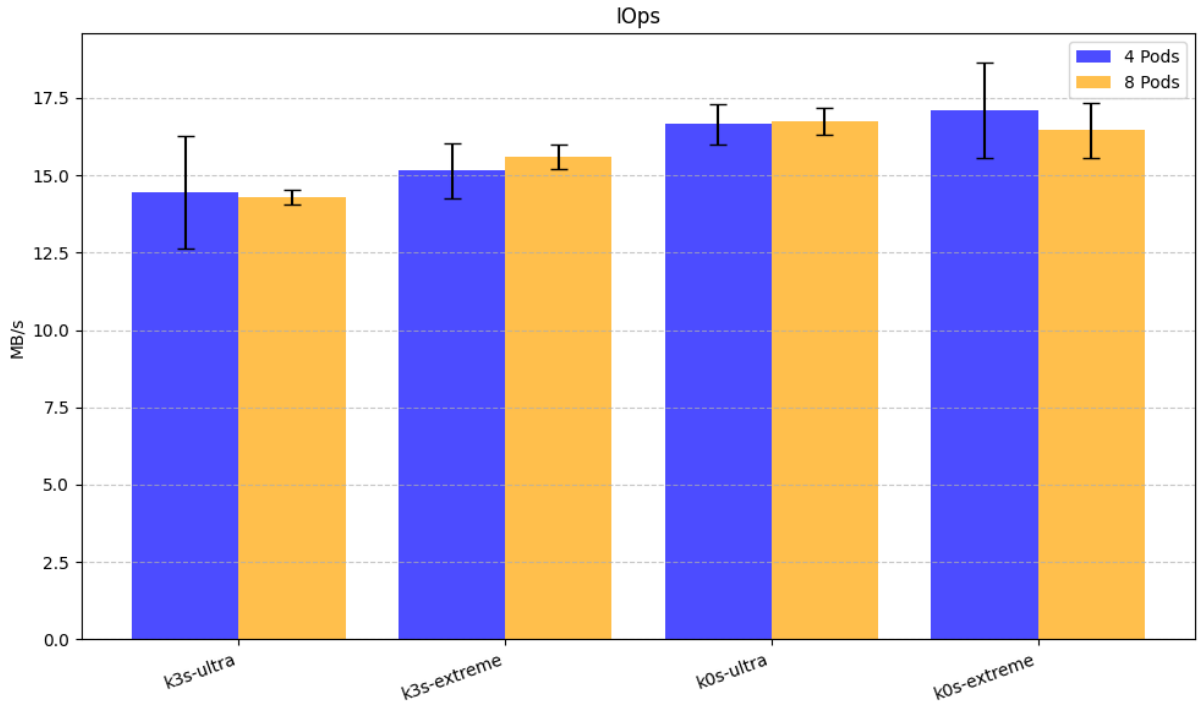


Figura 19 – Entrada/Saída por Segundo (IOPS), indicando a intensidade de acesso ao disco de cada orquestrador. (Fonte: Próprio Autor)

O uso médio de Disco pouco se alterou ou não se alterou com a mudança de cartões, nem com o aumento de número de *pods*, Figura 19. O único valor que sofreu alteração relevante e significativa (p valor < 0,01) é o aumento de *Throughput* quando mudou do cartão Ultra ao

Extreme, para 8 *pods* e K3s, aumento de 9%. A alteração do COP resulta em valores maiores de IOps para o K0s, em média 12.5% maiores. Esse aumento tem significância quando comparamos as barras de 8 *pods* combinados com o cartão Sandisk Ultra (p valor < 0,01) e 4 *pods* com o cartão Sandisk Extreme (p valor = 0,01). A seguir apresentamos os gráficos do consumo de CPU, classificado nos diferentes modos de operação, dentro de uma faixa de 30 minutos para os COPs K0s e K3s.

Como podemos ver na Figura 20, em Idle o K0s consome 8,1%, enquanto no K3s o consumo chega a 17,5% de CPU. Praticamente toda essa diferença se deve aos recursos extras do K3s que precisam de fazer acesso ao disco. Como podemos ver nos outros cenários (4 e 8 *pods*) isso atrapalha muito a aplicação se ela também precisa fazer acesso ao disco, possivelmente pelo aumento de *cache-miss*. Podemos verificar essa questão na diferença dos gráficos de 4 para 8 *pods* no K0s. Era esperado que, com o dobro de carga no sistema, o consumo de disco (CPU IOwait) duplicasse. No entanto ele só aumentou 42,9%. Essa otimização se deve possivelmente a reutilização das imagens pela cache.

Figura 20 – Perfil de utilização de CPU: Impacto do tempo de espera por disco (I/O *Wait*) versus tempo ocioso (*Idle*) nos cenários de 4 e 8 *Pods*. (Fonte: Próprio Autor)

Para sintetizar os resultados da avaliação experimental, quantificamos o impacto percentual que cada fator experimental teve sobre o desempenho do sistema. A Análise de Impacto de Fatores é um método estatístico utilizado para quantificar o quanto cada variável (fator) de um experimento contribui para as alterações observadas nas métricas de desempenho do sistema.

No contexto do trabalho, baseada no Design Experimental Fatorial, essa análise serve para "decompor"os resultados e entender quem é o verdadeiro responsável pelas mudanças de performance: se é o software (orquestrador), o hardware (cartão SD) ou a carga de trabalho (quantidade de *pods*). Além disso podemos visualizar o impacto que ocorre apenas quando dois ou mais fatores mudam juntos (ex: BC). Isso revela se, por exemplo, o K3s (Fator B) se comporta de maneira desproporcionalmente pior quando a carga aumenta (Fator C), criando um gargalo que não existiria no K0s.

A análise é visualizada através de um gráfico de radar (Figura 21), onde cada eixo representa uma métrica (como CPU, RAM, IOps). Quanto mais a linha de um fator se estica em direção à borda do círculo, maior é a influência daquele fator sobre aquela métrica específica.

A mudança do cartão (A) tem um impacto mínimo, só é relevante quando o sistema faz muito uso do cartão (AB) e possivelmente nessa situação começa a chegar no limite físico de transferência do cartão, conforme Figura 20. Podemos observar que o principal impacto é o COP (B). Parte disso pode ser atribuído à diferença dos modos de instalação entre os Orquestradores, o K3s traz uma grande quantidade de serviços em sua instalação padrão.

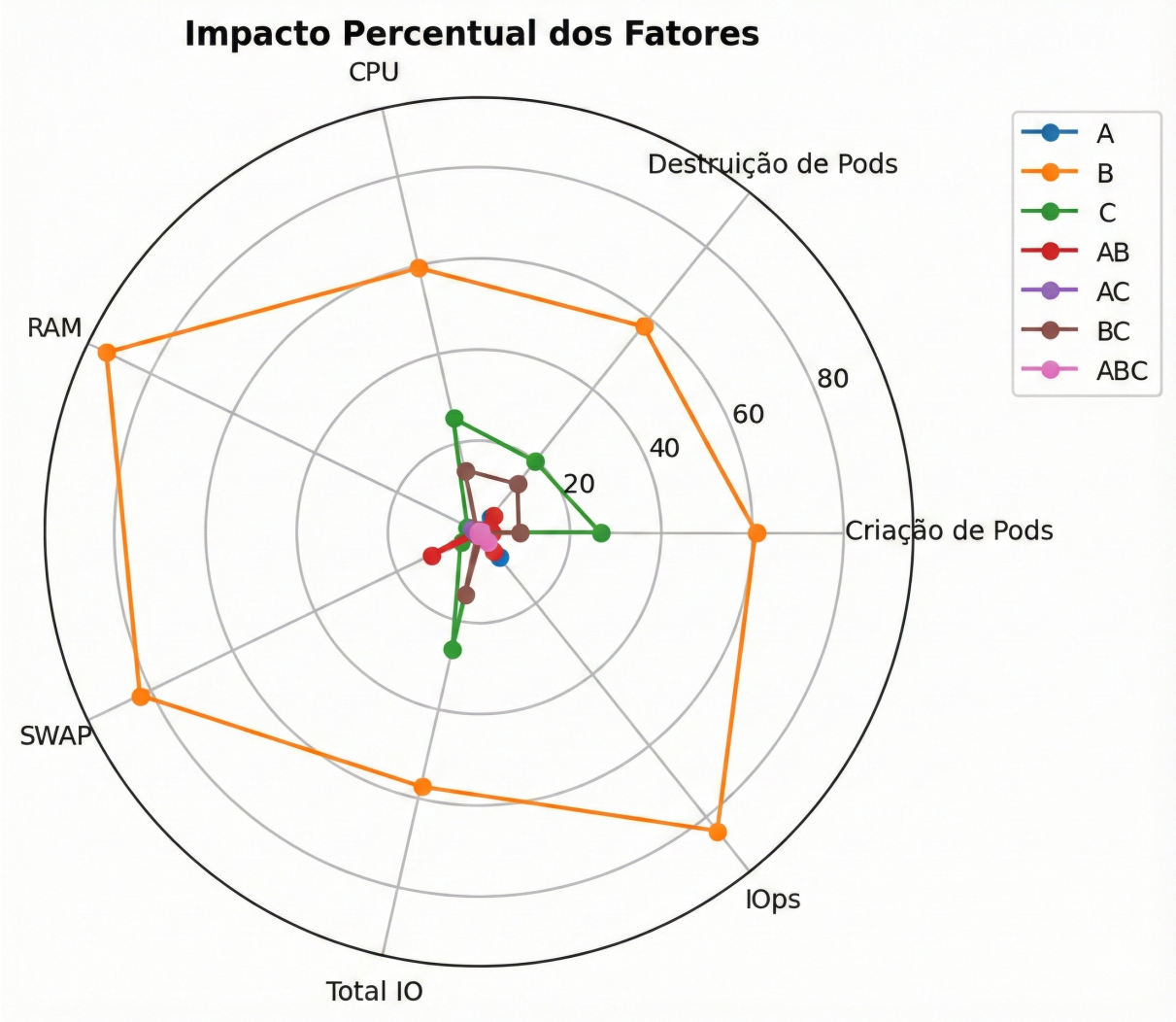


Figura 21 – Gráfico de Radar da Análise Fatorial 2^k: Impacto percentual dos fatores: Armazenamento (A), Orquestrador (B) e Carga (C) sobre as métricas do sistema. (Fonte: Próprio Autor)

Também notamos que os maiores impactos do COP, são em RAM, *Swap* e IOps. Acreditamos que isso se deve ao fato do nosso teste não submeter carga significativa de trabalho a aplicação em termos de memória e disco. Além disso, como quase não percebemos impacto do modelo do cartão, estes parâmetros são dominados pelas diferenças do COP.

Em seguida podemos observar a Quantidade de *pods* (C) como o segundo fator mais impactante. O seu impacto isolado só é percebido em CPU minutes, Criação de *Pods*, Destruição de *Pods* e Total IO. Como as aplicações executadas não fazem uso significativo nem de memória nem de disco, o aumento da quantidade de *pods* não impacta em RAM, *Swap* e IOps. Além dos impactos isolados do COP e Quantidade de *pods*, percebemos que a combinação desses fatores (BC) também é significativa. Isso se deve dar pelo fato de que no K3s além do aumento de demanda esperado pelo aumento de *pods*, os serviços extras também ficam mais sobrecarregados, o que não acontece no K0s.

A Análise Fatorial 2^k indicou que o Fator B (Plataforma de Orquestração) exerceu a maior

influência sobre as métricas de Memória RAM, uso de Swap e IOps, sobrepondo-se ao impacto gerado pela troca do Fator A (Tipo de Armazenamento). Isso demonstra que as diferenças na arquitetura do software e nos serviços instalados por padrão são determinantes para o consumo de recursos do sistema.

O Fator C (Quantidade de Pods) apresentou efeito direto sobre o tempo de CPU e o volume total de I/O. Além disso, a interação entre o orquestrador e a carga de trabalho (Interação BC) mostrou-se relevante, indicando que o aumento na quantidade de pods resulta em uma demanda de recursos distinta para cada plataforma, com o K3s apresentando maior consumo nessas condições.

5

Conclusão

Com base na análise de 41 artigos selecionados utilizando a estrutura PICOC, o estudo destacou o Kubernetes como a principal ferramenta de orquestração, embora tenha observado que as ferramentas existentes, projetadas para ambientes de nuvem homogêneos, enfrentam desafios significativos quando aplicadas à natureza heterogênea, com recursos restritos e distribuída dos ambientes de borda.

Os principais desafios identificados incluem limitações de recursos, aumento da complexidade de agendamento considerando latência e mobilidade, e a adaptação da escalabilidade e confiabilidade para cenários de borda dinâmicos. As abordagens de pesquisa propostas focam na adaptação de plataformas existentes usando distribuições leves e extensões específicas para a borda, como o KubeEdge, bem como no desenvolvimento de soluções personalizadas centradas em agendamento ciente de atraso, posicionamento de serviço sofisticado e técnicas de gerenciamento dinâmico de recursos.

Essas estratégias de orquestração estão sendo aplicadas em diversos domínios de aplicação, como Cidades Inteligentes, Saúde e IoT Industrial. As avaliações comumente se baseiam em bancadas de teste (*testbeds*) reais ou emuladas, medindo o sucesso com base em métricas de Desempenho críticas como Latência, Vazão (*Throughput*) e Utilização de Recursos.

Apesar desses avanços, a adequação dos *frameworks* existentes para dispositivos de baixos recursos e a insuficiência do agendamento padrão para minimizar a latência na borda permanecem como limitações. Consequentemente, pesquisas futuras devem priorizar o desenvolvimento de novos *frameworks* nativos da borda, algoritmos de agendamento complexos adaptados para a heterogeneidade, a otimização do auto-scaling e a condução de experimentos rigorosos no mundo real para aproveitar efetivamente todo o potencial do contínuo borda-nuvem.

5.1 Análise de Requisitos do Nó de Controle em Ambientes de Borda

A determinação dos requisitos mínimos para o *control plane* (*Control Plane*) em cenários de *Edge Computing* exige uma análise multidimensional, visto que o nó de controle acumula as

funções de gerenciamento da API, agendamento de tarefas e manutenção do estado do cluster. Com base nos experimentos realizados, a Tabela 8 sintetiza as necessidades de hardware para o funcionamento estável do nó de controle em placas SBC.

Tabela 8 – Requisitos Mínimos e Recomendados.

Recurso	**Mínimo**	**Recomendado**
Memória RAM	1 GB	2 GB a 4 GB
Armazenamento	MicroSD Classe 10	SSD USB 3.0 ou NVMe
Rede	Fast Ethernet (100 Mbps)	Gigabit Ethernet (1000 Mbps)

Para distribuições leves, o consumo base do *control plane* gira em torno de 512 MB a 800 MB. Contudo, a ausência de *Swap* impõe que o nó possua ao menos 2 GB para evitar o acionamento do *OOM Killer (Out of Memory Killer)*, um mecanismo de segurança do kernel do Linux projetado para salvar o sistema de um colapso total quando a memória RAM física se esgota completamente, em cenários de re-agendamento de carga.

O maior limitador de performance em nós de controle na borda não é a CPU, mas o I/O. O banco de dados de estado realiza escritas constantes. Em cartões MicroSD, a alta latência de escrita pode levar ao *timeout* do *control plane*, causando instabilidade no cluster. Além disso, a alta frequência de I/O do *control plane* exige mecanismos eficientes de Wear Leveling. Para implantações que utilizem COPs em SBCs, recomenda-se fortemente a substituição de cartões SD padrão por tecnologias mais resilientes como SSDs via USB/NVMe, ou ainda a configuração do orquestrador para manter logs e métricas efêmeras em memória RAM (tmpfs), visando reduzir estender a vida útil do dispositivo.

A comunicação entre o nó de controle e os nós trabalhadores (*Workers*) deve ser resiliente. Em ambientes de borda, a flutuação de latência na rede pode ser interpretada pelo *control plane* como falha do nó, disparando processos de recuperação desnecessários que consomem CPU e banda.

5.2 Análise Comparativa de Desempenho: K3s vs. K0s

Os experimentos demonstraram que o K0s apresentou um desempenho superior no cenário proposto de nó único com recursos restritos, processando com sucesso a maioria das requisições, mesmo sob a carga máxima de teste. Por sua vez, o K3s exibiu dificuldades para concluir muitas requisições, resultando frequentemente em tempos de espera esgotados (*timeouts*). Essa diferença pode-se dar, em grande parte, aos serviços adicionais (como Traefik, Service LB e Metrics Server) que são incluídos por padrão na instalação do K3s.

É plausível que parte dessa disparidade de desempenho possa ser mitigada customizando a instalação do K3s para equiparar a quantidade de serviços à do K0s. Outro fator relevante é o modo de instalação otimizado para nó único (single-node) do K0s, um recurso não disponível de

forma nativa na instalação padrão do K3s. Portanto, não se pode afirmar conclusivamente que o K3s seja inferior ao K0s pois não houve um esforço da configuração do primeiro. Contudo, para cenários de implantação em nó único, a simplicidade e a configuração enxuta da instalação padrão do K0s oferecem um caminho mais direto para um melhor desempenho "out-of-the-box".

Agora sob uma ótica estratégica de adoção empresarial, a escolha entre K3s e K0s transcende a mera análise de métricas brutas, refletindo diferentes prioridades de engenharia e maturidade operacional. Organizações que buscam acelerar o *time-to-market* e reduzir a carga cognitiva de suas equipes de DevOps tendem a favorecer o K3s, cuja filosofia entrega uma plataforma operacional imediata com ferramentas integradas de rede, balanceamento de carga e armazenamento. Em contrapartida, empresas com arquiteturas de microsserviços já consolidadas, que priorizam a eficiência granular e a otimização de custos de hardware em larga escala, encontram no K0s uma solução superior. Sua abordagem minimalista permite que os engenheiros desenhem um ecossistema sob medida, selecionando apenas os componentes estritamente necessários para compor a infraestrutura, eliminando o desperdício de recursos computacionais e maximizando a performance do cluster em ambientes de borda onde cada megabyte de memória é crítico.

5.3 Impacto do Meio de Armazenamento no Desempenho

Os testes demonstraram que um COP, principalmente com serviços adicionais ativados, impõe uma demanda significativa sobre o subsistema de disco. O I/O de disco foi o principal gargalo de desempenho do sistema nos testes propostos. No entanto, essa diferença parece estar mais relacionada ao padrão de acesso a disco das COPs — que realizam operações de escrita e leitura intensivas e constantes — do que à velocidade intrínseca do hardware.

Como ambos os cartões SD utilizados possuem especificações de velocidade de leitura similares, não foram observadas mudanças significativas apenas com a troca do cartão. Sugere-se, para trabalhos futuros, a avaliação de tecnologias de armazenamento com barramentos distintos (e.g., SSD via USB ou NVMe, se aplicável) para investigar mais a fundo este fator.

5.4 Influência da Quantidade de *pods* na Carga do Sistema

Conforme a análise de fatores, a variação na quantidade de *pods* (Fator C), quando analisada isoladamente, impactou principalmente as métricas de tempo de uso da CPU (CPU Minutes) e o total de operações de I/O. Este resultado está alinhado com as expectativas teóricas, visto que mais aplicações em execução demandam naturalmente mais ciclos de processamento e acesso a disco para suas operações.

5.5 Conclusões Gerais e Perspectivas Futuras

A utilização de COPs no cenário estabelecido é tecnicamente viável, mas pode requerer otimizações em cenários específicos. Orquestradores são sistemas complexos e em dispositivos de baixa capacidade computacional (como um Raspberry Pi), o *control plane* pode-se tornar o principal fator de impacto no desempenho devido ao consumo elevado de recursos apenas para manter suas funções básicas, conforme ilustrado pelo alto tempo de espera por I/O (IOWait) na Figura 20, especialmente se serviços auxiliares estiverem instalados.

Para sistemas embarcados com baixa capacidade computacional, o K0s em modo single-node demonstrou ser a opção menos custosa, pois ele fornece uma configuração inicial enxuta. Isso permite que sistemas mais simples possam se beneficiar da gestão de aplicações e da resiliência que um COP oferece.

A alta demanda de recursos de um COP justifica por que muitos sistemas embarcados de baixa capacidade ainda recorrem a sistemas operacionais de tempo real (RTOS) com baixíssimo consumo de recursos, como FreeRTOS, Zephyr e QNX. No entanto, existem iniciativas promissoras que buscam trazer um nível de sofisticação similar para a gestão de recursos nesses ambientes. O Zephyr RTOS, por exemplo, já é capaz de executar código compilado com WebAssembly (Wasm), estabelecendo um princípio de conteinerização ao oferecer portabilidade e isolamento. Esse tipo de recurso é particularmente relevante para cenários conectados de IoT, dada a necessidade de atualizações constantes e execução dinâmica de código. Contudo, essas tecnologias emergentes ainda não oferecem o mesmo nível de abstração e a riqueza de funcionalidades de um orquestrador completo como o Kubernetes.

Referências

ALASMARY, H. Scalabledigitalhealth (sdh): An iot-based scalable framework for remote patient monitoring. *Sensors*, v. 24, n. 4, 2024. Cited by: 9; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85185541218&doi=10.3390%2fs24041346&partnerID=40&md5=7179952f67bf0c0d04d8996abd7a7d77>. Citado 7 vezes nas páginas 32, 35, 36, 38, 39, 41 e 44.

ALMEIDA, G. M. et al. Ric-o: Efficient placement of a disaggregated and distributed ran intelligent controller with dynamic clustering of radio nodes. *IEEE Journal on Selected Areas in Communications*, v. 42, n. 2, p. 446–459, Feb 2024. ISSN 1558-0008. Citado 7 vezes nas páginas 32, 35, 39, 40, 41, 43 e 45.

ASCENSÃO, P. et al. Assessing kubernetes distributions: A comparative study. In: IEEE. *2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)*. [S.l.], 2024. p. 832–837. Citado 2 vezes nas páginas 48 e 49.

BAHY, M. B. et al. Resource utilization comparison of kubeedge, k3s, and nomad for edge computing. In: IEEE. *2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. [S.l.], 2023. p. 321–327. Citado 2 vezes nas páginas 49 e 50.

BASILI, V. R.; WEISS, D. M. A methodology for collecting valid software engineering data. *IEEE Transactions on software engineering*, IEEE, n. 6, p. 728–738, 1984. Citado na página 15.

BLANCO, L. et al. A novel approach for scalable and sustainable 6g networks. *IEEE Open Journal of the Communications Society*, v. 5, p. 1673–1692, 2024. ISSN 2644-125X. Citado 6 vezes nas páginas 31, 35, 40, 41, 43 e 45.

BÖHM, S.; WIRTZ, G. Profiling lightweight container platforms: Microk8s and k3s in comparison to kubernetes. In: *ZEUS*. [S.l.: s.n.], 2021. p. 65–73. Citado 2 vezes nas páginas 48 e 49.

BORSATTI, D. et al. Kubetwin: A digital twin framework for kubernetes deployments at scale. *IEEE Transactions on Network and Service Management*, v. 21, n. 4, p. 3889–3903, Aug 2024. ISSN 1932-4537. Citado 4 vezes nas páginas 32, 35, 43 e 45.

BOTEZ, R. et al. Sdn-based network slicing mechanism for a scalable 4g/5g core network: A kubernetes approach. *Sensors*, v. 21, n. 11, 2021. Cited by: 26; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85106739884&doi=10.3390%2fs21113773&partnerID=40&md5=8ba15b751674a9b759ad8e25ce633ad3>. Citado 5 vezes nas páginas 32, 35, 41, 42 e 46.

CAI, Z.; BUYYA, R. Inverse queuing model-based feedback control for elastic container provisioning of web systems in kubernetes. *IEEE Transactions on Computers*, v. 71, n. 2, p. 337–348, Feb 2022. ISSN 1557-9956. Citado 6 vezes nas páginas 18, 32, 35, 38, 39 e 45.

CASALICCHIO, E. Container orchestration: A survey. *Systems Modeling: Methodologies and Tools*, Springer, p. 221–235, 2018. Citado na página 18.

CENTOFANTI, C. et al. Taming latency at the edge: A user-aware service placement approach. *Computer Networks*, v. 247, 2024. Cited by: 8; All Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85191287452&doi=10.1016%2fj.comnet.2024.110444&partnerID=40&md5=ef3bb1a3525ad9a0bbb36f9708c118bb>. Citado 6 vezes nas páginas 21, 33, 35, 36, 37 e 46.

CHAHOUD, M. et al. On-demand-fl: A dynamic and efficient multicriteria federated learning client deployment scheme. *IEEE Internet of Things Journal*, v. 10, n. 18, p. 15822–15834, Sep. 2023. ISSN 2327-4662. Citado 6 vezes nas páginas 32, 35, 38, 39, 43 e 45.

CHAN, Y.-W. et al. Implementation of a cluster-based heterogeneous edge computing system for resource monitoring and performance evaluation. *IEEE Access*, v. 10, p. 38458–38471, 2022. ISSN 2169-3536. Citado 7 vezes nas páginas 18, 31, 35, 38, 41, 42 e 45.

CHAUDHRY, S. R. et al. Improved qos at the edge using serverless computing to deploy virtual network functions. *IEEE Internet of Things Journal*, v. 7, n. 10, p. 10673–10683, Oct 2020. ISSN 2327-4662. Citado 4 vezes nas páginas 31, 35, 42 e 45.

CONTAINERIZATION. 2020. <https://devopedia.org/containerization>. Acessado: 2025-07-22. Citado 2 vezes nas páginas 8 e 18.

CONTINUING our long-standing commitment to Kubernetes and CNCF. 2020. <https://devopedia.org/containerization>. Acessado: 2025-07-22. Citado na página 19.

DALGKITSIS, A. et al. Data driven service orchestration for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, v. 22, n. 7, p. 4100–4109, July 2021. ISSN 1558-0016. Citado 5 vezes nas páginas 24, 31, 35, 38 e 45.

DENG, S. et al. Dependent function embedding for distributed serverless edge computing. *IEEE Transactions on Parallel and Distributed Systems*, v. 33, n. 10, p. 2346 – 2357, 2022. Cited by: 54. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85122313018&doi=10.1109%2fTPDS.2021.3137380&partnerID=40&md5=32d0aacfd914caca53d5c5d744d42c37>. Citado 4 vezes nas páginas 31, 35, 43 e 46.

EDGE Computing. 2019. <https://aprendiendoarduino.wordpress.com/tag/edge-computing>. Acessado: 2025-11-12. Citado 2 vezes nas páginas 8 e 17.

FAYOS-JORDAN, R. et al. Performance comparison of container orchestration platforms with low cost devices in the fog, assisting internet of things applications. *Journal of Network and Computer Applications*, v. 169, 2020. Cited by: 35. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85089210001&doi=10.1016%2fj.jnca.2020.102788&partnerID=40&md5=c3fab5a69b1ab8266b2a16670a15d609>. Citado 6 vezes nas páginas 18, 32, 35, 36, 40 e 44.

FOGLI, M. et al. Performance evaluation of kubernetes distributions (k8s, k3s, kubeedge) in an adaptive and federated cloud infrastructure for disadvantaged tactical networks. In: IEEE. *2021 International Conference on Military Communication and Information Systems (ICMCIS)*. [S.l.], 2021. p. 1–7. Citado 2 vezes nas páginas 48 e 50.

GHASEMI, A.; SCHRANZ, M. Bottom-up resource orchestration in edge computing: An agent-based modeling approach. *International IEEE Conference proceedings, IS*, 2024. Cited by: 1. Disponível em: <https://www.scopus.com/inward/record.uri?eid=

2-s2.0-85208439081&doi=10.1109%2fIS61756.2024.10705177&partnerID=40&md5=f75df76966e8c8d4ef267263711dcb3d>. Citado 6 vezes nas páginas 31, 35, 37, 38, 42 e 46.

GOETHALS, T.; TURCK, F. D.; VOLCKAERT, B. Extending kubernetes clusters to low-resource edge devices using virtual kubelets. *IEEE Transactions on Cloud Computing*, v. 10, n. 4, p. 2623–2636, Oct 2022. ISSN 2168-7161. Citado 8 vezes nas páginas 19, 22, 31, 35, 36, 41, 42 e 45.

IORIO, M. et al. Computing without borders: The way towards liquid computing. *IEEE Transactions on Cloud Computing*, v. 11, n. 3, p. 2820–2838, July 2023. ISSN 2168-7161. Citado 8 vezes nas páginas 19, 21, 22, 31, 35, 40, 42 e 46.

JAIN, R. *The art of computer systems performance analysis*. [S.l.]: john wiley & sons, 1990. Citado na página 53.

JING, Y.; QIAO, Z.; SINNOTT, R. O. Benchmarking container technologies for iot environments. In: IEEE. *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2022. p. 1–8. Citado na página 48.

K0S Requirements. 2025. <https://docs.k0sproject.io/stable/system-requirements/>. Acessado: 2025-11-13. Citado na página 52.

K0S Requirements. 2025. <https://prometheus.io/>. Acessado: 2025-11-13. Citado na página 53.

K0S Requirements. 2025. <https://grafana.com/>. Acessado: 2025-11-13. Citado na página 53.

K3S Requirements. 2025. <https://docs.k3s.io/installation/requirements>. Acessado: 2025-11-13. Citado na página 52.

KAISER, S. et al. Container technologies for arm architecture: A comprehensive survey of the state-of-the-art. *IEEE Access*, v. 10, p. 84853–84881, 2022. ISSN 2169-3536. Citado 10 vezes nas páginas 16, 18, 19, 24, 31, 35, 39, 41, 43 e 45.

KAUR, K. et al. Keids: Kubernetes-based energy and interference driven scheduler for industrial iot in edge-cloud ecosystem. *IEEE Internet of Things Journal*, v. 7, n. 5, p. 4228 – 4237, 2020. Cited by: 115. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85084917974&doi=10.1109%2fJIOT.2019.2939534&partnerID=40&md5=0161b0abbd66f5745743f8be33df8f5f>. Citado 10 vezes nas páginas 13, 19, 23, 32, 35, 36, 37, 38, 41 e 44.

KIM, S.-H.; KIM, T. Local scheduling in kubeedge-based edge computing environment. *Sensors*, v. 23, n. 3, 2023. Cited by: 19; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85147895085&doi=10.3390%2fs23031522&partnerID=40&md5=58fa6a2c9f3fa2c3afbc0d97f7e6e5e0>. Citado 4 vezes nas páginas 32, 35, 37 e 44.

KIM, T. et al. Partition placement and resource allocation for multiple dnn-based applications in heterogeneous iot environments. *IEEE Internet of Things Journal*, v. 10, n. 11, p. 9836–9848, June 2023. ISSN 2327-4662. Citado 2 vezes nas páginas 39 e 42.

KJORVEZIROSKI, V.; FILIPOSKA, S. Kubernetes distributions for the edge: serverless performance evaluation. *Journal of Supercomputing*, v. 78, n. 11, p. 13728 – 13755, 2022. Cited by: 33. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85127088770&doi=10.1007%2fs11227-022-04430-6&partnerID=40&md5=0a102585fbfa22f6af6217317d68e494>. Citado 7 vezes nas páginas 24, 32, 35, 40, 41, 46 e 49.

KOZIOLEK, H.; ESKANDANI, N. Lightweight kubernetes distributions: A performance comparison of microk8s, k3s, k0s, and microshift. In: *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering*. [S.l.: s.n.], 2023. p. 17–29. Citado 2 vezes nas páginas 48 e 49.

KRISTIANI, E. et al. The implementation of a cloud-edge computing architecture using openstack and kubernetes for air quality monitoring application. *Mobile Networks and Applications*, v. 26, n. 3, p. 1070 – 1092, 2021. Cited by: 45. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088006707&doi=10.1007%2fs11036-020-01620-5&partnerID=40&md5=1b0b770d5da493e4afa9112410286c5a>. Citado 8 vezes nas páginas 13, 16, 33, 35, 41, 42, 43 e 44.

KUBERNETES Components. 2025. <https://kubernetes.io/docs/concepts/overview/components/>. Acessado: 2025-11-13. Citado 2 vezes nas páginas 8 e 20.

KUBERNETES usage report. 2022. <https://juju.is/cloud-native-kubernetes-usage-report-2022>. Acessado: 2025-07-22. Citado na página 50.

LAI, W.-K.; WANG, Y.-C.; WEI, S.-C. Delay-aware container scheduling in kubernetes. *IEEE Internet of Things Journal*, v. 10, n. 13, p. 11813–11824, July 2023. ISSN 2327-4662. Citado 7 vezes nas páginas 19, 31, 35, 37, 39, 41 e 45.

LIANG, Q. et al. Model-driven cluster resource management for ai workloads in edge clouds. *ACM Transactions on Autonomous and Adaptive Systems*, v. 18, n. 1, 2023. Cited by: 27; All Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85152618386&doi=10.1145%2f3582080&partnerID=40&md5=151e26e3d54679e00ec32d27b5069ef8>. Citado 7 vezes nas páginas 24, 32, 35, 37, 41, 43 e 44.

LOBATO, E. et al. A monitoring system for electric vehicle charging stations: A prototype in the amazon. *Energies*, v. 16, n. 1, 2023. Cited by: 13; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85145769581&doi=10.3390%2fen16010152&partnerID=40&md5=0478d64be4bd599eae6934a481d2263d>. Citado 3 vezes nas páginas 31, 35 e 46.

LV, W. et al. Microservice deployment in edge computing based on deep q learning. *IEEE Transactions on Parallel and Distributed Systems*, v. 33, n. 11, p. 2968–2978, Nov 2022. ISSN 1558-2183. Citado 7 vezes nas páginas 32, 35, 37, 38, 39, 41 e 44.

MOON, J. et al. Enhancing autonomous driving robot systems with edge computing and ldm platforms. *Electronics (Switzerland)*, v. 13, n. 14, 2024. Cited by: 4; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85199651261&doi=10.3390%2felectronics13142740&partnerID=40&md5=dc38e78ae22798846505fc68411abf8f>. Citado 2 vezes nas páginas 41 e 43.

NGUYEN, K. et al. Parked vehicles task offloading in edge computing. *IEEE Access*, v. 10, p. 41592–41606, 2022. ISSN 2169-3536. Citado 5 vezes nas páginas 37, 38, 39, 41 e 43.

NGUYEN, N. D. et al. Elasticfog: Elastic resource provisioning in container-based fog computing. *IEEE Access*, v. 8, p. 183879–183890, 2020. ISSN 2169-3536. Citado 7 vezes nas páginas 17, 18, 31, 35, 37, 41 e 44.

NGUYEN, Q.-M.; PHAN, L.-A.; KIM, T. Load-balancing of kubernetes-based edge computing infrastructure using resource adaptive proxy. *Sensors*, v. 22, n. 8, 2022. Cited by: 25; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85127706036&doi=10.3390%2fs22082869&partnerID=40&md5=450ed6279c8fa454e5e857d5c3e07132>. Citado 8 vezes nas páginas 24, 32, 35, 37, 38, 40, 42 e 44.

NGUYEN, T.-T. et al. Horizontal pod autoscaling in kubernetes for elastic container orchestration. *Sensors (Switzerland)*, v. 20, n. 16, p. 1 – 18, 2020. Cited by: 118; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85089612460&doi=10.3390%2fs20164621&partnerID=40&md5=0bd64ccc5e149c37898e1b3e4c197267>. Citado na página 24.

PALLEWATTA, S.; KOSTAKOS, V.; BUYYA, R. Microfog: A framework for scalable placement of microservices-based iot applications in federated fog environments. *Journal of Systems and Software*, v. 209, 2024. Cited by: 9; All Open Access, Green Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85179470424&doi=10.1016%2fj.jss.2023.111910&partnerID=40&md5=21b1ceef6e8bb21ec1a6750b388908d6>. Citado 9 vezes nas páginas 13, 22, 23, 32, 35, 39, 41, 42 e 46.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: BCS LEARNING & DEVELOPMENT. *12th international conference on evaluation and assessment in software engineering (EASE)*. [S.l.], 2008. Citado 2 vezes nas páginas 26 e 33.

PHUC, L. H.; PHAN, L.-A.; KIM, T. Traffic-aware horizontal pod autoscaler in kubernetes-based edge computing infrastructure. *IEEE Access*, v. 10, p. 18966–18977, 2022. ISSN 2169-3536. Citado 9 vezes nas páginas 18, 19, 33, 35, 37, 38, 40, 42 e 44.

PéREZ, R. et al. Moving microgrid hierarchical control to an sdn-based kubernetes cluster: A framework for reliable and flexible energy distribution. *Sensors*, v. 23, n. 7, 2023. Cited by: 10; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85152314716&doi=10.3390%2fs23073395&partnerID=40&md5=2045cec82905e5a4331755a172c5b1fd>. Citado 4 vezes nas páginas 21, 38, 41 e 42.

QIAN, Y. et al. The internet of things for smart cities: Technologies and applications. *IEEE Network*, v. 33, n. 2, p. 4–5, 2019. Citado na página 16.

QIAO, Y. et al. Edgeoptimizer: A programmable containerized scheduler of time-critical tasks in kubernetes-based edge-cloud clusters. *Future Generation Computer Systems*, v. 156, p. 221 – 230, 2024. Cited by: 6. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85187808041&doi=10.1016%2fj.future.2024.03.007&partnerID=40&md5=43b58d7ead6f049b547b3ba983f416b8>. Citado 6 vezes nas páginas 31, 35, 37, 41, 42 e 44.

QUAN, P. K.; KUNDROO, M.; KIM, T. Experimental evaluation and analysis of federated learning in edge computing environments. *IEEE Access*, v. 11, p. 33628–33639, 2023. ISSN 2169-3536. Citado 7 vezes nas páginas 31, 35, 38, 41, 42, 43 e 44.

RAC, S.; BRORSSON, M. Cost-aware service placement and scheduling in the edge-cloud continuum. *ACM Transactions on Architecture and Code Optimization*, v. 21, n. 2, 2024. Cited by: 6; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85193585724&doi=10.1145%2f3640823&partnerID=40&md5=f542ce7636e7abb6db64217741418837>. Citado 10 vezes nas páginas 16, 19, 22, 24, 31, 35, 36, 37, 39 e 44.

RAHMAN, A. et al. Security misconfigurations in open source kubernetes manifests: An empirical study. *ACM Transactions on Software Engineering and Methodology*, v. 32, n. 4, 2023. Cited by: 24; All Open Access, Bronze Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85164297169&doi=10.1145%2f3579639&partnerID=40&md5=144a0030b77d8a84791b7b639e578cff>. Citado 6 vezes nas páginas 18, 32, 35, 37, 40 e 45.

REHAN, M. et al. Supply chain management using an industrial internet of things hyperledger fabric network. *Human-centric Computing and Information Sciences*, v. 13, 2023. Cited by: 20. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85166009605&doi=10.22967%2fHCIS.2023.13.004&partnerID=40&md5=518457fdc55b178448d1a2e6122b3bd6>. Citado 4 vezes nas páginas 23, 32, 35 e 46.

RISCO, S. et al. Rescheduling serverless workloads across the cloud-to-edge continuum. *Future Generation Computer Systems*, v. 153, p. 457 – 466, 2024. Cited by: 4; All Open Access, Green Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85180528330&doi=10.1016%2fj.future.2023.12.015&partnerID=40&md5=8aebdd6ce80e280e8f6d6a417614563c>. Citado 2 vezes nas páginas 13 e 21.

ROBLES-ENCISO, A.; SKARMETA, A. F. Adapting containerized workloads for the continuum computing. *IEEE Access*, v. 12, p. 104102–104114, 2024. ISSN 2169-3536. Citado 7 vezes nas páginas 19, 31, 35, 36, 39, 40 e 45.

SEGATTO W.; FREITAS, V. *Parsifal: Perform Systematic Literature Reviews*. 2025. <https://parsif.al/>. Acessado em: 2025-07-22. Citado na página 26.

SLAMNIK-KRIJEŠTORAC, N. et al. Edge-aware cloud-native service for enhancing back situation awareness in 5g-based vehicular systems. *IEEE Transactions on Vehicular Technology*, v. 73, n. 1, p. 660–677, Jan 2024. ISSN 1939-9359. Citado 4 vezes nas páginas 31, 35, 43 e 45.

SOFIA, R. C. et al. A framework for cognitive, decentralized container orchestration. *IEEE Access*, v. 12, p. 79978–80008, 2024. ISSN 2169-3536. Citado 9 vezes nas páginas 19, 31, 35, 36, 38, 39, 40, 42 e 46.

SUBRAMANYA, T.; RIGGIO, R. Centralized and federated learning for predictive vnf autoscaling in multi-domain 5g networks and beyond. *IEEE Transactions on Network and Service Management*, v. 18, n. 1, p. 63–78, March 2021. ISSN 1932-4537. Citado 7 vezes nas páginas 31, 35, 38, 41, 42, 43 e 45.

TRAN, M.-N.; KIM, Y. Optimized resource usage with hybrid auto-scaling system for knative serverless edge computing. *Future Generation Computer Systems*, v. 152, p. 304 – 316, 2024. Cited by: 9. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85177054703&doi=10.1016%2fj.future.2023.11.010&partnerID=40&md5=b4c7707c1ab4ca78b79da7effaec8f52>. Citado 6 vezes nas páginas 24, 32, 35, 38, 43 e 46.

TUSA, F. et al. Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time iot analytics. *Future Generation Computer Systems*, v. 155, p. 204 – 218, 2024. Cited by: 5; All Open Access, Green Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85185531334&doi=10.1016%2fj.future.2024.02.006&partnerID=40&md5=d015b0db6a9d278a03224ac0dbd68823>. Citado 7 vezes nas páginas 13, 22, 24, 32, 35, 42 e 46.

VALANTASIS, A.; MAKRIS, N.; KORAKIS, T. Orchestration software for resource constrained datacenters: an experimental evaluation. In: IEEE. *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. [S.l.], 2022. p. 121–126. Citado 2 vezes nas páginas 48 e 50.

ČILIć, I. et al. Performance evaluation of container orchestration tools in edge computing environments. *Sensors*, v. 23, n. 8, 2023. Cited by: 23; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85153724751&doi=10.3390%2fs23084008&partnerID=40&md5=c787ab3049fb892e0f9a9d78adee4b7f>. Citado 9 vezes nas páginas 32, 35, 36, 37, 39, 40, 41, 44 e 49.

ANEXO A – Benchmark Script

```
NAMESPACE="benchmark"
ITERATIONS=10

calculate_time() {
  local START=$1
  local END=$2
  echo $((END - START))
}

log_with_date() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

benchmark(){
    local DEPLOY_FILE=$1
    log_with_date "Starting benchmark for $DEPLOY_FILE"
    for ((i=1; i<=ITERATIONS; i++)); do
        log_with_date "Starting iteration $i of $ITERATIONS..."

        # Create namespace
        kubectl create namespace $NAMESPACE

        # Record pod creation start time
        START_TIME=$(date +%s)
        # Launch pods using Deployment
        kubectl apply -f $DEPLOY_FILE
        sleep 15

        # Wait for all pods to be ready
        kubectl wait --for=condition=Ready pods --all --namespace=$NAMESPACE
            --timeout=900s

        # Record pod creation end time
        CREATION_END_TIME=$(date +%s)

        # Calculate pod creation latency
        CREATION_LATENCY=$(calculate_time $START_TIME $CREATION_END_TIME)
        log_with_date "Iteration $i - Pod Creation Time: $CREATION_LATENCY seconds"

        # Record pod teardown start time
        TEARDOWN_START_TIME=$(date +%s)

        # Cleanup: Delete namespace and all pods
        kubectl delete namespace $NAMESPACE --wait=true

        # Record pod teardown end time
        TEARDOWN_END_TIME=$(date +%s)

        # Calculate pod teardown latency
```

```
49          TEARDOWN_LATENCY=$(calculate_time $TEARDOWN_START_TIME $TEARDOWN_END_TIME)
50          log_with_date "Iteration $i - Pod Teardown Time: $TEARDOWN_LATENCY seconds"
51          log_with_date "----------------------------------------"
52          sleep 120
53      done
54  }
```

ANEXO B – Deploy file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pod-creation-benchmark
  namespace: benchmark
spec:
  replicas: 4  # Number of pods to create
  selector:
    matchLabels:
      app: test-pod
  template:
    metadata:
      labels:
        app: test-pod
    spec:
      containers:
      - name: nginx
        image: nginx
```