UNIVERSIDADE FEDERAL DO PAMPA

Braulio Marques de Souza

Aprimorando o Provisionamento de Aplicações na Borda com Gerenciamento Adaptativo de Imagens de Contêiner

Alegrete
Julho de 2026

Braulio Marques de Souza

Aprimorando o Provisionamento de Aplicações na Borda com Gerenciamento Adaptativo de Imagens de Contêiner

Dissertação apresentada ao Programa de Pós-graduação Stricto Sensu em Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Fábio Diniz Rossi

Alegrete
Julho de 2026

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

S729a Souza, Braulio Marques de
Aprimorando o Provisionamento de Aplicações na Borda com Gerenciamento Adaptativo de Imagens de Contêiner / Braulio Marques de Souza.
55 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA ELÉTRICA, 2026.
"Orientação: Fábio Diniz Rossi".

1. Computação em Borda. 2. Docker. 3. Desempenho. 4. Provisionamento. I. Título.

BRAULIO MARQUES DE SOUZA

APRIMORANDO O PROVISIONAMENTO DE APLICAÇÕES NA BORDA COM GERENCIAMENTO ADAPTATIVO DE IMAGENS DE CONTÊINER

Dissertação apresentada ao Programa de Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia Elétrica

Dissertação defendida e aprovada em: 06/07/2026

Banca examinadora:

_____

Prof. Dr. Fábio Diniz Rossi

Orientador

IFFar

_____

Prof. Dr. Marcelo Caggiani Luizelli

Unipampa

_____

Prof. Dr. Heleno Carmo Borges Cabral

IFFar

Assinado eletronicamente por **MARCELO CAGGIANI LUIZELLI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/07/2026, às 15:51, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO DINIZ ROSSI**, **PESSOAL VOLUNTÁRIO**, em 14/07/2026, às 16:03, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Heleno Carmo Borges Cabral**, **Usuário Externo**, em 14/07/2026, às 16:37, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2072153** e o código CRC **4B76247C**.

AGRADECIMENTOS

A jornada não foi fácil, pelo contrário, muitas coisas aconteceram durante todo o processo, entre pandemia, perdas e recomeços, mas o que aconteceu faz parte do passado, foi aprendizado, foi dor, marcou, mas já foi, agora só me resta a agradecer a todos que fizeram parte dessa jornada, de uma forma ou de outra, que estiveram presentes, apoiaram, incentivaram, me salvaram em alguns momentos, e foi preciso.

Primeiramente à minha família, meu filho e minha mulher, meus amores e razões de existir, amo vocês, Heitor (meu maior presente na vida) e Sabrina. A dois dos meus salvadores, Tânia e Paulo, por cuidarem do Heitor em diversos, incontáveis momentos, e fazer ele se sentir tão em casa e amado, sem vocês teria sido impossível, muito obrigado, de coração. Ao professor Gustavo, pelo convite que me fez ingressar no programa, apesar de, primeiramente, não ter dado muito certo, minha culpa, obviamente. Aos professores que compartilharam conhecimento e mostraram mais possibilidades na área. Aos colegas de IFFarroupilha, pelo suporte e paciência, Thales, grande parceria, contribuição gigantesca.

E finalmente, à banca de avaliação, professores Marcelo e Heleno, tanto na qualificação quanto na defesa, pelas contribuições e por não engrossar "tanto"o caldo, e em especial ao meu orientador, professor Fábio, sem palavras para agradecer por esse resgate, que foi o que fizestes, pois já estava me dando por vencido quando me puxou de volta e as coisas se alinharam, obrigado de coração, minha eterna (mentira, não somos eternos) gratidão.

Grande abraço e mais uma vez, obrigado.

"Tudo o que temos a decidir é o que fazer com o tempo que nos é concedido."
(Gandalf)

RESUMO

A computação em borda reduz a latência e diminui a dependência da nuvem, tornando-se essencial para infraestruturas de rede modernas. O Docker facilita o deployment de aplicações conteinerizadas em ambientes de borda, mas seu limite padrão de três downloads simultâneos de imagens por cliente pode atrasar o provisionamento quando os recursos de rede estão subutilizados. Este artigo propõe o ADAPT, uma abordagem adaptativa que ajusta dinamicamente o número de downloads paralelos com base nas condições de rede em tempo real. Ao aumentar a concorrência durante períodos de baixo congestionamento e limitá-la durante picos de uso, nosso método otimiza o tempo de provisionamento e melhora o desempenho do sistema. Resultados experimentais extensivos confirmam a viabilidade da proposta e demonstram que o ADAPT supera de forma consistente tanto a configuração padrão do Docker quanto diversas técnicas de ponta para distribuição de imagens de contêiner. Os resultados mostram reduções no tempo de provisionamento de até 33% em cenários realistas de computação em borda, consolidando o ADAPT como uma solução robusta e prática para infraestruturas dinâmicas.

Palavras-chave: Computação em Borda, Docker, Desempenho, Provisionamento.

ABSTRACT

Edge computing reduces latency and decreases dependency on the cloud, making it essential for modern network infrastructures. Docker facilitates containerized application deployment in edge environments, but its default limit of three concurrent images downloads per client can delay provisioning when network resources are underutilized. This paper proposes ADAPT, an adaptive approach that dynamically adjusts parallel downloads based on real-time network conditions. By increasing concurrency during low congestion and limiting it during peak usage, our method optimizes provisioning times and improves system performance. Extensive experimental results confirm the feasibility of the proposed method and demonstrate that ADAPT consistently outperforms both the default Docker configuration and multiple state-of-the-art techniques for container image distribution. The results show provisioning time reductions of up to 33% under realistic edge computing scenarios, reinforcing ADAPT as a robust and practical solution for dynamic infrastructures.

Keywords: Edge computing, Docker, performance, provisioning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta o contexto no qual se insere o problema investigado nesta dissertação, destacando a crescente relevância da computação em borda para aplicações sensíveis à latência e dependentes de respostas em tempo real. Além da literatura internacional recente, trabalhos acadêmicos desenvolvidos no âmbito de programas de pós-graduação brasileiros têm reforçado a importância de mecanismos adaptativos de gerenciamento em ambientes de borda, seja na perspectiva do controle de fluxo [1], da análise de aplicações urbanas sensíveis ao atraso [2] ou da construção de arquiteturas híbridas resilientes entre nuvem e borda [3]. Esse conjunto de evidências ajuda a situar o problema do provisionamento de imagens de contêiner como parte de um desafio mais amplo de eficiência operacional em infraestruturas distribuídas e heterogêneas.

1.1 MOTIVAÇÃO

A computação em borda emergiu como um paradigma fundamental nas infraestruturas de rede modernas, atendendo à crescente demanda por processamento de baixa latência e tratamento localizado de dados [4]. Ao posicionar estrategicamente recursos computacionais mais próximos dos usuários finais, dispositivos IoT e fontes de dados, a computação em borda melhora a responsividade dos sistemas, reduz a dependência de data centers em nuvem remotos e alivia gargalos de largura de banda nas redes centrais [5]. Essa arquitetura distribuída leva a uma escalabilidade aprimorada, maior autonomia dos dispositivos e à viabilização de uma nova classe de aplicações sensíveis à latência, anteriormente impraticáveis sob modelos tradicionais centrados em nuvem [6].

A importância da computação em borda se torna particularmente evidente em domínios como veículos autônomos, automação industrial, monitoramento de saúde e infraestruturas de cidades inteligentes, onde o processamento e a tomada de decisão em tempo real são cruciais [7]. Nesses ambientes, até mesmo pequenos atrasos na transmissão ou no processamento de dados podem resultar em degradação da qualidade do serviço ou riscos à segurança. Ao permitir o processamento de dados no ponto de geração ou próximo a ele, a computação em borda melhora a responsividade das aplicações e reduz o volume de dados que precisa atravessar as redes de backbone, promovendo a eficiência geral do sistema [8].

Essa percepção também aparece em dissertações recentes, que exploram a computação em borda em cenários aplicados e ressaltam a necessidade de respostas rápidas, alocação eficiente de recursos e robustez arquitetural. Sato [2], por exemplo, investiga o uso da borda para análise de imagens em cidades inteligentes, enquanto Silva [3] discute uma arquitetura híbrida nuvem-borda voltada à resiliência em sistemas IoT. Em conjunto, esses trabalhos reforçam que a eficiência das camadas de suporte, entre elas, o provisionamento e a disponibilização de aplicações, exerce papel determinante sobre o desempenho percebido pelo usuário final.

Apesar dessas vantagens, a natureza distribuída e heterogênea da computação em borda impõe desafios significativos à gestão de recursos, especialmente no que diz respeito ao provisionamento de aplicações em nós altamente variáveis e com recursos limitados [9]. A variabilidade nas condições de rede, nas capacidades dos dispositivos e na dispersão geográfica cria um ambiente operacional complexo, onde estratégias tradicionais de provisionamento centralizado muitas vezes se mostram ineficazes.

O Docker [10] tornou-se o padrão de fato para a implantação de aplicações em contêineres em ambientes de borda, graças à sua abordagem leve de virtualização, rápida escalabilidade e simplicidade de orquestração. Ao encapsular aplicações e suas dependências em contêineres isolados e portáteis, o Docker simplifica a implantação em diferentes ambientes de hardware e software, garantindo consistência e confiabilidade mesmo diante da heterogeneidade da infraestrutura subjacente.

Contêineres oferecem diversos benefícios em cenários de borda: permitem o compartilhamento eficiente de recursos, suportam instâncias rápidas e facilitam a migração transparente de aplicações entre nós de borda, características críticas para manter a continuidade do serviço em implantações dinâmicas e móveis [11]. A arquitetura em camadas das imagens do Docker otimiza ainda mais a eficiência da implantação ao permitir atualizações diferenciais, transmitindo apenas camadas novas ou modificadas durante a distribuição de contêineres [12].

Entretanto, as condições de rede dinâmicas e frequentemente limitadas, características dos ambientes de borda, exigem uma gestão cuidadosa e adaptativa da distribuição de imagens de contêiner para manter a estabilidade e a eficiência do sistema. Uma limitação intrínseca do Docker é sua restrição padrão quanto ao número de downloads paralelos de imagens, fixada em três por cliente[1]. Essa limitação é aplicada pelo cliente Docker no nó que executa o *pull*, independentemente de a imagem ser obtida do Docker Hub, de um registro privado, de um espelho local ou de outro registro compatível. Embora essa política conservadora evite congestionamento excessivo durante períodos de alta demanda, como em fases massivas de build ou implantações em larga escala, ela pode se tornar um gargalo de desempenho quando a capacidade de rede disponível está subutilizada.

Em cenários onde os recursos de rede estão disponíveis em abundância, esse limite fixo de concorrência restringe artificialmente a velocidade de provisionamento, atrasando a disponibilidade das aplicações e comprometendo as vantagens de responsividade que a computação em borda busca oferecer [13]. Em contrapartida, quando muitos nós de borda iniciam downloads simultâneos em um enlace compartilhado, o aumento indiscriminado da concorrência pode intensificar filas, perdas e variação de atraso. Um exemplo típico ocorre em infraestruturas de operadoras, nas quais vários dispositivos, estações rádio-base ou servidores de borda compartilham um gargalo de acesso entre a rede de rádio, o nó de borda e o usuário final. Assim, o congestionamento considerado neste trabalho ocorre

[1] https://docs.docker.com/reference/cli/docker/image/pull/

quando o tráfego agregado dos downloads ativos se aproxima ou excede a capacidade disponível desse enlace compartilhado, o mecanismo proposto busca explorar capacidade ociosa sem ultrapassar esse limite operacional. A incapacidade de ajustar dinamicamente o número de downloads simultâneos de acordo com as condições reais da rede compromete a eficiência do provisionamento e, consequentemente, impacta a qualidade do serviço em aplicações sensíveis ao tempo.

1.2 PROBLEMA DE PESQUISA

Em ambientes de computação em borda baseados em contêineres, o provisionamento de aplicações depende, em grande medida, da transferência das camadas das imagens a partir de registros centralizados ou distribuídos para nós geograficamente dispersos. Esse processo é essencial para inicializar serviços, aplicar atualizações e manter a continuidade operacional das aplicações. Entretanto, como os ambientes de borda são marcados por heterogeneidade de dispositivos, conectividade intermitente e variações significativas de largura de banda, a eficiência dessa etapa passa a influenciar diretamente o desempenho do sistema e a experiência percebida pelos usuários.

Nesse contexto, a Figura 1 ilustra um cenário representativo no qual a limitação fixa de downloads simultâneos imposta pelo Docker se torna um fator restritivo para o provisionamento. O cenário assume nós de borda executando clientes Docker e requisitando camadas de imagens a partir de registros remotos ou locais, com a decisão de concorrência aplicada no próprio nó que realiza o download. Mesmo quando há capacidade de rede disponível, o limite padrão de concorrência pode impedir o aproveitamento adequado dos recursos, prolongando o tempo necessário para a obtenção das imagens e, consequentemente, para a disponibilização das aplicações.

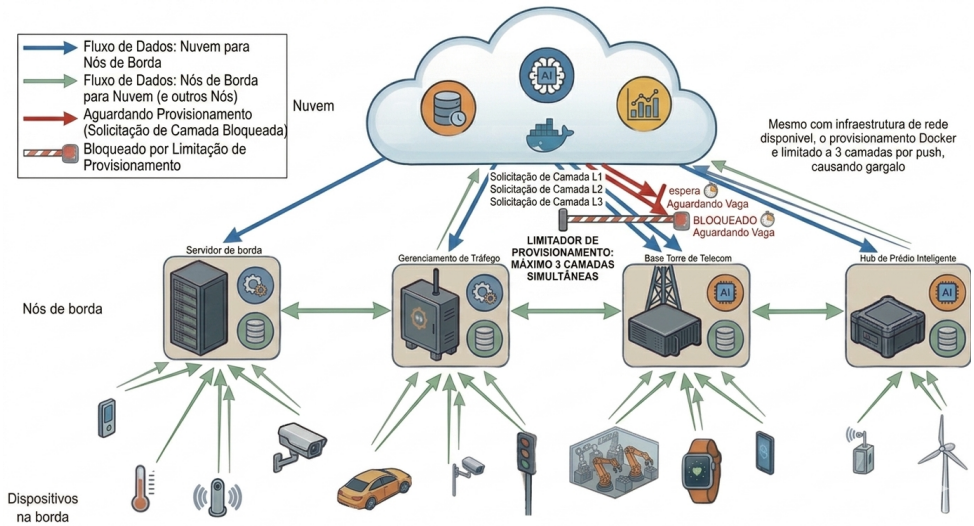

Figura 1 – Exemplo de cenário em que a limitação de downloads simultâneos no Docker gera gargalos no provisionamento de aplicações em ambientes de computação em borda.

O Docker, padrão para gerenciamento de contêineres, simplifica esse processo ao coordenar operações do ciclo de vida das aplicações, incluindo a obtenção e a implantação das imagens. No entanto, a plataforma impõe, por padrão, um limite de três downloads simultâneos por cliente, como forma de evitar sobrecarga da rede durante períodos de maior demanda. Embora essa decisão seja adequada para preservar estabilidade e equidade no uso dos recursos compartilhados, ela deixa de considerar a variabilidade das condições reais de operação na borda.

Embora esse limite conservador contribua para mitigar o risco de congestionamento em condições de tráfego intenso, ele também introduz ineficiências em períodos de baixa utilização da rede. Quando a largura de banda disponível é suficiente e a contenção é reduzida, restringir os clientes a um número fixo e baixo de downloads simultâneos leva à subutilização dos recursos de comunicação. Como consequência, os tempos de implantação são ampliados desnecessariamente, atrasando a disponibilização dos serviços e reduzindo a responsividade esperada em cenários de computação em borda.

Esse descompasso entre uma política estática de concorrência e a natureza dinâmica das condições de rede evidencia as limitações de abordagens baseadas em limiares fixos. Em cenários de borda, nos quais os nós podem experimentar flutuações expressivas de largura de banda em razão da mobilidade, da variação de carga e da heterogeneidade da infraestrutura, limites rígidos de concorrência deixam de refletir o contexto operacional em evolução.

Além disso, em infraestruturas de borda multi-inquilino, nas quais múltiplos serviços e aplicações coexistem e competem por recursos limitados, a ausência de mecanismos adaptativos de download pode agravar a subutilização dos recursos e impactar negativamente a qualidade do serviço. Estratégias de provisionamento mais rápidas e flexíveis são essenciais para viabilizar escalonamento ágil, recuperação de falhas e implantação oportuna de atualizações, capacidades particularmente relevantes para aplicações sensíveis à latência e críticas à missão.

Diante desse cenário, torna-se necessário investigar mecanismos capazes de ajustar dinamicamente os limites de concorrência em função das condições observadas da rede. Uma abordagem dessa natureza tende a aproveitar melhor a largura de banda disponível, reduzir atrasos de provisionamento e aumentar a agilidade operacional das implantações em borda. É precisamente nesse problema que se insere a proposta do ADAPT, concebida para oferecer uma solução leve e responsiva para o gerenciamento adaptativo de downloads de imagens de contêiner em ambientes de borda dinâmicos.

1.3 JUSTIFICATIVA

Para enfrentar essa limitação crítica, este trabalho propõe o ADAPT (**A**daptive **D**ownload **A**cceleration for **P**rovisioning in Edge **T**opologies), uma abordagem dinâmica e sensível à rede para otimização de downloads de imagens de contêiner em ambientes

de borda. O ADAPT monitora métricas de congestionamento em tempo real e ajusta o número de downloads paralelos de forma adaptativa. Durante períodos de baixa utilização da rede, a concorrência é aumentada para explorar a largura de banda disponível e acelerar o provisionamento. Por outro lado, em momentos de congestionamento elevado, o número de downloads paralelos é limitado para preservar a estabilidade da rede.

1.3.1 CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

- uma estratégia adaptativa, sensível ao estado da rede, para ajustar o limite de downloads paralelos de imagens de contêiner;
- uma formalização do problema e do mecanismo de adaptação utilizado pelo ADAPT;
- uma avaliação experimental por simulação, comparando o ADAPT com a configuração padrão do Docker e com abordagens representativas da literatura.

Ao adaptar a concorrência dos downloads com base no estado atual da rede, o ADAPT otimiza o tempo de provisionamento, melhora a utilização de recursos e mantém o desempenho do sistema mesmo em infraestruturas de borda heterogêneas e altamente dinâmicas. Diferentemente de políticas estáticas, o ADAPT assegura que a capacidade de rede seja explorada de forma eficaz quando disponível, sem introduzir instabilidade em períodos de pico. Essa estratégia adaptativa é particularmente vantajosa em deployments de borda nos quais a instância oportuna de aplicações influencia diretamente a disponibilidade do serviço, a experiência do usuário e a eficiência operacional.

Em cenários de borda caracterizados por cargas de rede relativamente baixas ou flutuantes, permitir um maior número de downloads concorrentes melhora substancialmente a eficiência no deployment de aplicações. Ciclos de provisionamento mais rápidos se traduzem em maior continuidade dos serviços, tempos reduzidos de inicialização de aplicações e menores atrasos operacionais, todos aspectos críticos para aplicações sensíveis à latência. Ao alinhar dinamicamente as estratégias de provisionamento com o estado real da rede, o ADAPT representa uma melhoria viável, prática e robusta para o deployment de aplicações containerizadas em ambientes contemporâneos de computação em borda.

2 REFERENCIAL TEÓRICO E TRABALHOS RELACIONADOS

Este capítulo tem por objetivo consolidar a base conceitual que sustenta a proposta desenvolvida nesta dissertação, articulando fundamentos de computação em borda, contêineres, registros de imagens e mecanismos de orquestração com estudos recentes da literatura. Além de artigos e conferências, dissertações de mestrado publicadas recentemente oferecem evidências relevantes sobre desafios de controle de fluxo, balanceamento dinâmico de cargas e uso da borda em aplicações sensíveis à latência [1, 14, 2]. A incorporação desses trabalhos amplia a perspectiva do referencial teórico e contribui para situar o ADAPT em um contexto mais abrangente de pesquisa aplicada em infraestruturas distribuídas.

2.1 REFERENCIAL TEÓRICO

A containerização modernizou o deployment de software ao fornecer uma solução leve, portátil e escalável para empacotamento e execução de aplicações [15]. Diferentemente das máquinas virtuais tradicionais, que exigem uma instância completa de sistema operacional para cada aplicação, os contêineres compartilham o mesmo kernel do sistema hospedeiro. Essa diferença arquitetural reduz significativamente a sobrecarga de recursos e permite a instanciação e o deployment rápidos de aplicações em ambientes computacionais diversos.

A capacidade de isolar aplicações em nível de processo, mantendo um consumo mínimo de recursos, torna os contêineres especialmente adequados para cenários de computação distribuída [16]. Em ambientes de computação em borda, computação em névoa e multicloud, onde os recursos são frequentemente limitados e heterogêneos, os contêineres permitem o deployment e a migração transparente de aplicações, promovendo consistência, resiliência e portabilidade entre diferentes tipos de infraestrutura.

Essa discussão tem sido ampliada por trabalhos acadêmicos recentes que, embora não tratem exatamente do mesmo problema investigado nesta dissertação, ajudam a delimitar o contexto técnico em que ele se insere. Costa [1] mostra, por meio de uma revisão sistemática, que estratégias de controle de fluxo continuam sendo centrais para o bom desempenho de sistemas de borda, especialmente quando a variabilidade de carga e de conectividade interfere diretamente na entrega de serviços. Em paralelo, Carvalho Neto [14] evidencia, no contexto de ambientes Kubernetes, que políticas estáticas de decisão tendem a ser insuficientes diante da dinâmica operacional de infraestruturas distribuídas, reforçando a necessidade de mecanismos mais responsivos e adaptativos.

A adoção de aplicações containerizadas assegura desempenho consistente, ciclos de provisionamento mais rápidos e uso eficiente dos recursos do sistema, tornando os contêineres uma escolha ideal para ambientes dinâmicos, escaláveis e com restrições de recursos. Além disso, os contêineres facilitam a implementação de arquiteturas baseadas em microsserviços, ao decompor aplicações complexas em serviços menores e independentes que podem ser implantados, escalados e atualizados de forma autônoma [17]. Essa modula-

ridade aumenta a flexibilidade, manutenibilidade e escalabilidade dos sistemas modernos, atendendo às exigências de cargas de trabalho altamente dinâmicas e distribuídas.

Nesse contexto, o Docker tornou-se a plataforma mais amplamente utilizada para construção, implantação e gerenciamento de aplicações em contêineres [18]. Ele simplifica o processo de empacotamento de aplicações ao encapsular todas as dependências, configurações e ambientes de execução em contêineres portáteis e autossuficientes, garantindo que o software se comporte de forma consistente nas fases de desenvolvimento, teste e produção.

Sob uma perspectiva aplicada, Sato [2] e Silva [3] também reforçam que a utilidade da computação em borda depende não apenas da proximidade física do processamento, mas da capacidade de disponibilizar serviços com agilidade e previsibilidade em cenários heterogêneos. Essa observação dialoga diretamente com o problema abordado neste trabalho: em aplicações nas quais tempo de resposta e continuidade operacional são fatores críticos, atrasos no provisionamento de imagens de contêiner podem comprometer parte dos ganhos prometidos pela própria adoção da borda.

A criação de um contêiner Docker começa com a escrita de um *Dockerfile*, um script declarativo que especifica a imagem base da aplicação, as bibliotecas necessárias, os parâmetros de execução, as variáveis de ambiente e os comandos de inicialização. Ao executar o comando *docker build*, o Docker gera uma estrutura de imagem em camadas, onde cada instrução no Dockerfile corresponde a uma nova camada imutável empilhada sobre a anterior. Essa abordagem em camadas permite armazenamento e transferência eficientes, pois apenas as camadas modificadas precisam ser atualizadas ou redistribuídas entre sistemas [19].

Uma vez criadas, as imagens de contêiner são armazenadas em um registro Docker, um sistema de armazenamento e distribuição projetado para facilitar o gerenciamento escalável de imagens [9]. Registros públicos, como o Docker Hub, fornecem acessibilidade global. No entanto, muitas organizações que implantam aplicações na borda da rede preferem registros privados e locais, a fim de garantir segurança, reduzir a latência e eliminar dependências de serviços externos. Ao manter o controle sobre o processo de distribuição de imagens, os registros privados ajudam a mitigar riscos relacionados à soberania de dados e interrupções de conectividade.

Os registros Docker organizam as imagens de contêiner em formato de camadas, possibilitando downloads diferenciais, em que apenas camadas ausentes ou atualizadas são recuperadas durante uma operação de *pull*. Esse modelo de distribuição otimiza o consumo de banda, acelera o tempo de deployment e melhora a tolerância a falhas, aspectos especialmente críticos em ambientes de computação em borda, caracterizados por links de rede intermitentes ou com largura de banda limitada [20].

O Docker adota uma arquitetura cliente-servidor, em que o cliente Docker emite comandos via API REST para o daemon Docker, o processo do lado servidor responsável

por gerenciar o ciclo de vida dos contêineres, incluindo sua construção, execução, parada e remoção. Essa arquitetura viabiliza a automação e orquestração de fluxos de trabalho complexos, permitindo o gerenciamento eficiente de deployments em larga escala sem necessidade de intervenção manual [21].

Durante o download de imagens de contêiner a partir de um registro, o daemon Docker busca as camadas necessárias com base na solicitação de *pull* do cliente, garantindo que downloads redundantes sejam evitados por meio de mecanismos locais de cache. Esse design arquitetural aumenta a eficiência operacional e facilita a integração com frameworks de orquestração como o Kubernetes, permitindo altos níveis de escalabilidade e disponibilidade para serviços containerizados [22].

Ao adotar aplicações containerizadas, as organizações podem implementar estratégias de deployment ágeis, resilientes e eficientes no uso de recursos, atendendo à crescente demanda por entrega de aplicações flexível e escalável em cenários computacionais diversos. O fluxo de trabalho simplificado do Docker e seu ecossistema robusto de ferramentas tornam-no uma tecnologia indispensável nas infraestruturas computacionais modernas, que exigem soluções leves, portáteis e escaláveis.

A Figura 2 ilustra a arquitetura central e os fluxos de trabalho do Docker, destacando a interação entre seus principais componentes: *Dockerfile*, imagens, registros e contêineres.

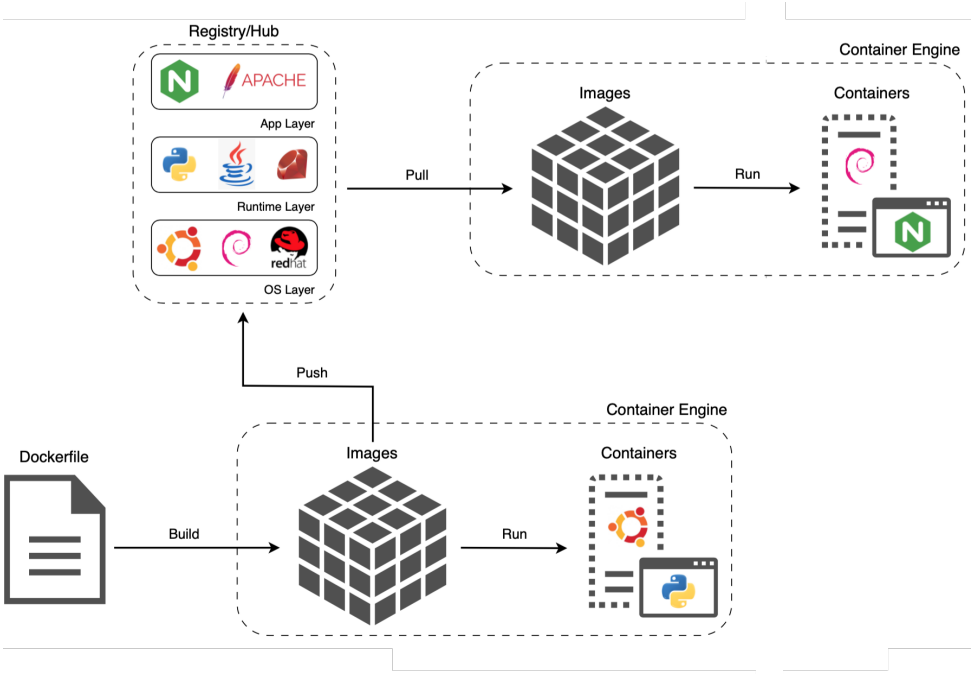

Figura 2 – Infraestrutura Docker.

O processo se inicia com a criação de um *Dockerfile*, um script de configuração que define as dependências da aplicação, os parâmetros de execução e as instruções de construção. Esse arquivo serve como blueprint para a geração de imagens de contêiner

via o comando *docker build*, onde cada instrução adiciona uma nova camada à imagem resultante.

As imagens geradas, compostas por múltiplas camadas que podem incluir sistema operacional base, ambientes de execução, bibliotecas e código da aplicação, podem ser carregadas em um registro Docker. Esses registros, como o Docker Hub, podem ser públicos ou hospedados localmente para melhorar a segurança, a conformidade e o desempenho. A figura também destaca o papel da operação *pull*, na qual as imagens são recuperadas de um registro para o mecanismo de contêiner para execução. Durante esse processo, apenas camadas ausentes são baixadas, otimizando o armazenamento e a eficiência de largura de banda.

Por fim, o motor de contêiner utiliza essas imagens para instanciar e gerenciar contêineres em execução, permitindo o deployment de aplicações de forma leve, isolada e eficiente em ambientes heterogêneos. Esse fluxo de trabalho simplificado exemplifica a capacidade do Docker de abstrair a complexidade subjacente, ao mesmo tempo em que aprimora a escalabilidade, a portabilidade e a agilidade operacional em cenários de computação distribuída.

2.2 TRABALHOS RELACIONADOS

Na computação em borda, o deployment eficiente de aplicações containerizadas é essencial para garantir a continuidade dos serviços, minimizar a latência e otimizar a utilização de recursos. Diversos estudos propuseram métodos inovadores para melhorar a distribuição de imagens de contêineres, abordando desafios semelhantes aos enfrentados pelo ADAPT. Esta seção revisa os trabalhos relacionados, apresentando uma discussão detalhada sobre suas metodologias e contrastando-as com os princípios de projeto e objetivos do ADAPT.

Mou et al. [23] introduziram um algoritmo conjunto de escalonamento de tarefas e cache de imagens para mitigar os atrasos de download de imagens de contêineres em ambientes de computação em borda. A abordagem modela o problema como um Processo de Decisão de Markov (MDP), cujo objetivo é escalonar tarefas computacionais e armazenar em cache imagens de contêineres em nós de borda estratégicos para reduzir a latência total de deployment. Eles utilizam técnicas de aprendizado por reforço profundo para derivar políticas adaptativas que equilibram a priorização de tarefas e o gerenciamento de cache, diminuindo significativamente o tempo de deployment e os atrasos de espera. Embora a solução de Mou et al. apresente excelente desempenho em ambientes com fluxos de tarefas previsíveis e ampla capacidade de armazenamento local, ela pressupõe a disponibilidade de grandes caches e cooperação estável entre os nós, condições nem sempre viáveis em deployments dinâmicos ou com restrições severas de armazenamento.

Azzolini et al. [24] propuseram um framework de raciocínio declarativo para distribuição adaptativa de imagens de contêineres no cloud-edge continuum. O método

utiliza programação lógica para inferir o estado do sistema e otimizar dinamicamente o posicionamento e a migração de imagens de acordo com a disponibilidade de recursos, qualidade da rede e custos de armazenamento. O sistema avalia continuamente políticas de posicionamento para equilibrar eficiência de custo e responsividade no deployment. Embora a abordagem declarativa ofereça alta flexibilidade e adaptabilidade granular, ela impõe uma sobrecarga computacional significativa devido à complexidade da inferência lógica contínua, o que pode limitar sua aplicabilidade em redes de borda de grande escala e com recursos limitados.

Becker et al. [25] desenvolveram o EdgePier, um registro de contêiner totalmente descentralizado que utiliza conexões peer-to-peer (P2P) entre nós de borda para distribuir imagens sem orquestração centralizada. Permitindo que os nós compartilhem diretamente camadas de imagens, o EdgePier reduz substancialmente o tempo de deployment em até 65% quando comparado com registries centralizados tradicionais mesmo em condições de banda restrita. A natureza descentralizada do EdgePier melhora a resiliência e escalabilidade do sistema, mas introduz desafios relacionados à descoberta de peers, consistência e segurança, especialmente em ambientes não confiáveis ou altamente móveis, nos quais a disponibilidade dos pares pode variar imprevisivelmente.

Knob et al. [26] propuseram uma estratégia de posicionamento baseada em comunidades para registros de contêineres, com o objetivo de acelerar o deployment de aplicações em cenários de computação em borda. A abordagem envolve o posicionamento estratégico de registros em comunidades locais de nós de borda, otimizando sua localização para reduzir a latência de rede e melhorar a utilização de recursos dentro de cada comunidade. Ao agrupar inteligentemente os nós e posicionar os registros de forma correspondente, a estratégia reduz a distância média e o tempo necessário para recuperação de imagens. Embora eficaz em topologias relativamente estáveis, a abordagem assume que as estruturas comunitárias são persistentes ao longo do tempo, o que pode limitar sua efetividade em ambientes altamente dinâmicos e sujeitos a mobilidade e churn.

Liang et al. [27] introduziram o HDID (Hybrid Docker Image Distribution), um sistema voltado para data centers de larga escala que integra arquiteturas peer-to-peer e cliente-servidor para melhorar a eficiência da distribuição de imagens. No HDID, os nós podem trocar camadas de imagens diretamente (P2P), recorrendo a servidores centralizados quando necessário. Esse mecanismo híbrido reduz a carga dos servidores, diminui o consumo de banda e acelera a implantação, especialmente durante períodos de alta demanda. Apesar de demonstrar melhorias significativas em contextos de data center, suas suposições quanto à conectividade confiável e alta largura de banda base podem não se generalizar bem às condições mais heterogêneas e voláteis das redes de borda.

Wang et al. [28] apresentaram o FID (Faster Image Distribution), um sistema híbrido de distribuição de imagens otimizado especificamente para a plataforma Docker. O FID aprimora a distribuição P2P tradicional ao incorporar escalonamento inteligente

e técnicas de eliminação de redundância, alcançando transferências de imagem mais rápidas e escaláveis em infraestruturas extensas. Ao ajustar dinamicamente a estratégia de distribuição com base nas condições da rede e na disponibilidade dos peers, o FID melhora o desempenho da entrega de imagens e a escalabilidade geral. No entanto, assim como o HDID, as suposições subjacentes do FID, como a cooperação estável entre peers e conexões de alta vazão, podem ser menos aplicáveis em cenários de computação em borda caracterizados por conectividade intermitente, banda assimétrica e dispositivos com recursos restritos.

Trabalhos mais recentes (2025) reforçam a relevância do problema de distribuição de imagens e do tempo de entrega no *edge*. Deng et al. [29] propõem o PeerSync, um sistema descentralizado (P2P) que adapta decisões de download com base em popularidade e condições de rede em tempo real, visando acelerar a entrega de imagens e reduzir tráfego entre domínios. Em outra direção, Tang et al. [30] exploram explicitamente a estrutura em camadas das imagens para reduzir custo de download e acelerar o deployment por meio de um escalonador sensível a camadas e adaptativo a recursos. Em contraste com essas abordagens, o ADAPT ajusta dinamicamente o número de downloads paralelos de imagens com base nas condições de rede em tempo real, otimizando os tempos de provisionamento e a utilização de recursos sem depender de algoritmos de escalonamento complexos, frameworks declarativos ou registries descentralizados. A Tabela 1 compara as principais características e vantagens dessas abordagens com o ADAPT.

O ADAPT oferece uma solução de baixa complexidade com alta redução no tempo de deployment e grande adaptabilidade à rede, tornando-se uma opção promissora para distribuição eficiente de imagens de contêineres em ambientes de computação em borda.

Tabela 1 – Comparação dos Trabalhos Relacionados com o ADAPT

Abordagem	**Ajuste Dinâmico**	**Complexidade**	**Redução no Tempo de Deployment**	**Adaptabilidade de Rede**
Mou et al. [23]	Sim	Alta	Moderada	Moderada
Azzolini et al. [24]	Sim	Alta	Moderada	Alta
Becker et al. [25]	Não	Moderada	Alta	Baixa
Knob et al. [26]	Não	Moderada	Alta	Baixa
Liang et al. [27]	Não	Moderada	Alta	Baixa
Wang et al. [28]	Não	Moderada	Alta	Baixa
Liu et al. [31]	Não	Alta	Baixa	Alta
Wen et al. [32]	Não	Alta	Baixa	Alta
ADAPT	**Sim**	**Baixa**	**Alta**	**Alta**

3 ADAPT

O ADAPT ajusta dinamicamente o número de downloads simultâneos em cada dispositivo de borda com base em medições em tempo real de congestionamento da rede e da largura de banda disponível, conforme formalizado no Capítulo 3.2 (Gerenciamento Adaptativo de Downloads). A decisão de concorrência é local ao nó de borda que executa o cliente Docker, cada nó estima sua condição de rede, calcula seu limite $D(t)$ e controla quantos *pulls* permanecem ativos ou em espera. Portanto, o ADAPT não pressupõe um orquestrador global, consenso entre nós ou modificação do registro de imagens, em ambientes com Kubernetes, Docker Swarm ou outro orquestrador, esses sistemas continuam responsáveis por decidir onde e quando implantar aplicações, enquanto o ADAPT atua apenas na admissão local dos downloads necessários ao provisionamento. A ideia central é manter o equilíbrio entre a maximização da utilização da rede e a minimização da latência de provisionamento, sem ultrapassar os limites físicos da infraestrutura subjacente, conforme discutido no Capítulo 3.3 (Objetivo da Otimização).

3.1 FORMALIZAÇÃO DO PROBLEMA

Para facilitar a leitura, a Tabela 2 apresenta os principais símbolos, parâmetros e métricas usados neste capítulo e ao longo do Algoritmo 1.

Tabela 2 – Lista de Notações para Descrição do Problema e Algoritmo

Símbolo	**Descrição**
$E(t)$	Conjunto de dispositivos de borda requisitando downloads no instante t
e_i	Dispositivo de borda i
L_i	Conjunto de camadas requeridas pelo dispositivo e_i
l_{ij}	Camada j requisitada pelo dispositivo e_i
$S(l_{ij})$	Tamanho (em bytes) da camada l_{ij}
$d_{ij}(t)$	Estado do download da camada l_{ij} no tempo t (1 = ativo, 0 = pendente/em espera)
$B(t)$	Largura de banda disponível no instante t
$T(t)$	Tráfego total de rede gerado pelos downloads ativos no tempo t
τ	Intervalo de sondagem para estimativa de congestionamento
$D_{\min}$	Limite mínimo de downloads simultâneos por dispositivo (padrão = 3)
$D(t)$	Limite de downloads simultâneos por dispositivo no instante t
$D_{\max}$	Limite superior de concorrência definido por política do sistema
α	Fator de escala (tamanho médio de camada) usado na Equação 3.6
fitness	Valor da função de avaliação da eficiência do provisionamento

3.1.1 MODELAGEM DOS ELEMENTOS DO SISTEMA

Seja $E(t)$ o conjunto de dispositivos de borda que requisitam downloads no instante t:

$$E(t) = \{e_1, e_2, \ldots, e_n\} \tag{3.1}$$

Cada dispositivo e_i requer um conjunto de camadas de imagem de contêiner L_i:

$$L_i = \{l_{i1}, l_{i2}, \ldots, l_{im}\} \tag{3.2}$$

O tamanho de uma camada l_{ij} é denotado por $S(l_{ij})$, correspondente ao tamanho em bytes dos dados a serem transferidos. Em ambientes de borda, essas camadas são frequentemente armazenadas em registros remotos, e seu download constitui uma etapa crítica para o provisionamento de aplicações.

3.1.2 TRÁFEGO DE REDE E CONTROLE DE DOWNLOAD

O tráfego de rede total $T(t)$ gerado por todos os downloads ativos no instante t é:

$$T(t) = \sum_{i=1}^{n} \sum_{j=1}^{m} S(l_{ij}) \cdot d_{ij}(t) \tag{3.3}$$

onde $d_{ij}(t)$ é uma variável binária indicando se a camada l_{ij} está sendo baixada ativamente ($d_{ij} = 1$) ou está em espera na fila ($d_{ij} = 0$). Essa variável permite modelar o paralelismo dos downloads e a contenção por largura de banda.

3.1.3 LIMITAÇÃO DA ABORDAGEM PADRÃO DO DOCKER

Na configuração padrão do Docker, cada dispositivo é limitado por:

$$\sum_{j=1}^{m} d_{ij}(t) \leq 3, \quad \forall i \tag{3.4}$$

onde o número máximo de downloads simultâneos é fixado em três por dispositivo. Essa política, embora simples e robusta para cenários de baixa largura de banda, não considera variações dinâmicas da rede e continua válida independentemente da origem da imagem, pois a restrição é aplicada pelo cliente Docker que realiza o download.

Essa restrição torna-se problemática quando:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} S(l_{ij}) \gg B(t) \tag{3.5}$$

onde $B(t)$ representa a largura de banda disponível no instante t. Em situações com largura de banda abundante, essa limitação estática leva à subutilização dos recursos e ao aumento da latência de provisionamento. Por outro lado, quando a banda é escassa, o limite

fixo pode ser adequado, mas não há mecanismo para reduzir ainda mais a concorrência caso necessário.

3.2 GERENCIAMENTO ADAPTATIVO DE DOWNLOADS

Para lidar com essa ineficiência, propomos ajustar dinamicamente o número máximo de downloads simultâneos $D(t)$ com base em medições em tempo real de congestionamento da rede e disponibilidade de largura de banda. Esse ajuste é executado de forma distribuída e independente em cada nó de borda, sem exigir troca de estado, eleição de líder ou consenso entre dispositivos. Essa escolha reduz o acoplamento operacional e evita que o mecanismo de controle se torne um ponto único de falha, embora limite sua visão ao tráfego observável localmente.

3.2.1 FORMULAÇÃO DA REGRA DE ADAPTAÇÃO

A função de adaptação é definida como:

$$D(t) = \min\left(\max\left(3, \frac{B(t)}{\alpha}\right), D_{\max}\right) \tag{3.6}$$

onde: - α é um fator de escala representando o **tamanho médio** das camadas de imagem, calculado dinamicamente com base nos tamanhos observados dos downloads ($\alpha \approx$ média$(S(l_{ij}))$); - $D_{\max}$ é o limite máximo de concorrência definido pelas políticas do sistema.

O termo $\frac{B(t)}{\alpha}$ fornece uma estimativa de quantos downloads poderiam ser executados simultaneamente considerando a banda atual e o tamanho típico das camadas. Os operadores max e min garantem que o valor final respeite o limite inferior de 3 (compatível com o padrão Docker) e o limite superior $D_{\max}$, evitando sobrecarga.

3.2.2 ESTIMATIVA DINÂMICA DOS PARÂMETROS

Para considerar a variabilidade da rede, α pode ser recalculado periodicamente usando uma média móvel, e opcionalmente complementado com medidas de desvio padrão para estimar picos de tráfego de forma conservadora. Na arquitetura Docker ilustrada na Figura 2, os componentes envolvidos no processo de download correspondem diretamente a essas variáveis: as camadas de imagem (l_{ij}) são obtidas via chamadas REST API entre o daemon Docker (cliente) e os servidores de registro, onde o congestionamento pode afetar a latência do pull.

3.3 OBJETIVO DA OTIMIZAÇÃO

O problema de provisionamento pode ser visto como uma otimização da vazão de dados, respeitando os limites de rede e dos dispositivos. A função objetivo busca maximizar

a quantidade total de bytes transferidos por unidade de tempo.

A otimização do provisionamento pode ser formulada como:

$$\text{maximizar} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}(t) \cdot S(l_{ij}) \tag{3.7}$$

sujeito a:

$$\sum_{j=1}^{m} d_{ij}(t) \leq D(t), \quad T(t) \leq B(t), \quad D(t) \geq 3 \tag{3.8}$$

A primeira restrição limita o número de downloads simultâneos por dispositivo ao valor adaptativo $D(t)$. A segunda garante que o tráfego total não exceda a largura de banda disponível, prevenindo congestionamento. A terceira assegura um patamar mínimo de concorrência, evitando subutilização extrema. Essa formulação permite ao sistema balancear eficiência e respeito aos recursos finitos da rede. Dessa forma, o ADAPT tende a ampliar a concorrência quando identifica capacidade ociosa e a conter novas admissões quando o tráfego agregado se aproxima do gargalo observado, se a estimativa de $B(t)$ for imprecisa ou atrasada, aumentos excessivos de concorrência podem induzir congestionamento temporário, aspecto tratado como limitação e oportunidade de refinamento.

3.4 AVALIAÇÃO DE DESEMPENHO: TEMPO ACORDADO VS. TEMPO ESPERADO

Para avaliar dinamicamente a eficiência dos downloads atuais, o sistema calcula uma métrica de desempenho com base na diferença entre o **tempo acordado de download** e o **tempo esperado de download**.

- *Tempo Acordado*: duração do download negociada em condições estáveis (ex.: calculada com base na largura de banda alocada);
- *Tempo Esperado*: duração do download estimada sob as condições atuais da rede em tempo real.

Essa métrica permite detectar desvios entre o desempenho esperado e o observado, orientando decisões de ajuste fino do mecanismo adaptativo.

A função de desempenho é:

$$\text{fitness}(a, b) = a \cdot \sum_{k=1}^{n} (\text{Tempo Acordado}_k - \text{Tempo Esperado}_k) + b \cdot n \tag{3.9}$$

onde a e b são fatores de peso que controlam o equilíbrio entre a minimização da latência e o controle de fluxo. Valores maiores de a priorizam reduzir a diferença entre os tempos (ou seja, cumprir o acordado), enquanto b incentiva a conclusão de um maior

número de downloads, mesmo que com pequenos atrasos. Essa função pode ser utilizada tanto para diagnóstico quanto para realimentação do algoritmo de adaptação.

3.5 VISÃO GERAL DO ALGORITMO

O ADAPT opera iterativamente sobre o conjunto de dispositivos de borda $E(t)$, garantindo que cada dispositivo adapte dinamicamente seu conjunto de downloads ativos $d_{ij}(t)$ de acordo com as condições instantâneas da rede. O pseudocódigo do ADAPT é apresentado no Algoritmo 1, que integra diretamente as definições formais introduzidas anteriormente.

A Tabela 2 também reúne os símbolos utilizados ao longo do Algoritmo 1. Esses elementos formalizam a representação de dispositivos, camadas de imagem de contêiner, estados de download, parâmetros de rede e métricas de avaliação. Essa unificação torna a descrição do ADAPT mais coesa, ao concentrar em um único ponto as definições necessárias para sua implementação e análise. Os estados de download, o tráfego e a largura de banda são atualizados dinamicamente a cada ponto de decisão para adaptar as estratégias de provisionamento em tempo real.

Algorithm 1 Gerenciamento Adaptativo de Downloads ADAPT

Require: $E(t)$, $D_{\min}$
Ensure: Controle adaptativo de concorrência e provisionamento eficiente de camadas de imagem

for all $e_i \in E(t)$ **do**
    Garantir que $\sum_{l_{ij} \in L_i} d_{ij}(t) \geq D_{\min}$
end for
Calcular a fitness inicial.
while $\exists e_i \in E(t)$ tal que $d_{ij}(t) = 0$ **do**
    for all $e_i \in E(t)$ ordenado por $S(l_{ij})$ crescente **do**
        Selecionar $l_{ij} \in L_i$ tal que $d_{ij}(t) = 0$
        Avaliar caminhos candidatos para l_{ij} com base em $B(t)$
        Atribuir $d_{ij}(t) \leftarrow 1$
        Recalcular a fitness
        if nova fitness melhora **then**
            Confirmar alocação de l_{ij}
            Atualizar $T(t)$ e $B(t)$
        else
            Atribuir $d_{ij}(t) \leftarrow 0$
            Remover e_i da lista de candidatos
        end if
    end for
end while

3.6 FUNÇÃO DE AVALIAÇÃO NO ADAPT

A função de avaliação (fitness) orienta as decisões de download ao avaliar a responsividade do sistema às condições atuais da rede. Relembre que a métrica é definida como:

$$\text{fitness}(a, b) = a \cdot \sum_{k=1}^{n} (\text{Tempo Acordado}_k - \text{Tempo Esperado}_k) + b \cdot n \tag{3.10}$$

onde:

- Tempo Acordado_k é a duração ideal esperada para o fluxo k em condições estáveis;
- Tempo Esperado_k é a duração estimada com base no congestionamento atual;
- n é o número de fluxos ativos;
- a e b são parâmetros de peso que balanceiam latência e concorrência.

Um valor de fitness menor indica uma utilização mais eficiente da rede com menores atrasos. Durante a execução, sempre que o ADAPT considera iniciar um novo download ($d_{ij}(t) \leftarrow 1$), ele avalia o impacto na fitness global. Se houver melhora (redução), o download é autorizado; caso contrário, é adiado.

3.7 CONTROLE DE CONCORRÊNCIA

A cada ponto de decisão, o ADAPT impõe:

$$\sum_{j=1}^{m} d_{ij}(t) \leq D(t) \tag{3.11}$$

onde $D(t)$ é recalculado dinamicamente conforme a Equação 3.6 no Capítulo 3.2, garantindo que a concorrência escale com a largura de banda disponível em tempo real. Ao reequilibrar continuamente as alocações de download com base nas métricas da rede, o ADAPT minimiza atrasos de fila, reduz migrações e acelera o provisionamento de imagens de contêiner sem exigir modificações na arquitetura central do Docker.

3.8 COMPLEXIDADE COMPUTACIONAL

Nesta seção, definimos explicitamente as variáveis de entrada para evitar ambiguidade de notação. Seja $n = |E(t)|$ o número de dispositivos de borda ativos no instante t, e seja p o número de amostras (ou pacotes de sondagem) coletados em cada ciclo de medição.

O custo do **controlador ADAPT** por ciclo de decisão é linear no número de dispositivos monitorados, isto é, $O(n)$, pois o algoritmo percorre os dispositivos ativos

para atualizar estado, aplicar o limite de concorrência e decidir admissões na fila local. Portanto, quando se afirma complexidade linear do método, o parâmetro principal é o número de dispositivos de borda.

Já o custo da **sonda de rede** é $O(p)$ por ciclo de medição, uma vez que depende da quantidade de amostras usada para estimar largura de banda e congestionamento. Como p é tipicamente pequeno e limitado por configuração, sua contribuição prática permanece baixa.

Assim, o custo total por ciclo pode ser expresso por:

$$C_{\text{ciclo}} = O(n + p) \tag{3.12}$$

Na prática, com $p \ll N$ ou com p limitado por janela fixa, o termo dominante é $O(n)$, linear no número de dispositivos de borda.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta a avaliação experimental do ADAPT e discute seus resultados à luz do problema formulado ao longo da dissertação. Para além da comparação direta com métodos representativos da literatura, a análise procura manter coerência com tendências metodológicas observadas em trabalhos acadêmicos recentes, nos quais mecanismos de borda são avaliados em ambientes dinâmicos, com forte preocupação com variabilidade de carga, escalabilidade e realismo experimental [14, 2, 3].

4.1 CENÁRIO DE SIMULAÇÃO

Para avaliar o desempenho do método proposto ADAPT, foi conduzido um conjunto abrangente de simulações utilizando o simulador EdgeSimPy [33]. O ambiente de simulação foi projetado para representar um cenário dinâmico de computação em borda, no qual a topologia da rede segue o modelo de Barabási–Albert [34]. Esse modelo é amplamente utilizado por capturar propriedades realistas de redes complexas, como a formação de hubs e a distribuição não uniforme de conexões.

Além disso, foi incorporado o modelo de mobilidade baseado em caminhos (pathway mobility model) [35], permitindo simular padrões de movimentação mais realistas dos dispositivos de borda, bem como variações dinâmicas na carga de trabalho ao longo do tempo. Essa combinação de topologia e mobilidade contribui para a construção de um ambiente experimental mais próximo de cenários reais, nos quais há heterogeneidade de dispositivos, flutuações de demanda e mudanças contínuas na conectividade da rede.

Para fins de comparação, além da configuração padrão do Docker, que impõe um limite fixo de três downloads simultâneos por cliente, foram simulados seis métodos representativos do estado da arte [23, 24, 25, 26, 27, 28]. Em vez de reproduzir integralmente as arquiteturas originais desses trabalhos, optou-se por modelar seus comportamentos característicos principais. Essa abordagem permitiu realizar uma comparação direta (head-to-head) dentro de uma mesma infraestrutura simulada, garantindo maior controle experimental e consistência entre os cenários avaliados.

Cabe destacar que, para alguns métodos concorrentes, foram necessárias aproximações comportamentais devido à indisponibilidade de implementações completas ou código-fonte. Por exemplo, no método de Mou et al. [23], o ajuste de recursos sensível a custo foi representado de forma simplificada, sem a inclusão de todo o pipeline de otimização, o que pode subestimar sua capacidade de resposta. Já no caso de Wang et al. [28], a política de enfileiramento por prioridade foi implementada sem o modelo estocástico de atraso originalmente proposto, o que pode favorecer ligeiramente suas métricas de latência. Ainda assim, tais simplificações foram cuidadosamente conduzidas para preservar as características essenciais de cada abordagem, mantendo a validade da comparação.

A Tabela 3 apresenta um resumo das principais restrições adotadas na modelagem de cada técnica:

- Docker (padrão): limite fixo de três downloads concorrentes por cliente.
- Mou et al. [23]: uso de cache local com taxa de acerto de 30%, reduzindo proporcionalmente a necessidade de downloads na rede.
- Azzolini et al. [24]: ajuste adaptativo da concorrência a cada cinco downloads, com base no nível de congestionamento observado.
- Becker et al. [25]: aproximadamente 30% dos downloads são realizados por conexões peer-to-peer mais rápidas.
- Knob et al. [26]: metade dos nós acessa registries mais próximos, reduzindo a latência média em cerca de 20%.
- Liang et al. [27]: distribuição híbrida cliente-servidor e peer-to-peer, com taxa de offload de 50% para pares.
- Wang et al. [28]: distribuição híbrida otimizada, com 70% dos downloads sendo descarregados para conexões peer-to-peer.
- **ADAPT**: ajuste dinâmico do nível de concorrência com base em informações em tempo real sobre congestionamento de rede e largura de banda disponível.

Nos experimentos de simulação, as taxas de utilização da infraestrutura foram sistematicamente variadas em 15%, 25%, 40%, 50% e 70%, permitindo avaliar o comportamento das abordagens sob diferentes níveis de carga. Nesta avaliação, a taxa de utilização é representada pela quantidade de aplicações simultâneas a serem provisionadas, os demais elementos da infraestrutura indicam a quantidade fixa de dispositivos e componentes disponíveis no cenário. Essa variação é fundamental para compreender a robustez e a escalabilidade dos métodos em cenários que vão desde baixa até alta demanda.

A configuração detalhada do ambiente de simulação para cada nível de utilização é apresentada na Tabela 3. Observa-se que o número de estações base, servidores de borda e switches permanece fixo, enquanto o número de aplicações cresce progressivamente conforme a taxa de utilização, refletindo o aumento da carga no sistema. Assim, apenas a linha “Aplicações” varia de acordo com a utilização, as demais linhas descrevem a quantidade de dispositivos de rede presentes em todos os cenários.

Os valores apresentados foram definidos de modo a representar um aumento gradual e consistente da carga de trabalho, permitindo observar o comportamento dos métodos tanto em condições de baixa quanto de alta utilização.

As métricas de avaliação consideradas incluem: (i) o tempo de provisionamento, que mede o intervalo necessário para disponibilizar uma aplicação; (ii) o número de migrações realizadas, indicando a frequência de realocação de serviços no sistema; e (iii) a razão do tempo de provisionamento em relação a um baseline, permitindo uma

Tabela 3 – Configuração dos cenários: taxa de utilização representada pela quantidade de aplicações e infraestrutura fixa representada pela quantidade de dispositivos.

	Taxa de Utilização (aplicações)				
Parâmetro	**15%**	**25%**	**40%**	**50%**	**70%**
Estações Base (dispositivos)	100	100	100	100	100
Servidores de Borda (dispositivos)	50	50	50	50	50
Switches (dispositivos)	100	100	100	100	100
Aplicações	40	65	100	130	180

comparação direta da eficiência relativa entre as abordagens. Essas métricas possibilitam uma análise abrangente do desempenho, capturando tanto aspectos de eficiência quanto de adaptabilidade em cenários dinâmicos de computação em borda.

4.2 TEMPO DE PROVISIONAMENTO

A Figura 3 apresenta os resultados de tempo de provisionamento para todas as abordagens testadas, ilustrando como diferentes estratégias afetam a eficiência da implantação sob diferentes condições de carga de rede. Como esperado, a configuração padrão do Docker exibe os maiores tempos de provisionamento em todos os níveis de utilização. O limite fixo de três downloads simultâneos por cliente impõe um gargalo, especialmente sob cargas baixas a moderadas, nas quais os recursos de rede permanecem subutilizados.

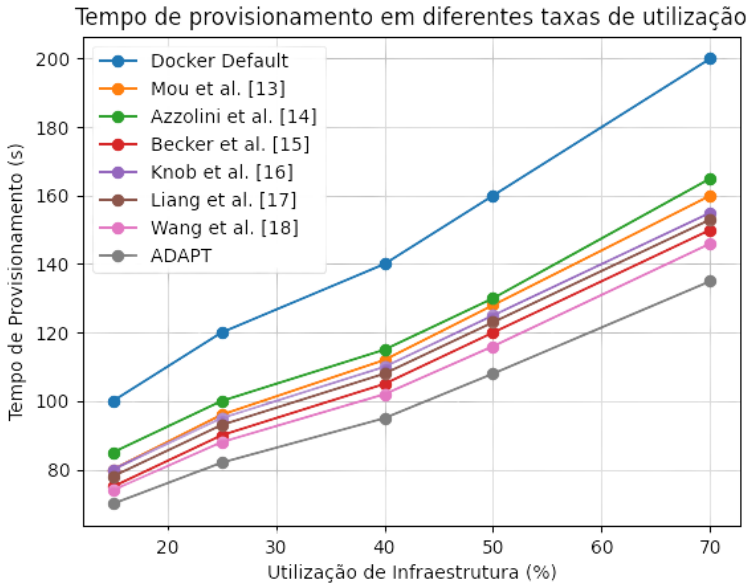

Figura 3 – Tempo de provisionamento sob diferentes taxas de utilização da infraestrutura. Valores menores são melhores.

Entre as abordagens alternativas, Mou et al. [23] e Becker et al. [25] demonstram

reduções substanciais nos tempos de provisionamento. A abordagem de Mou et al. se beneficia de agendamento inteligente de tarefas e cache de imagens, reduzindo a necessidade de downloads repetidos e minimizando o congestionamento de rede. Já o EdgePier de Becker et al. aproveita a distribuição ponto a ponto (P2P), permitindo que nós de borda compartilhem camadas de imagens diretamente, acelerando o provisionamento.

Azzolini et al. [24] obtêm melhorias moderadas por meio de controle de concorrência adaptativo declarativo, ajustando dinamicamente as decisões de colocação de imagens com base em raciocínio lógico sobre estados de recursos e qualidade da rede. Embora eficazes no equilíbrio entre custo e resposta, a sobrecarga computacional associada à avaliação lógica contínua limita os ganhos em comparação com cache e P2P.

Knob et al., Liang et al. e Wang et al. demonstram melhorias notáveis por meio de otimizações arquitetônicas. Knob et al. posicionam estrategicamente repositórios em clusters comunitários, reduzindo o tempo médio de recuperação. HDID de Liang et al. e FID de Wang et al. integram modelos cliente-servidor e ponto a ponto, diversificando os caminhos de aquisição de imagens.

O método ADAPT proposto supera consistentemente todas as abordagens, especialmente sob condições de utilização moderada a alta (25%–70%). O ajuste dinâmico da concorrência, com base em medições de congestionamento em tempo real, permite ao ADAPT explorar a largura de banda disponível de forma mais eficaz que estratégias estáticas ou semi-estáticas. Por exemplo, com 40% de utilização, um ponto crítico onde a contenção de rede começa a impactar significativamente o desempenho, o ADAPT reduziu o tempo de provisionamento em 29% em relação à linha de base do Docker, destacando sua capacidade de adaptação. Além disso, o ADAPT obteve uma redução de 12% em relação a Mou et al. [23], e 7% em relação a Wang et al. [28].

4.3 QUANTIDADE DE MIGRAÇÕES

A Figura 4 mostra a quantidade de migrações, que quantifica o número de transferências de tarefas ou serviços necessárias devido a atrasos no download de camadas de imagens. Em ambientes de borda, migrações são operações custosas, consumindo largura de banda adicional, introduzindo latência e potencialmente interrompendo serviços. Portanto, minimizá-las é fundamental.

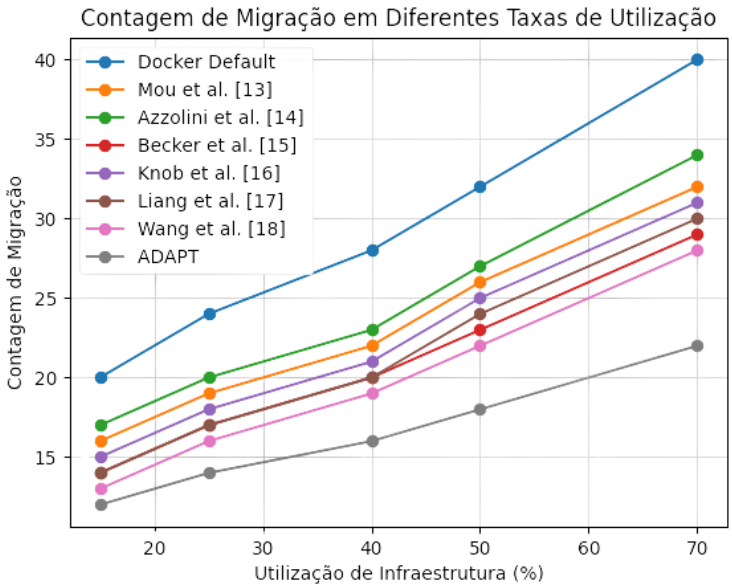

Figura 4 – Contagem de migrações sob diferentes taxas de utilização da infraestrutura. Valores menores são melhores.

Os resultados mostram que abordagens estáticas, incluindo Docker padrão, Mou et al. [23] e Azzolini et al. [24], experimentam significativamente mais migrações à medida que a utilização da infraestrutura aumenta. Sob alta contenção de rede (50% e 70%), a limitação de concorrência rígida e os mecanismos de adaptação menos responsivos causam gargalos no provisionamento de imagens.

Por outro lado, soluções com assistência por pares, como EdgePier [25], HDID [27] e FID [28], mitigam taxas de migração ao distribuir a carga de provisionamento entre múltiplos nós. No entanto, elas introduzem sobrecargas com protocolos de descoberta, sincronização e manutenção de pares, o que pode impactar negativamente recursos limitados.

O ADAPT, ao ajustar dinamicamente a concorrência com base nas condições da rede, mantém baixas contagens de migração mesmo sob alta utilização. Sua capacidade de completar o provisionamento dentro de prazos aceitáveis reduz significativamente a necessidade de migrações de tarefas ou serviços.

4.4 RAZÃO DE TEMPO DE PROVISIONAMENTO

A Figura 5 mostra a razão de tempo de provisionamento para cada abordagem em relação à configuração padrão do Docker. Uma razão abaixo de 1 indica maior eficiência. Essa métrica normalizada fornece uma comparação clara dos ganhos relativos em diferentes cenários.

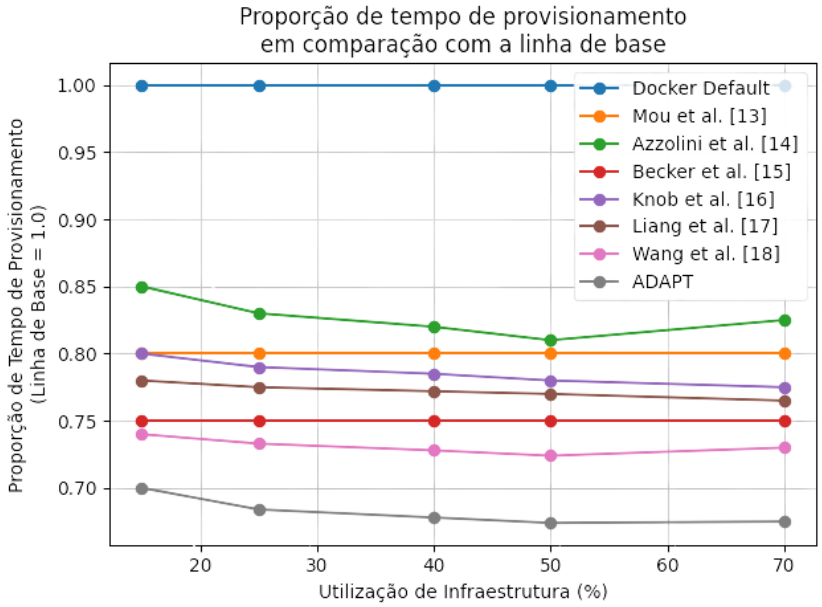

Figura 5 – Razão de tempo de provisionamento em relação ao Docker padrão (menor é melhor). Valor 1.0 indica desempenho igual.

O ADAPT apresenta consistentemente as menores razões em todos os cenários testados, destacando sua superior adaptabilidade. Mesmo sob cargas moderadas (25%–40%), já se observa vantagem. Conforme a utilização aumenta (50%–70%), o ajuste dinâmico do ADAPT torna-se ainda mais essencial.

Mou et al. e Wang et al. também demonstram ganhos substanciais graças ao cache estratégico e à distribuição híbrida, respectivamente. No entanto, o desempenho dessas abordagens se deteriora sob condições de rede adversas. Em contraste, o ADAPT ajusta continuamente a taxa de download, aproveitando a largura de banda quando possível e recuando quando necessário.

4.5 DISCUSSÃO

Os resultados confirmam que, embora abordagens como sistemas ponto a ponto [25, 27, 28] e estratégias baseadas em cache [23] ofereçam melhorias significativas, elas dependem de componentes adicionais, mudanças arquiteturais ou pressupostos ambientais que nem sempre são viáveis em ambientes heterogêneos.

As arquiteturas P2P exigem protocolos de descoberta, estruturas de confiança e disponibilidade estável dos pares. Estratégias de cache dependem da previsibilidade dos padrões de implantação e da disponibilidade de armazenamento. Em ambientes com alta mobilidade ou diversidade, esses requisitos podem não ser atendidos.

O ADAPT, por outro lado, opera sem modificações arquitetônicas nem dependência de pares externos. Ele ajusta dinamicamente o número de downloads concorrentes com

base em medições locais em tempo real. Isso permite sua integração transparente a fluxos Docker, sem sobrecarga extra.

Ao escalar a concorrência de maneira oportunista ou conservadora, conforme necessário, o ADAPT assegura um equilíbrio entre desempenho e estabilidade. Sua adaptabilidade permite manter a qualidade do serviço mesmo sob cenários imprevisíveis, comuns em ambientes de borda, como mobilidade de dispositivos, conectividade intermitente e recursos limitados.

5 CONSIDERAÇÕES FINAIS

Nesta dissertação, foi apresentado o **ADAPT**, uma abordagem adaptativa para o gerenciamento de downloads concorrentes de imagens de contêiner em ambientes de computação em borda. A proposta ajusta dinamicamente o nível de paralelismo em função das condições instantâneas da rede, com o objetivo de reduzir o tempo de provisionamento sem comprometer a estabilidade do enlace. Os resultados obtidos indicam que tal adaptação se mostra especialmente vantajosa em cenários caracterizados por largura de banda limitada e sensibilidade à latência, nos quais o limite fixo adotado pelo Docker tende a subutilizar os recursos disponíveis ou a responder de maneira inadequada às variações de carga.

De modo geral, a avaliação realizada evidenciou que o ADAPT supera de forma consistente a configuração padrão do Docker e apresenta desempenho competitivo em relação às abordagens de comparação consideradas ao longo do trabalho. O resultado mais expressivo foi observado na razão de tempo de provisionamento, o que indica que o ajuste dinâmico da concorrência permite explorar de maneira mais eficiente a largura de banda disponível, sobretudo em situações nas quais há margem para ampliar a taxa de transferência sem induzir congestionamento. Tais achados reforçam que a proposta constitui uma solução promissora para ambientes de borda, ao combinar simplicidade de integração, baixo acoplamento arquitetural e capacidade de adaptação a condições operacionais variáveis.

Não obstante os resultados alcançados, a análise desenvolvida também evidencia limitações que delimitam o escopo desta dissertação e orientam sua continuidade. A efetividade da sondagem em tempo real depende de janelas de medição suficientemente representativas, o que pode se tornar desafiador diante de padrões de tráfego altamente explosivos. Como a decisão do ADAPT é local a cada nó de borda e não envolve consenso entre dispositivos, o mecanismo reduz a complexidade e evita coordenação global, mas não captura integralmente decisões concorrentes tomadas por outros nós sobre o mesmo gargalo compartilhado. Nesses casos, estimativas locais defasadas podem levar a aumentos simultâneos de concorrência e, temporariamente, induzir congestionamento até que novas medições reduzam o limite de downloads. Ademais, embora a sobrecarga computacional da abordagem permaneça reduzida, cenários com recursos extremamente restritos podem demandar um balanceamento mais criterioso entre frequência de monitoramento e responsividade. Soma-se a isso o fato de que a avaliação se concentrou em cargas sintéticas e em um conjunto limitado de rastros reais, além do emprego de aproximações comportamentais para parte das abordagens concorrentes. Nesse contexto, a seção seguinte apresenta os principais desdobramentos capazes de ampliar a validação experimental e de fortalecer a aplicabilidade prática do ADAPT.

5.1 TRABALHOS FUTUROS

Como desdobramento natural deste trabalho, um primeiro eixo de investigação consiste na ampliação da avaliação do ADAPT para cenários mais diversos, contemplando cargas de trabalho mais heterogêneas, volumes maiores de requisições e rastros reais adicionais, de modo a caracterizar com maior precisão seu comportamento em condições operacionais mais amplas. Mostra-se igualmente relevante aprofundar a comparação com abordagens concorrentes por meio de implementações mais completas, quando disponíveis, reduzindo a dependência de aproximações comportamentais e fortalecendo a validade externa dos resultados.

Um segundo eixo refere-se ao refinamento do próprio mecanismo adaptativo. Entre as extensões mais promissoras, destacam-se o uso de suavização EWMA para a estimativa de largura de banda, a substituição do fator representativo do tamanho das camadas por medidas mais robustas, como percentis, e a adoção de estratégias de crescimento gradual da concorrência inspiradas em *slow-start*. Além disso, a incorporação de realimentação baseada na variação do throughput pode tornar as decisões de ajuste mais estáveis em cenários marcados por flutuações rápidas de carga, preservando, ao mesmo tempo, a leveza computacional da abordagem [24, 12, 28, 11].

Por fim, uma linha particularmente relevante para a aplicabilidade prática do ADAPT diz respeito à sua implementação como agente não intrusivo nos nós de borda, integrado ao ecossistema Docker por meio de mecanismos já existentes, como plugins de autorização, *wrappers* do comando `docker` ou componentes de orquestração externa em ambientes Kubernetes [20, 25, 21]. A investigação dessas alternativas de integração, bem como de seus impactos em termos de latência, disponibilidade e robustez operacional, mostra-se fundamental para consolidar o ADAPT como uma solução efetivamente utilizável em infraestruturas de borda dinâmicas e sensíveis à latência.

REFERÊNCIAS

1 COSTA, A. A. F. d. Estratégias de controle de fluxo na computação na borda: uma revisão sistemática. Universidade Federal do Pampa, 2025. Citado 2 vezes nas páginas 21 e 27.

2 SATO, G. M. *Computação na borda para análise de imagens em cidades inteligentes*. Tese (Doutorado) — [sn], 2025. Citado 4 vezes nas páginas 21, 27, 28 e 41.

3 SILVA, L. F. A. d. Arquitetura híbrida nuvem-borda para sistemas de irrigação iot resilientes. 2025. Citado 3 vezes nas páginas 21, 28 e 41.

4 SATYANARAYANAN, M. The emergence of edge computing. *Computer*, v. 50, n. 1, p. 30–39, 2017. Citado na página 21.

5 CHA, J.-G.; KIM, S. W. Design and evaluation of container-based networking for low-latency edge services. In: *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.: s.n.], 2021. p. 1287–1289. Citado na página 21.

6 AVASALCAI, C.; TSIGKANOS, C.; DUSTDAR, S. Decentralized resource auctioning for latency-sensitive edge computing. In: *2019 IEEE International Conference on Edge Computing (EDGE)*. [S.l.: s.n.], 2019. p. 72–76. Citado na página 21.

7 MILADINOVIC, I. et al. Multi-access edge computing: An overview and latency evaluation. In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. [S.l.: s.n.], 2021. v. 1, p. 744–748. Citado na página 21.

8 KISHANI, M.; BECVAR, Z. Reducing computation, communication, and storage latency in vehicular edge computing. In: *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*. [S.l.: s.n.], 2024. p. 1–7. Citado na página 21.

9 TEMP, D. C. et al. Mobility-aware registry migration for containerized applications on edge computing infrastructures. *Journal of Network and Computer Applications*, v. 217, p. 103676, 2023. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804523000954>. Citado 2 vezes nas páginas 22 e 28.

10 FAVA, F. B. et al. Assessing the performance of docker in docker containers for microservice-based architectures. In: *2024 32nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. [S.l.: s.n.], 2024. p. 137–142. Citado na página 22.

11 LAI, W.-P.; LAI, H.-L. A fast and scalable deployment scheme of 5g containerized network functions using docker compose. In: *2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*. [S.l.: s.n.], 2024. p. 821–822. Citado 2 vezes nas páginas 22 e 50.

12 STANISIC, S. et al. Security aspects of container orchestration in kubernetes environments. In: *2025 24th International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2025. p. 1–5. Citado 2 vezes nas páginas 22 e 50.

13 SILVA, V. S. da et al. Synergistically rebalancing the edp of container-based parallel applications. *IEEE Transactions on Parallel and Distributed Systems*, v. 35, n. 3, p. 484–498, 2024. Citado na página 22.

14 NETO, P. M. d. C. et al. Balanceamento dinâmico de pods em ambientes kubernetes. 2024. Citado 2 vezes nas páginas 27 e 41.

15 XAVIER, M. G. et al. Performance evaluation of container-based virtualization for high performance computing environments. In: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. [S.l.: s.n.], 2013. p. 233–240. Citado na página 27.

16 KUMAR, R.; THANGARAJU, B. Performance analysis between runc and kata container runtime. In: *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. [S.l.: s.n.], 2020. p. 1–4. Citado na página 27.

17 MARQUES, W. d. S. et al. Evaluating container-based virtualization overhead on the general-purpose iot platform. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.: s.n.], 2018. p. 00008–00013. Citado na página 27.

18 AGARWAL, S.; JAIN, S.; KUMAR, A. Gui docker implementation: Run common graphics user applications inside docker container. In: *2021 10th International Conference on System Modeling and Advancement in Research Trends (SMART)*. [S.l.: s.n.], 2021. p. 424–427. Citado na página 28.

19 WANG, W. Research on using docker container technology to realize rapid deployment environment on virtual machine. In: *2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC)*. [S.l.: s.n.], 2022. p. 541–544. Citado na página 28.

20 DAS, S. et al. Deduplication of docker image registry. In: *2021 IEEE Madras Section Conference (MASCON)*. [S.l.: s.n.], 2021. p. 1–8. Citado 2 vezes nas páginas 28 e 50.

21 MABOTHA, E.; MABUNDA, N. E.; ALI, A. Performance evaluation of a dynamic restful api using fastapi, docker and nginx. In: *2024 Global Energy Conference (GEC)*. [S.l.: s.n.], 2024. p. 174–181. Citado 2 vezes nas páginas 29 e 50.

22 CHEN, T.; SUO, H. Architecture design of enterprise information system based on docker and devops technology. In: *2023 7th International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS)*. [S.l.: s.n.], 2023. p. 19–23. Citado na página 29.

23 MOU, F. et al. Joint task scheduling and container image caching in edge computing. *arXiv preprint arXiv:2310.00560*, 2023. Citado 8 vezes nas páginas 30, 32, 41, 42, 43, 44, 45 e 46.

24 AZZOLINI, D.; FORTI, S.; IELO, A. Continuous reasoning for adaptive container image distribution in the cloud-edge continuum. *arXiv preprint arXiv:2407.12605*, 2024. Citado 7 vezes nas páginas 30, 32, 41, 42, 44, 45 e 50.

25 BECKER, S.; SCHMIDT, F.; KAO, O. Edgepier: P2p-based container image distribution in edge computing environments. In: IEEE. *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. [S.l.], 2021. p. 1–8. Citado 8 vezes nas páginas 31, 32, 41, 42, 43, 45, 46 e 50.

26 KNOB, L. A. D. et al. Community-based placement of registries to speed up application deployment on edge computing. In: IEEE. *2021 IEEE International Conference on Cloud Engineering (IC2E)*. [S.l.], 2021. p. 147–153. Citado 4 vezes nas páginas 31, 32, 41 e 42.

27 LIANG, M. et al. Hdid: An efficient hybrid docker image distribution system for datacenters. In: SPRINGER. *National Software Application Conference*. [S.l.], 2016. p. 179–194. Citado 6 vezes nas páginas 31, 32, 41, 42, 45 e 46.

28 WANG, K. et al. Fid: A faster image distribution system for docker platform. In: IEEE. *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. [S.l.], 2017. p. 191–198. Citado 8 vezes nas páginas 31, 32, 41, 42, 44, 45, 46 e 50.

29 DENG, Y. et al. Peersync: Adaptive peer-to-peer container image distribution for edge environments. In: *Proceedings of the IEEE International Conference on Edge Computing*. [S.l.: s.n.], 2025. p. 1–10. Citado na página 32.

30 TANG, Z. et al. Layer-resource scheduler for efficient container image delivery in cloud-edge systems. *IEEE Transactions on Cloud Computing*, v. 13, n. 2, p. 1–14, 2025. Citado na página 32.

31 LIU, L. et al. On-device recommender systems: A comprehensive survey. *arXiv preprint arXiv:2206.05395*, 2022. Citado na página 32.

32 WEN, J. et al. A survey on federated learning: challenges and applications. *Complex & Intelligent Systems*, v. 9, p. 1–20, 2023. Citado na página 32.

33 SOUZA, P. S.; FERRETO, T.; CALHEIROS, R. N. Edgesimpy: Python-based modeling and simulation of edge computing resource management policies. *Future Generation Computer Systems*, Elsevier, v. 148, p. 446–459, 2023. ISSN 0167-739X. Citado na página 41.

34 BARABáSI, A.-L.; PóSFAI, M. *Network science*. Cambridge: Cambridge University Press, 2016. ISBN 9781107076266 1107076269. Disponível em: <http://barabasi.com/networksciencebook/>. Citado na página 41.

35 TIAN, J. et al. Graph-based mobility model for mobile ad hoc network simulation. In: *Proceedings of the 35th Annual Simulation Symposium*. USA: IEEE Computer Society, 2002. (SS '02), p. 337. Citado na página 41.