UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Erick Henrique Campolina da Silva

MIGRAÇÃO DE SERVIÇOS EM COMPUTAÇÃO NA BORDA

Belo Horizonte
2024

Erick Henrique Campolina da Silva

MIGRAÇÃO DE SERVIÇOS EM COMPUTAÇÃO NA BORDA

Relatório Final apresentado à Monografia em Sistemas de Informação II, como requisito para a obtenção do título de Bacharel em Sistemas de Informação, da Universidade Federal de Minas Gerais.

Orientador: Daniel Fernandes Macedo

Coorientador: Marcos Magno de Carvalho

Tipo de pesquisa: Mista (científica/tecnológica)

Belo Horizonte
2024

RESUMO

A utilização de dispositivos móveis tem se tornado cada vez maior na sociedade, o que tem gerado um aumento no tráfego de dados nas redes que compõem a internet. Esse aumento pode ocasionar problemas na rede, como o aumento na latência, o que pode ocasionar degradação da qualidade do serviço (QoS) e, consequentemente, da Qualidade de Experiência dos usuários (QoE).

Para mitigar esses problemas, a indústria e a academia têm proposto novas soluções, como a computação na borda, que visa trazer a computação mais próxima do usuário, diminuindo o caminho entre a fonte dos dados (dispositivos) e o processamento. Contudo, mesmo com o desenvolvimento dessa arquitetura e a aproximação da computação ao usuário, outros problemas são enfrentados em relação à QoS e QoE. Por exemplo, diversos servidores presentes nessa arquitetura são distribuídos em localizações geográficas diferentes, o que pode gerar problemas de QoS, como aumento da latência, indisponibilidade/instabilidade dos servidores. Dessa forma é necessário que seja implementado um algoritmo que saiba lidar com esses problemas, para a otimização do sistema.

O objetivo principal deste trabalho é propor uma arquitetura de migração de serviços em ambiente de computação em nuvem, visando a melhoria da QoE dos usuários, aplicada a transmissão de vídeos de realidade virtual em 360 graus. Para isso, foi feito o levantamento de informações em relação à arquitetura de computação na borda e computação em nuvem, assim como pesquisa sobre trabalhos relacionados à otimização de QoE dos usuários.

O sistema proposto realiza tomada de decisão em relação à migração dos serviços. Uma avaliação experimental mostrou que essa arquitetura aumentou em média 24,54% a QoE dos usuários em relação ao sistema de gerenciamento que não considera a migração de serviço.

Palavras-chave: computação na borda, migração de serviços

ABSTRACT

The use of mobile devices has been increasingly prevalent in society. In general, a larger volume of data is being consumed more frequently and continuously. Whether on smartphones, smartwatches, or vehicles (using onboard computers), everything is connected to the internet, generating network congestion and necessitating the development of technologies to manage data consumption. This congestion can also lead to a loss of Quality of Experience (QoE) for users and degradation of the Quality of Service (QoS) provided.

To mitigate these issues, edge computing architecture was developed, aiming to bring computing closer to the edge of the internet, thereby reducing congestion on the main networks.

Despite the development of this architecture and the proximity of computing to the user, other challenges related to user experience are encountered. The various servers in this architecture are distributed across different geographical locations, which can lead to QoS issues, such as increased latency and server unavailability/instability. Therefore, it is necessary to implement an algorithm capable of addressing these issues to optimize the system.

The main objective of this work is to propose a service migration architecture in a cloud computing environment, aimed at improving users' QoE, specifically applied to the transmission of 360-degree virtual reality videos. To achieve this, information was gathered on edge computing and cloud computing architectures, as well as research on related works focused on optimizing users' QoE.

The proposed system makes decisions regarding service migration. An experimental evaluation showed that this architecture increased users' QoE by an average of 24.54% compared to a management system that does not consider service migration.

Keywords: edge computing, service migration

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A computação em nuvem vem sendo amplamente utilizada para a disponibilização de diversos serviços ao redor do mundo. Esse tipo de plataforma fornece recursos sob demanda sem que o provedor de serviço tenha a necessidade de manter servidores físicos por conta própria. Esses recursos podem ser do âmbito computacional (armazenamento, CPU, memória, etc.), de rede, e de software (bancos de dados, ferramentas de análise de dados, modelos de inteligência artificial (IA).

Essas plataformas oferecem alguns benefícios, como, por exemplo, a economia de custo. Ao mover seus serviços para a nuvem, o cliente elimina algumas despesas fixas, como a compra de hardware, software e a contratação de profissionais especializados na área. Outro benefício importante da computação em nuvem é a agilidade no provisionamento automático dos recursos, o que elimina o esforço humano (David, Alves, & Nunes, 2022). Essa agilidade só é possível devido à virtualização de recursos por meio de tecnologias como máquinas virtuais e contêineres.

Por outro lado, o conceito de computação na borda móvel representa uma arquitetura de rede que traz a computação em nuvem para próximo da borda da rede, ou seja, faz com que os serviços sejam executados o mais próximo possível do usuário. A Figura 1 apresenta uma rede ponta-a-ponta, onde o serviço pode ser hospedado na nuvem e/ou na borda. Ao necessitar de um recurso na nuvem, é necessário passar pelo núcleo da rede celular, bem como por toda a internet. Por outro lado, disponibilizando recursos computacionais/rede na borda é possível encurtar esse caminho. Dessa forma diminui-se a latência dos serviços ao passo que isso reduz o congestionamento nas demais partes da rede (núcleo e internet) (Shi, Cao, & Zhang, 2016).

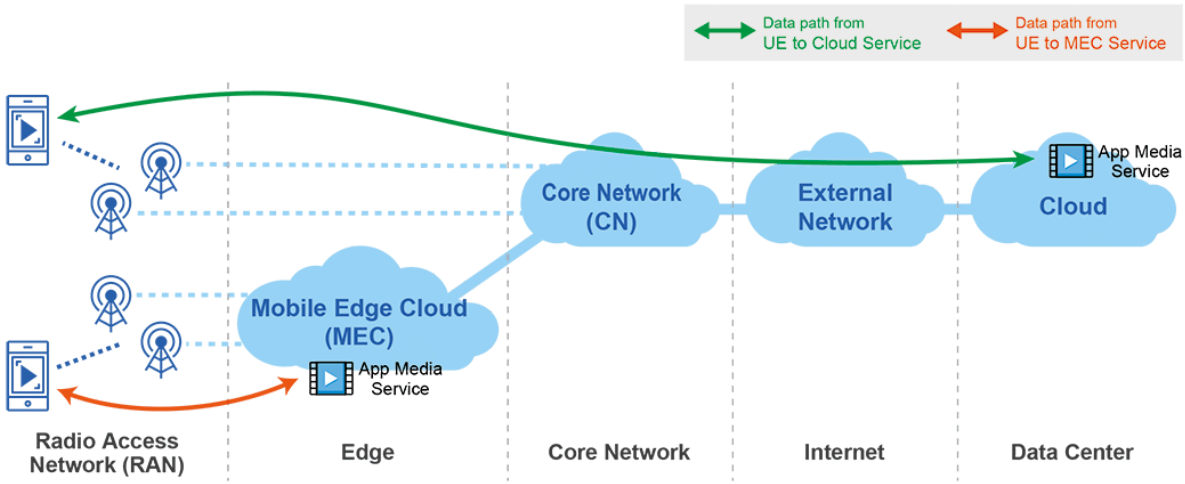

Figura 1 - Arquitetura de Rede (computação em nuvem e computação na borda móvel)

Fonte: Site Devopedia[1]

Essa arquitetura é apropriada para aplicações que demandam baixa latência e alta confiabilidade. Por exemplo, um carro autônomo precisa de informações atualizadas o mais rápido possível para responder aos estímulos que recebe em uma rodovia (como desviar de uma pessoa ou animal) (Liu, Liu, & Tang, 2019). A análise de vídeos de câmeras de segurança também é outro exemplo, sendo necessário o acionamento das autoridades responsáveis em caso de invasão (Machida, Fujiwaka, & Koizumi, 2015).

A latência é uma métrica de Qualidade de Serviço (QoS) que afeta diretamente a Qualidade de Experiência do usuário (do inglês, *Quality of Experience* - QoE). A QoE mede o nível geral de satisfação de um usuário em relação ao serviço/aplicação (Kim, Lee, & Lee, 2010). Portanto, garantir baixa latência melhora a experiência dos usuários.

[1] https://devopedia.org/multi-access-edge-computing

1.1. Motivação

Apesar dos benefícios da computação na borda móvel, ainda existem desafios a serem enfrentados em prol da melhoria da QoE dos usuários. Isso porque existem aspectos da computação móvel que ainda podem degradar a QoS e, consequentemente, a QoE dos usuários (Spinelli, & Mancuso, 2020).

Um desses aspectos é a distribuição geográfica dos servidores para atender as redes de acesso (estação base), conforme ilustrado na Figura 2. Apesar da distribuição dos servidores e das aplicações virtualizadas garantirem uma maior disponibilidade dos serviços, essa prática pode afetar alguns parâmetros de QoS, como a latência. Isso porque a distância física entre esses servidores pode fazer com que o caminho a ser percorrido pelos dados aumente, sendo necessária a passagem por mais nós intermediários, podendo ocorrer aumento no tempo de resposta. Pode ocorrer também instabilidade inesperada em relação à qualidade de conexão dos servidores, afetando também a QoS.

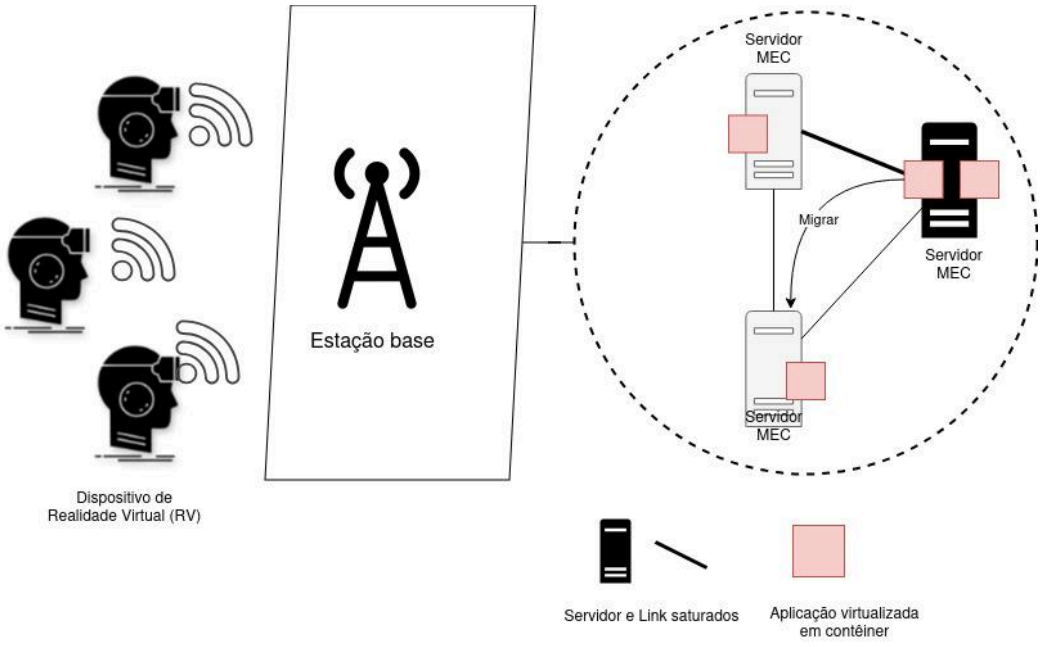

Figura 2: Servidores MEC distribuídos geograficamente em uma determinada região (Chen, Tang, & Xia, 2023 e Rui, Zhang, & Gao, 2021)

Outro fator importante é a limitação de recursos computacionais, quando comparados com os servidores de computação em nuvem (Khan, Baccour, & Chkirbene, 2022). Nesse cenário, quando um determinado servidor/link está saturado, os serviços podem ser migrados para outros servidores (como ilustrado na Figura 2). Com isso, os sistemas de gerenciamento devem conter uma inteligência capaz de avaliar quando migrar os serviços, de forma que a experiência do usuário final seja a melhor possível.

1.2. Objetivo Geral e Objetivo Específicos

O objetivo geral deste trabalho é propor um sistema de migração de serviços na borda para aplicações de transmissão de vídeo de realidade virtual. Neste trabalho, considera-se que a aplicação seja virtualizada em contêiner. Portanto, a migração de serviço é realizada através da transferência dos contêineres entre os servidores. Para alcançar esse objetivo, ele foi separado em alguns objetivos específicos:

- Levantamento de informações sobre computação em nuvem e computação na borda móvel;
- Levantamento de trabalhos relacionados à melhoria da qualidade de experiência do usuário (QoE);
- Desenvolvimento de ambiente experimental;
- Desenvolvimento do sistema de migração e do algoritmo de decisão de migração;
- Avaliação do algoritmo de migração proposto, por meio de experimentação.

1.3. Organização

O restante do trabalho está organizado como se segue. O Capítulo 2 apresenta o referencial teórico, onde são discutidos os principais conceitos utilizados neste trabalho. O Capítulo 3 apresenta os trabalhos correlatos, que foram encontrados a partir de pesquisa na literatura formal. O capítulo 3 capítulo é dividido em duas subseções: A Subseção 3.1 aborda a melhoria de QoE utilizando a migração de serviços, e a Subseção 3.2 aborda essa melhoria sem migração de serviços. O Capítulo 4 apresenta a solução proposta para o problema de migração, discutindo a arquitetura do sistema e o algoritmo proposto. O Capítulo 5 contém a avaliação experimental, explicitando informações do ambiente de avaliação e os resultados obtidos. Por fim, temos o Capítulo 6 que contém a conclusão e trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo contém 3 subseções que discutem o referencial teórico do trabalho. A Subseção 2.1 apresenta informações sobre computação em nuvem, a Subseção 2.1 discute sobre computação na borda e por fim a Subseção 2.3 explicita o referencial teórico relacionado à migração de serviços na borda.

2.1 Computação em nuvem

A computação em nuvem é uma arquitetura que fornece recursos computacionais através da internet. Esses recursos podem ser tanto de *software* quanto de *hardware*, compondo uma infraestrutura utilizada sob demanda por diversos clientes. Esses clientes podem demandar os recursos computacionais/redes a qualquer momento e de qualquer lugar do mundo. Dessa forma, a nuvem precisa atender ao conceito de disponibilidade de serviço, relacionado com a garantia de que os serviços tenham o menor tempo de inatividade possível. Além disso, a computação em nuvem também

deve garantir escalabilidade, sendo capaz de aumentar ou diminuir a provisão de recursos de acordo com a demanda (Jyoti, Shrimali, & Mishra, 2019).

A nuvem também proporciona economia de custo, de forma que os usuários pagam pela utilização de recursos conforme o uso. A provisão desses serviços é fundamental em um acordo de nível de serviço (do inglês, *Service Level Agreement* - SLA), que define os requisitos mínimos que um provedor de nuvem deve garantir aos clientes, especificando o que os usuários podem esperar em termo de disponibilidade, desempenho, e como utilizar os serviços (Jyoti, Shrimali, & Mishra, 2019).

A computação em nuvem é composta por três principais modelos: Infraestrutura como Serviço, Plataforma como Serviço e *Software* como Serviço. No modelo de Infraestrutura como Serviço ocorre os usuários têm acesso a um conjunto de recursos computacionais, como CPU, memória e disco. No modelo de Plataforma como Serviço é oferecido um ambiente de implantação e desenvolvimento de software, fazendo com que o usuário final dessa plataforma não tenha a necessidade de se preocupar com a infraestrutura do serviço. Por fim, o modelo de *Software* como Serviço a aplicação fornecida pode ser acessada direto pelo usuário via internet, sem a necessidade de instalação ou transferência de *software* para o seu dispositivo local. Alguns exemplos de aplicação que utilizam esse modelo são o *Google Docs* (ferramenta de criação de documentos), *Slack* (plataforma de comunicação de equipe), e o *Dropbox* (serviço de armazenamento e compartilhamento de arquivos).

A virtualização é um conceito essencial na computação em nuvem, permitindo criar versões virtuais de sistemas operacionais. Dessa forma é possível que uma única máquina física possa executar diferentes sistemas operacionais, simultaneamente, emulando o comportamento de várias máquinas. Com isso é possível compartilhar o mesmo hardware entre diversas máquinas virtuais, reduzindo o consumo de energia e de recursos computacionais (Ayo, Ajayi, & Okereke, 2017). A virtualização é um conceito chave na computação em nuvem, pois remove o acoplamento entre *hardware* e *software*, de forma que o *software* é encapsulado separadamente do hardware. A conteinerização é uma forma de virtualização ao nível de sistema operacional, diferente da forma comum de virtualização ao nível de hardware. Os contêineres compartilham um único kernel de sistema operacional, operando de forma isolado, permitindo maior

portabilidade em relação a máquinas virtuais. Eles são feitos para virtualizar uma única aplicação e utilizam apenas o básico do sistema operacional para operar, o que os tornam mais leves, funcionando com o mínimo de recursos necessários para executar sua tarefa principal (Ayo, Ajayi, & Okereke, 2017).

2.2 Computação na borda móvel

A computação na borda móvel é uma arquitetura que estende os recursos de computação em nuvem para a borda da rede, especificamente dentro da rede de acesso via rádio (do inglês, *Radio Access Network* - RAN). Ao invés de direcionar o tráfego do usuário final para a internet no geral, essa arquitetura conecta o usuário final diretamente aos serviços localizados na borda da rede. Isso otimiza as requisições dos usuários a uma determinada aplicação ou serviço, o que evita gargalos e falhas de sistema.

Como a operação ocorre isolada do resto da rede, isso faz com que a computação na borda móvel seja menos vulnerável (Haibeh, Yagoub, & Jarray, 2022). Ela também apresenta vantagem ao gerenciar e analisar grandes volumes de dados, sendo interessante o seu uso para aplicações de baixa latência, como aplicações de análise de vídeo em tempo real e transmissão de vídeo em realidade aumentada.

Com o fator de proximidade aos dispositivos dos usuários, ocorre-se a otimização de latência e aumento da largura de banda (Haibeh, Yagoub, & Jarray, 2022). O consumo de energia dos dispositivos também é otimizado nessa arquitetura, já que as tarefas computacionais são migradas para serem executadas em sistemas externos, aumentando a vida útil da bateria desses dispositivos (Xie, Song, & Cao, 2022).

A Figura 3 ilustra como a computação em nuvem e na borda coexistem. A rede em geral é caracterizada por uma arquitetura de três camadas, de forma hierárquica. A primeira camada é a nuvem, a segunda camada é a arquitetura de computação na borda móvel, e a terceira camada é onde se localizam os dispositivos móveis (Abbas, Zhang, & Taherkordi, 2018).

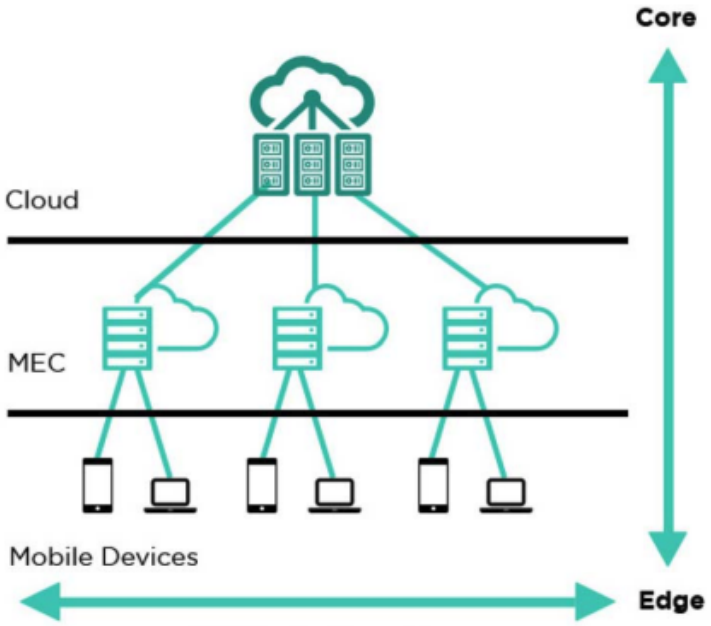

Figura 3 - Arquitetura de três camadas (Abbas, Zhang, & Taherkordi, 2018)

As principais diferenças entre a arquitetura de computação em nuvem e a arquitetura de computação na borda móvel pode ser vista na Tabela 1.

Parâmetro	**Computação em nuvem**	**Computação na borda móvel**
Domínio	Organizações privadas	Organizações de telecomunicação
Capacidade de armazenamento	Ampla	Limitada
Distância do usuário final	Grande	Pequena
Implantação	Rede principal	Borda da rede
Gerenciamento de sistema	Controle centralizado	Controle hierárquico (distribuído)
Conhecimento de localização	Não	Sim
Latência	Alta	Baixa
Atraso de rede	Alto	Muito baixo

Mobilidade	Suporte limitado	Suporte alto
Segurança	Requer plano de segurança menor	Requer maior nível de segurança

Tabela 1 - Principais diferenças entre computação em nuvem e computação na borda móvel (Munmun, Anis, & Hossain, 2022)

Como é possível observar, a computação na borda tem características adequadas para aplicações que demandam baixa latência (baixo atraso de rede e baixa latência). Contudo, essa arquitetura impõe novos desafios, principalmente no gerenciamento dos recursos computacionais e de rede. Isso porque, quando comparado com a computação em nuvem, a computação em borda é composta por diversos servidores distribuídos geograficamente. Essa distribuição, embora garanta uma maior disponibilidade dos serviços/aplicação, traz maiores desafios no gerenciamento dos recursos computacionais e de rede.

2.3 Migração de serviços em computação na borda móvel

Como mencionado anteriormente, a computação na borda móvel é composta por diversos servidores distribuídos geograficamente. Isso significa que, diversos servidores pode estar ligados a uma vasta gama de pontos de acesso/estações-base (*n* servidores atendendo aos *x* pontos de acesso/estações-base) ou serem acoplados diretamente aos seus respectivos pontos de acessos/estações-base (1 para 1) (Haibeh, Yagoub, & Jarray, 2022). O primeiro caso garante uma maior disponibilidade dos serviços, tendo em vista que, através da virtualização, os serviços podem ser escalonados entre diferentes servidores para atender a demanda dos usuários.

Contudo, a distribuição dos servidores em uma determinada região pode ocasionar uma degradação da QoS. Além disso, dado que os servidores da borda têm uma maior limitação de recursos computacionais, o mau gerenciamento desses recursos também pode ocasionar uma degradação da QoS (Spinelli, & Mancuso, 2020). Nesse sentido, a migração de serviços entre os servidores torna-se um mecanismo

para mitigar a degradação da QoS (Kaur, Guillemin, & Francoise, 2022). Neste trabalho, entende-se como migração de serviço, o processo de transportar um determinado serviço virtualizado (contêiner) de um servidor com degradação da QoS para outro servidor em melhor estado.

Apesar dos benefícios da migração de serviços, ainda existem os problemas relacionados ao custo de transmissão de dados. A forma com que se dará a migração de serviços também deve ser avaliada, considerando os requerimentos de QoS e sobrecarga de rede. O objetivo dessa migração é maximizar a QoS após a migração. O problema de migração se torna ainda mais complexo quando são considerados o número elevado de usuários e aplicações que existem em uma rede real, bem como a heterogeneidade dos servidores de borda.

Para uma migração de serviços de computação na borda móvel com melhor desempenho, é necessário um algoritmo eficiente de seleção de servidor na borda. Esse algoritmo deve levar em conta dois fatores: a trajetória do usuário e os parâmetros de QoS. É importante haver um reconhecimento de padrão para predição do movimento do usuário e a observação de parâmetros de QoS (latência de rede, consumo de energia e custo), para ser escolhido o melhor servidor possível (Wang, Xu, & Zhang, 2018).

3 TRABALHOS RELACIONADOS

Para a pesquisa de trabalhos relacionados foi utilizado o Google Scholar. Utilizou-se o seguinte critério para seleção dos artigos: artigos com 10 ou mais citações nos últimos 7 anos, ou artigos com 5 ou mais citações nos últimos 2 anos. Esse critério auxiliou na busca de trabalhos mais atuais e relevantes. Os trabalhos selecionados foram encontrados na revista digital IEEE Xplore Digital Library e na biblioteca ACM Digital Library.

Os trabalhos discutidos a seguir foram separados em duas subseções: A Subseção 3.1 apresenta os trabalhos que propõem melhoria de QoE na borda da rede

sem considerar a migração de serviços entre os servidores. Já a Subseção 3.2 apresenta os trabalhos cujo foco é a migração de serviços para melhorar a QoE dos usuários em diversas aplicações.

3.1 Melhoria de QoE sem empregar a migração de serviços

O objetivo do artigo de (Younis, Tran, & Pompili, 2019), é encontrar a melhor qualidade de vídeo possível a ser transmitido, considerando os recursos de rede disponíveis para cada usuário e o cache de vídeo disponível no servidor de computação na borda, visando a maximização da qualidade de experiência do usuário (QoE) geral. Os autores propõem um novo framework de streaming de vídeos que maximiza a QoE (VQM) para melhorar o sistema de streaming de vídeos sob demanda, levando em conta as propriedades de *Distortion Rate (DR)* de diferentes vídeos.

O termo *Distortion Rate* citado se refere à taxa de distorção utilizada durante o processo de compressão de dados, para diminuir o tamanho dos arquivos trafegados pela internet. Uma DR alta pode resultar em perda de qualidade perceptível na reprodução do vídeo, porém otimiza o tamanho do arquivo, sendo necessário balancear a compensação entre esses dois fatores (qualidade de serviço e qualidade de vídeo).

Um dos desafios para essa implementação é a diversidade de demanda: usuários com dispositivos de alta desempenho (com boa capacidade de armazenamento, processamento e velocidade de conexão) geralmente preferem vídeos de alta resolução, enquanto usuários com dispositivo de baixa desempenho preferem não consumir vídeos em alta resolução, devido aos vídeos não se acomodarem em suas telas, e a alta latência.

A otimização é feita através de um controlador VQM acoplado dentro do servidor de computação na borda, levando em conta as configurações do canal de comunicação para encontrar o *bitrate* ótimo para cada usuário no processo de *transcoding* do vídeo. Quando o usuário faz a requisição do vídeo, existem três caminhos a serem seguidos: se o vídeo já estiver armazenado (cache) no servidor de computação na borda na resolução requerida e a conexão suportar a transferência desses dados, o vídeo é consumido diretamente desse arquivo armazenado; se a conexão não suportar o vídeo

armazenado, o servidor de computação na borda faz o processo de *transcoding*, que consiste na transformação do arquivo original para outro arquivo requerido (resolução, formato, tamanho diferente), e transmite o vídeo em um *bitrate* menor; quando o vídeo não está disponível no servidor de computação na borda, é feita a requisição do arquivo para o servidor de origem com o maior *bitrate* possível. Agora com o vídeo disponível no servidor de computação na borda, é feito o *transcoding* do vídeo e em seguida ocorre a transmissão para o usuário. A QoE nesse caso é melhorada considerando o atraso inicial de início da transmissão do vídeo, e a qualidade média do vídeo transmitido. Esse problema é resolvido com uma formulação *Mixed-Integer Nonlinear Program* (MINLP), e por ser um problema NP-completo ele é decomposto em dois sub-problemas utilizando *Dual-Decomposition Method* (DDM) e relaxamento de Lagrange. Ao final, o desempenho dos servidores de computação na borda é analisado em termos de tempo de processamento de CPU e latência.

No artigo de (Rahman, Hong, & Huh, 2019), é apresentado um método de adaptação de taxa de transmissão, auxiliado pela computação de borda, para uma única estação base com múltiplos clientes. Diferente do trabalho anterior, esse estudo propõe melhorar a QoE do lado do cliente (e não do lado do servidor). É apresentado também um modelo de otimização de programação não-linear inteira para otimizar a experiência de clientes concorrentes. Com isso, é desenvolvido um algoritmo de heurística visando a maximização de QoE, que é feita a partir da seleção do *bitrate* propício para a aplicação cliente.

No artigo de (Xiaodong, Jiaxiang, & Xiaofeng, 2017), é proposto um *schema* de computação na borda de *bitrate* adaptativo otimizado para entrega de vídeos, realizando o cacheamento desses vídeos. O servidor de computação na borda age como um componente controlador para implementar a estratégia de cache, e ajusta a versão de *bitrate* transmitido dos vídeos de forma flexível.

Primeiramente é apresentado o cálculo estimativo da taxa de transmissão de dados, levando em conta o tamanho do vídeo e o tempo de *download* em relação ao cliente. Com isso é possível fazer a seleção do *bitrate* do vídeo ideal a ser transferido. Tudo isso é feito do lado do servidor de computação na borda, fazendo com que o trabalho do lado do usuário seja o menor possível.

O artigo apresenta também o problema de adaptação de *bitrate*. Basicamente, o vídeo é dividido em várias partes, porém a qualidade da conexão pode se alterar durante esse tempo. Isso afeta diretamente a QoE. Suponha que o vídeo esteja sendo transmitido em uma qualidade alta, devido à conexão estar funcionando bem. Se em algum momento ocorrer alguma intervenção, e a qualidade da conexão cair, estaremos ainda enviando um vídeo de alta qualidade para uma conexão ruim. O servidor de computação na borda precisa ser resiliente e saber lidar com esse problema, adaptando o *bitrate* o mais rápido possível. Esse problema é resolvido utilizando um algoritmo guloso que leva em conta os dados obtidos a partir da aplicação cliente.

Para os artigos explicitados nessa seção, o ponto-chave não abordado, e que é um diferencial no trabalho a ser desenvolvido, é a migração de serviços entre servidores de computação na borda, o que resulta em uma melhoria em relação à experiência do usuário.

3.2 Melhoria de QoE empregando a migração de serviços

O artigo de (Ouyang, 2019), propõe um novo mecanismo adaptativo de alocação de serviços gerenciados pelo usuário, que otimiza tanto a latência percebida, quanto o custo de realizar a migração, ponderado pelas preferências do usuário (como a qualidade do vídeo, por exemplo).

Para lidar com a falta de informação e dinamicidade do ambiente de transmissão de vídeo na arquitetura de computação na borda, a solução é formulada como um problema *Multi-armed Bandit* (MAB). A partir dessa formulação é proposto um algoritmo de aprendizado online baseado em amostragem de Thompson para explorar o ambiente dinâmico de computação na borda. O algoritmo faz com que seja possível a tomada de decisões adaptativas de alocação de serviço.

Para manter a QoE satisfatória, os serviços devem ser migrados dinamicamente entre o múltiplos servidores para lidar com o comportamento do usuário (mobilidade e diferentes padrões de requisição dos serviços). A latência do serviço é modelada de acordo com o atraso de tempo de computação (tempo de processamento necessário para cada requisição) e atraso de comunicação (que se refere à latência de acesso e

transferência de dados). O artigo ainda apresenta o termo custo de mudança, que se refere ao custo de migração entre pontos de acesso. Esses pontos podem ser as *base stations* (cada uma com um servidor de borda acoplado utilizando 4G/5G/*Wi-Fi*), o servidor remoto (*cloud*) ou o próprio dispositivo do usuário. Ao final, para fazer a decisão de migração, o problema é formulado como decisão de caminho mínimo em grafos, onde os nós representam cada um dos pontos de acesso acima citados, em determinado tempo *T*, visando minimizar esse custo.

Referente ao tempo de atraso de computação, as tarefas podem ser executadas no dispositivo do usuário, ou em servidores externos (servidores de computação na borda e servidores na nuvem). Como a necessidade de computação pode variar de acordo com a mobilidade do usuário, ou na requisição de vídeos mais acessados, a computação necessária é descrita em termos de ciclos de CPU de acordo com o tempo (a demanda varia de acordo com o tempo). A capacidade de computação de cada dispositivo também é diferente. Portanto, o atraso de computação precisa considerar todas essas variáveis.

Referente ao tempo de atraso de comunicação, ela é dada em termos de latência de acesso e de transferência de dados. Em uma rede de computação na borda complexa com vários dispositivos, existe o atraso referente à distância entre os devidos pontos, e à transferência de informações. A rede também pode ficar congestionada devido ao número de acessos. Todos esses pontos são levados em conta para o cálculo dessa variável.

Referente ao termo custo de mudança podemos ter um custo alto ao transferir os recursos de computação e dados para outro ponto de acesso, quando o usuário se move. Esse custo é levado em conta, então é feito um *trade-off* entre latência percebida e custo de migração. Se a latência percebida não diminuir tanto em relação ao custo de migração desses recursos, então não é necessária a migração.

O artigo de (Taewoon, 2021), apresenta um estudo para migração de serviços na borda em tempo real, em um ambiente virtualizado utilizando a tecnologia de contêineres. Durante o desenvolvimento são discutidas 3 técnicas de migração sem tempo de inatividade, baseado em duplicação e reprodução de estados. Quando os serviços estão sendo migrados, pode ser que o servidor fique inativo por algum

momento até o fim da migração. O termo 'sem tempo de inatividade' aqui faz referência a isso. As técnicas garantem que os servidores continuem respondendo durante a migração. Após isso é proposto o algoritmo ótimo de seleção de migração, que minimiza o tempo de resposta e o tempo de tráfego de dados, durante a migração.

As três técnicas de migração se baseiam em dois métodos: duplicação de estado, e reprodução de estado. Na duplicação de estado, o estado do servidor de origem é copiado para o servidor de destino. Dessa forma, o servidor de destino consegue continuar a execução de onde a máquina de origem parou, de forma eficiente. Por outro lado, a reprodução de estado apenas transmite informações do servidor de origem para o servidor de destino. Dessa forma é possível reproduzir as instruções da máquina de origem, e recriar o estado dentro da máquina de destino.

A primeira técnica é a migração de cópia completa. Essa técnica basicamente utiliza o método de duplicação de estado, transmitindo dados de um servidor ao outro, à medida que o usuário trafega entre pontos de acesso. A segunda técnica é a migração de cópia diferencial. Nessa técnica os dados que são acessados somente por leitura não são migrados, fazendo com que ocorra um ganho de performance, e que diminua a sobrecarga da rede em geral. A terceira técnica é a migração de reprodução de registro. Essa técnica utiliza o método de reprodução de estado e consegue reproduzir o estado da máquina de origem na máquina de destino, reproduzindo os passos do registro fornecido.

O algoritmo de decisão leva em conta o tempo de migração e o nível de sobrecarga que existirá na rede após a migração. O algoritmo se certifica de que o tráfego de dados ocorrerá de forma mais leve com essa decisão. Com esses dados, ele toma a decisão de qual técnica será utilizada.

Nesse artigo a melhoria de QoE está justamente na migração de serviços. Quando o usuário está se movimentando e fica longe do servidor de computação na borda principal, o tempo de resposta pode aumentar consideravelmente. A migração garante essa melhoria de performance.

4 MIGRAÇÃO DE SERVIÇO BASEADO NA LATÊNCIA

Esta seção descreve a arquitetura do sistema proposta para realizar a migração de contêineres em computação na borda. A Subseção 4.1 apresenta a arquitetura desse sistema através da Figura 4, discutindo sobre os principais componentes presentes. Já a Subseção 4.2 apresenta o pseudocódigo do algoritmo implementado.

4.1 Arquitetura do Sistema

A Figura 4 ilustra a arquitetura de sistema proposta. O sistema é separado em dois planos, descritos a seguir.

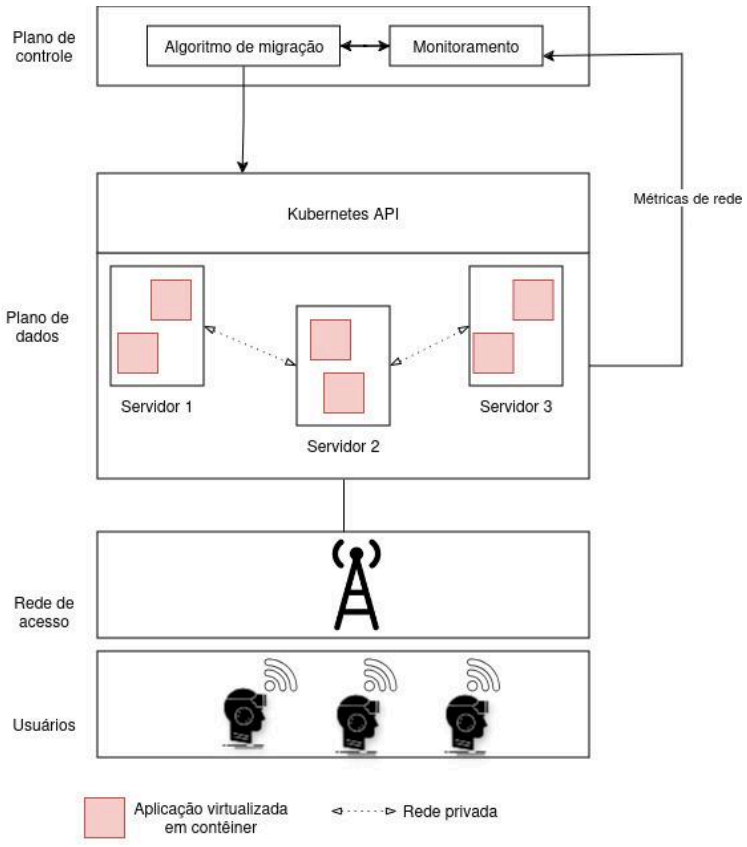

Figura 4 - Arquitetura do sistema proposta

Plano de dados: O plano de dados consiste em um sistema de computação em nuvem, distribuído em diversos servidores com localizações geográficas diferentes. Esses servidores são conectados entre si por uma rede privada. Cada servidor hospeda contêineres da aplicação utilizada no trabalho, onde é realizada a orquestração pelo *Kubernetes*. Cada um desses servidores simboliza um ponto de acesso na computação de borda, tendo múltiplos usuários se conectando a eles de forma simultânea.

Plano de controle: Neste trabalho é proposto um plano de controle separado do utilizado pelo *Kubernetes*, onde temos dois componentes: o **Algoritmo de migração**, que realiza a migração de contêineres para a melhoria de QoE, e o **Monitoramento** responsável pela aquisição de métricas de rede do plano de dados. Ambos componentes estão interconectados, de forma que o **Monitorador** auxilia na tomada de decisão. A API do *Kubernetes* faz a integração entre esses dois planos, garantindo a comunicação de forma eficaz.

4.2 Algoritmo Proposto

A principal contribuição deste trabalho é a realização da migração de serviços entre os conjuntos de servidores disponíveis na borda. Para isso, foi proposto um algoritmo de migração de serviço, apresentado no pseudo Algoritmo 1:

Algoritmo 1 - Algoritmo de decisão de migração de contêineres

```
1: procedure MAIN_MIGRAÇÃO(nodes, nodes_delay, delay_thresold)
2:     good_nodes, bad_nodes = CHECK_NODES(nodes, nodes_delay, delay_thresold)
3:     DELETE_POD_FROM_BAD_NODE(bad_nodes)
4:     MIGRATION(good_nodes)

5: procedure CHECK_NODES(nodes, nodes_delay, delay_thresold)
6:     good_nodes = [ ], bad_nodes = [ ]

7:     for each index, value in nodes_delay
```

```
8:              if value <= delay_thresold
9:                      good_nodes.append(nodes[index])
10:             If value >= delay_thresold
11:                     bad_nodes.append(nodes[index])
12:     return good_nodes, bad_nodes

13: procedure DELETE_POD_FROM_BAD_NODE(bad_nodes)
14:     for each node in bad_nodes:
15:             for each container in node:
16:                     delete container

16: procedure MIGRATION(good_nodes)
17:             coloque os novos contêineres em um dos servidores apropriados (de
forma randômica)
```

O algoritmo funciona como se segue: o módulo monitoramento (Figura 4) coleta as métricas de rede a serem utilizadas no algoritmo. Especificamente, este trabalho considerou somente a latência. Isso significa que a latência entre a rede de acesso e os servidores é levado em consideração na migração do serviço (ver detalhe na Seção 5).

Conforme os dados são coletados e inseridos no algoritmo, faz-se um filtro dos servidores com possível degradação da QoS (latência) e servidores com QoS (delay) apropriado para as aplicações, chamando o método *CHECK_NODES* (linha 2). A separação dos servidores (com degradação e sem degradação da QoS) é feita levando em consideração um limiar de QoS (latência) recebido como parâmetro (*delay_thresold*). Servidores com latência menor ou igual a esse limiar são considerados apropriados, o que significa que podem atender o requisito mínimo de latência para a aplicação. Por outro lado, servidores com latência maior do que o limiar são considerados inapropriados, o que significa que não são considerados aptos para manter os contêineres em operação.

Nesse sentido, quando for detectado servidores com degradação da QoS (abaixo do limiar), o processo de migração é inicialidade (linha 3 e 4). Se existir pelo

menos um servidor apropriado, os *contêineres* alocados em servidores inapropriados são migrados para os servidores apropriados. Essa verificação é muito importante, pois se não existirem nós apropriados, esses *contêineres* não terão onde serem alocados.

Nesse algoritmo pode haver mais de um servidor apropriado, mas nesse caso é escolhido apenas um servidor dentre eles, selecionando o servidor de forma randômica (linha 17 do Algoritmo 1).

5 AVALIAÇÃO EXPERIMENTAL

Essa seção apresenta detalhes de como foi feita a avaliação experimental. A Subseção 5.1 apresenta a descrição do ambiente experimental, com dados técnicos sobre ele. A Subseção 5.1.1 discute sobre como foi feito o processo de coleta de dados. A Subseção 5.1.2 apresenta a metodologia experimental utilizada para avaliação do algoritmo proposto. Por fim, a Subseção 5.2 apresenta os resultados obtidos no experimento.

5.1 Ambiente Experimental

O ambiente experimental foi implementado utilizando a plataforma Fed4FIRE Testbed Portal[2], especificamente, o testbed Virtual Wall II[3]. Nele foram utilizadas 4 máquinas para compor o plano de dados, uma máquina sendo o *master node*, e as outras 3 sendo *worker nodes* (Kubernetes). O ambiente também contém uma máquina para a aplicação cliente (VR-360). Essas máquinas possuem processador *Hexacore Intel E5645 (2.4GHz)*, 24GB de memória RAM e 250GB de disco rígido. O controlador de internet utilizado é o *Intel Corporation 82576 Gigabit Network Connection.*

[2] https://portal.fed4fire.eu/
[3] https://doc.ilabt.imec.be/ilabt/virtualwall/hardware.html#virtual-wall-2

Para os experimentos foi utilizada a aplicação de realidade virtual citada no artigo de (Islam, Rothenberg, & Gomes, 2023). A aplicação cliente realiza a simulação de um usuário assistindo um vídeo em realidade virtual.

5.1.1 Processo de coleta dos dados

A coleta de dados foi realizada variando os parâmetros de rede nos servidores utilizando a ferramenta *tc* do *Linux*[4]. Essa ferramenta permite simular variações de *delay*, *jitter* e perda na rede. Com isso é possível relacionar a variação da QoS com o QoE. É importante mencionar que assumimos que não há degradação da QoS na rede de acesso.

A aplicação de realidade virtual é particionada em azulejos, que são pequenas partes do vídeo em determinado instante de tempo. Cada um desses azulejos pertence a uma zona específica, conforme ilustrado na Figura 5. A Zona 1 possui apenas um azulejo, sendo a visão central do usuário. A Zona 2 contém todos os azulejos adjacentes à Zona 1. A Zona 3 contém o restante dos azulejos.

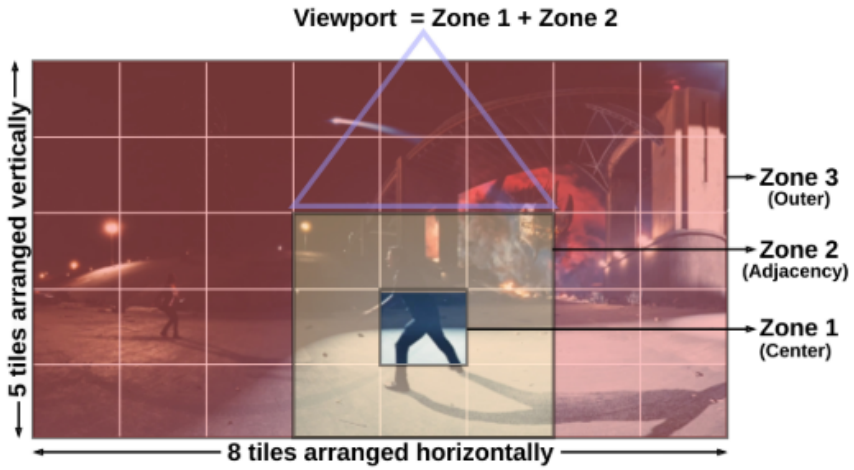

Figura 5 - Exemplo de esquema de 8x5: 3 zonas com centro tendo a mais alta qualidade, seguido por zona 2 e zona 3 (Islam, Rothenberg, & Gomes, 2023)

A Zona 1 possui maior resolução de vídeo por ser a visão central do usuário. A Zona 2 possui resolução de vídeo menor que a Zona 1, porém maior que a Zona 3,

[4] https://man7.org/linux/man-pages/man8/tc.8.html

devido ao fato de ser a zona que contém os azulejos que estão na visão periférica do usuário. Essas zonas mudam conforme a movimentação do campo de visão do usuário.

Ao final de cada sessão de vídeo são geradas métricas da transmissão de vídeo utilizadas para calcular a QoE do usuário, os quais são coletadas nos clientes. Para calcular a QoE dos clientes, utilizaram-se as equações apresentadas na Figura 6.

$$QoE_{\text{per Zone}} = q(R) - \mu \cdot t_{\text{stall}} - \lambda \cdot Q_{\text{switch}} - \omega \cdot T_{\text{startup}} \quad (1)$$

$$Q_{\text{switch}} = \sum_{\forall_z \in all\ Zones, \forall_z \in all\ segments} |q(R^{\text{new}}) - q(R^{\text{old}})| \quad (2)$$

$$QoE_{\text{overall}} = \alpha_1 \cdot QoE_{\text{Zone 1}} + \alpha_2 \cdot QoE_{\text{Zone 2}} + \alpha_3 \cdot QoE_{\text{Zone 3}} \quad (3)$$

Figura 6: Calculo do QoE (Islam, Rothenberg, & Gomes, 2023)

Primeiro, é calculado a QoE por zona (1) e a qualidade de troca de resolução (2). Posteriormente, é calculado a QoE geral (3). Alguns parâmetros importantes são a taxa de transmissão (*bitrate*) médio por zona determinado pela função *q(R)*, o tempo de início da transmissão ($T_{startup}$), o tempo total de parada (T_{stall}) causado por interrupções na transmissão de vídeo devido. Além disso, tem-se a quantidade de trocas de resolução de vídeo por zona (Q_{switch}). Os valores de alfa utilizados no trabalho foram os valores padrão utilizados no artigo ($\alpha_1 = 0,7$; $\alpha_2 = 0,3$; $\alpha_3 = 0$).

Foram definidos alguns limites de valores de QoS para serem coletados. Para o *delay* os limites foram 0ms, 50ms, 100ms e 150ms. Para o *jitter* os limites foram 0ms e 25ms. Para o *loss* os limites foram 0% e 3%. Esses valores foram definidos para simular uma com degradação extrema. Dessa forma foram realizados experimentos entre a combinação desses valores, resultando em 16 combinações.

Para cada uma dessas combinações foram executados 5 clientes de forma simultânea, simulando usuários assistindo ao vídeo de realidade virtual. Cada um desses clientes executou 834 experimentos. Dessa forma foram obtidas 66.720 instâncias desses dados para o cálculo da QoE. Todos os dados coletados se encontram no GitHub[5].

[5] https://github.com/erickhcs/tcc

5.1.2 Metodologia experimental para avaliação do algoritmo

Para a avaliação do algoritmo de migração foram gerados 5 valores diferentes de *delay* para cada *worker node*, caracterizando 5 diferentes configurações do ambiente. Essas configurações foram definidas para simular um ambiente real, onde existem valores de *delay* diferentes para cada servidor.

Esses valores foram obtidos de forma aleatória, porém dentro da faixa de 0ms a 150ms (faixa utilizada para a coleta de dados na Subseção 5.1.1). Embora os valores podem estar alto para uma arquitetura de computação na borda, esses valores podem refletir casos extremos ocorridos por diversos fatores, como a falta de recursos computacionais.

Na avaliação experimental, considerou-se 50ms como um limiar aceitável para a aplicação de realidade virtual. Para as configurações 1 e 5 foram apresentados ambientes onde dois *worker nodes* estão abaixo desse limiar (50ms), dessa forma é possível verificar como o algoritmo se comporta ao encontrar dois *worker nodes* apropriados. As configurações 3 e 4 apresentam ambientes onde nenhum dos *delays* estão abaixo do limite de 50ms. A Configuração 2 apresenta apenas um nó abaixo do limiar. Com isso é possível verificar como ocorre o funcionamento em um ambiente diverso.

	Worker Node 1 (delay)	**Worker Node 2 (delay)**	**Worker Node 3 (delay)**
Configuração 1	2ms	43ms	136ms
Configuração 2	14ms	135ms	129ms
Configuração 3	99ms	72ms	69ms
Configuração 4	123ms	136ms	106ms
Configuração 5	134ms	9ms	33ms

Tabela 2 - Configurações de *delay* da experimentação

Para cada uma dessas configurações foram feitos 5 experimentos utilizando o algoritmo de migração proposto na Subseção 4.2, e 5 experimentos utilizando o escalonamento padrão do *Kubernetes*. O objetivo dessa avaliação é verificar o quanto a presença do processo de migração de serviço afeta a QoE dos usuários.

Cada um desses experimentos utilizaram 8 clientes acessando a aplicação simultaneamente, sendo gerados 8 instâncias por cliente. Todos os experimentos foram realizados da seguinte forma: durante o primeiro minuto de inicialização do *deployment*, os *worker nodes* rodam sem nenhuma alteração no *delay*. Após o primeiro minuto, os valores de *delay* explicitados na Tabela 2 são aplicados para cada *worker node*. Após aplicar essas configurações, os experimentos são executados durante mais 5 minutos, resultando em um tempo de experimentação total de 6 minutos.

Para cada experimento o *deployment* foi reiniciado, para evitar viés de um experimento para o outro.

5.2 Resultados

A Tabela 3 apresenta uma análise de correlação de *Spearman* para determinar a correlação entre a QoE=S (delay, jitter, loss) e a QoE calculada utilizando a equação 1. A partir dessa análise foi possível avaliar que o *delay* possui forte correlação negativa com a QoE, sendo inversamente proporcional. O que significa que ao diminuir o *delay* temos uma melhoria de QoE, e vice-versa.

	QoE	**delay**	**jitter**	**loss**
QoE	1.000000	-0.950447	-0.124893	-0.130485
delay	-0.950447	1.000000	-0.001238	-0.016962
jitter	-0.124893	-0.001238	1.000000	-0.001658
loss	-0.130485	-0.016962	-0.001658	1.000000

Tabela 3 - Correlação de *Spearman* entre QoS e QoE

Dado essa alta correlação do *delay* com a QoE, a avaliação considerou-se somente o *delay* como parâmetro no algoritmo proposto. A Tabela 4 apresenta o QoE médio considerando o *Kubernetes* (sem migração) e o algoritmo de migração proposto neste trabalho. A partir dos dados é possível ver que, no geral, o algoritmo de migração proposto melhorou a QoE dos usuários.

	Kubernetes (sem migração) (QoE Médio)	**Algoritmo de migração (QoE Médio)**	**Ganho (%)**
Configuração 1	7,3394 × 10^5 +- 1,934 x 10^3	9,5365 × 10^5 +- 1,592 x 10^3	29,94%
Configuração 2	6,2707 × 10^5 +- 2,415 x 10^3	1,0744 × 10^6 +- 1,903 x 10^3	71,34%
Configuração 3	6,2046 × 10^5 +- 2,186 x 10^3	6,2631 × 10^5 +- 2,394 x 10^3	0,94%
Configuração 4	5,5403 × 10^5 +- 3,029 x 10^3	5,6413 × 10^5 +- 3,020 x 10^3	1,82%
Configuração 5	7,6201 × 10^5 +- 1,876 x 10^3	9,0436 × 10^5 +- 1,452 x 10^3	18,68%

Tabela 4 - QoE Médio

Contudo, a melhora foi maior nas configurações 1, 2 e 5. Isso pode ser explicado devido ao fato do valor de *delay* de nenhum dos 3 *worker nodes* serem menores que o valor limiar de 50ms nas configurações 3 e 4 (Tabela 2). Isso significa que o algoritmo não realizou nenhuma migração de serviço. Esse problema pode ser resolvido definindo outros valores de limite para o algoritmo. Contudo, é imprescindível realizar uma análise para determinar se o *trade-off* entre o custo de migração e a melhoria significativa na Qualidade da Experiência (QoE) é justificável.

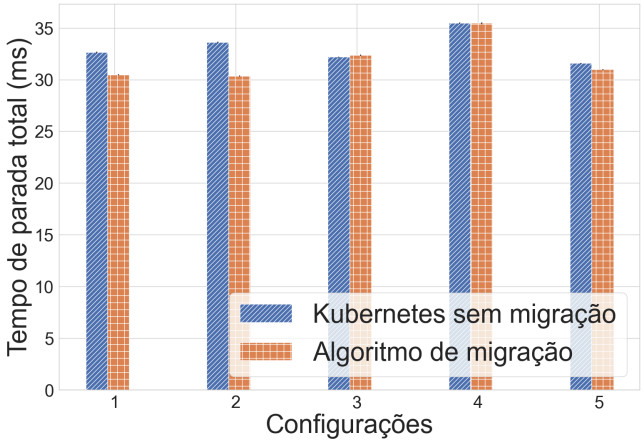

Figura 7 - Comparação do tempo de parada total entre *kubernetes* (sem migração) e algoritmo de migração

Na Figura 7 é possível concluir que o tempo de parada total foi otimizado para as duas primeiras configurações pelo algoritmo de migração, porém esse aumento não foi significativo de forma geral. Portanto, é possível concluir que a solução não foi eficaz na otimização deste parâmetro.

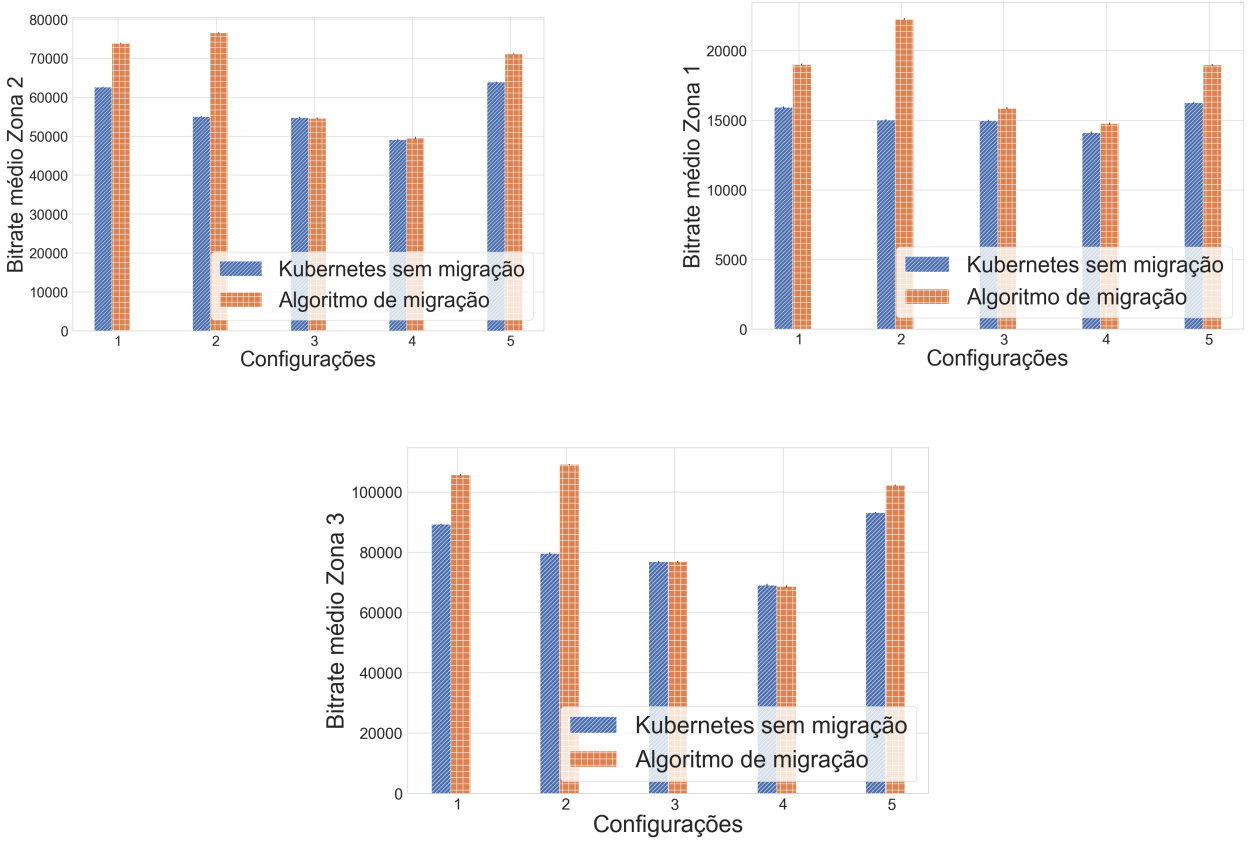

Figura 8 - Comparação do *bitrate* médio entre *kubernetes* (sem migração) e algoritmo de migração nas três zonas

Na Figura 8 é possível analisar a otimização do *bitrate* para todas as três zonas apresentadas. Em todas as configurações ocorreu o aumento desse parâmetro, para todas as zonas. Dessa forma é possível concluir que o algoritmo de migração foi eficaz na otimização do *bitrate*. As configurações 3 e 4 não apresentaram um aumento significativo porque são as duas configurações onde não houveram *delays* abaixo do *delay* limite de 50ms, nos *worker nodes*.

Em resumo, como apresentado na análise de dados, o algoritmo de migração proposto foi eficaz na melhoria de QoE em relação ao escalonamento padrão do *Kubernetes* (sem considerar a migração de serviço). Isso mostra que a migração de serviços é fator importante na melhoria da experiência do usuário, tomando decisão baseada na QoS apresentada no momento em que o usuário está consumindo o vídeo.

6. CONCLUSÃO E TRABALHOS FUTUROS

Esse trabalho propõe a melhoria de QoE utilizando a migração de serviços. Essa migração é feita a partir do monitoramento de métricas de QoS da rede, fazendo com que possa ser tomada a melhor decisão possível em tempo real.

Foi desenvolvido um ambiente experimental para a realização dos testes da arquitetura de sistema proposta. Dados foram coletados com o objetivo de correlacionar parâmetros de QoS com QoE, utilizando a correlação de *Spearman*. A partir disso, foi proposto o algoritmo de decisão de migração e seu funcionamento foi comparado com um ambiente sem a decisão de migração. Como evidenciado na Subseção 5.2, o algoritmo foi eficaz na melhoria da QoE, melhorando a QoE em até 29.94%

Para trabalhos futuros é importante que o algoritmo aceite diferentes gamas de limiar de latência para tomada de decisão de migração dentro do ambiente disponível. Dessa forma é possível fazer uma melhor análise do limite aceitável para a migração. É interessante que esse algoritmo utilize também outras métricas de QoS para a decisão de migração, como o *jitter* e o *loss*. É importante também que seja definida uma métrica

que responda se a migração de um contêiner é justificável, levando em conta o custo da operação de migração, e o ganho de QoE.

REFERÊNCIAS BIBLIOGRÁFICAS

YOUNIS, Ayman; TRAN, Tuyen X.; POMPILI, Dario. On-demand video-streaming quality of experience maximization in mobile edge computing. In: **2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)**. IEEE, 2019. p. 1-9.

RAHMAN, Waqas Ur; HONG, Choong Seon; HUH, Eui-Nam. Edge computing assisted joint quality adaptation for mobile video streaming. **IEEE Access**, v. 7, p. 129082-129094, 2019.

OUYANG, Tao et al. Adaptive user-managed service placement for mobile edge computing: An online learning approach. In: **IEEE INFOCOM 2019-IEEE conference on computer communications**. IEEE, 2019. p. 1468-1476.

KIM, Taewoon et al. Optimal container migration for mobile edge computing: Algorithm, system design and implementation. **IEEE Access**, v. 9, p. 158074-158090, 2021.

ISLAM, Md Tariqul; ROTHENBERG, Christian Esteve; GOMES, Pedro Henrique. Predicting XR services QoE with ML: Insights from in-band encrypted QoS features in 360-VR. In: **2023 IEEE 9th International Conference on Network Softwarization (NetSoft)**. IEEE, 2023. p. 80-88.

XU, Xiaodong; LIU, Jiaxiang; TAO, Xiaofeng. Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation. **IEEE Access**, v. 5, p. 16406-16415, 2017.

David, Davidson & Alves, Carlos & Nunes, Rafael & Oliveira, Roque. (2022). Benefícios e Riscos do Uso da Computação em Nuvem no Setor Público: uma análise baseada em artigos disponibilizados em bases dados acadêmicas de 2017 a 2021. **RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao**. E49. 537-549

SHI, Weisong et al. Edge computing: Vision and challenges. **IEEE internet of things journal**, v. 3, n. 5, p. 637-646, 2016.

KIM, Hyun Jong; CHOI, Seong Gon. A study on a QoS/QoE correlation model for QoE evaluation on IPTV service. In: 2010 **The 12th International Conference on Advanced Communication Technology (ICACT)**. IEEE, 2010. p. 1377-1382.

LIU, Shaoshan et al. Edge computing for autonomous driving: Opportunities and challenges. **Proceedings of the IEEE**, v. 107, n. 8, p. 1697-1716, 2019.

MACHIDA, Fumio et al. Optimizing resiliency of distributed video surveillance system for safer city. In: **2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)**. IEEE, 2015. p. 17-20.

ZHAO, Dan et al. A service migration strategy based on multiple attribute decision in mobile edge computing. In: **2017 IEEE 17th international conference on communication technology (ICCT)**. IEEE, 2017. p. 986-990.

KHAN, Muhammad Asif et al. A survey on mobile edge computing for video streaming: Opportunities and challenges. **IEEE Access**, v. 10, p. 120514-120550, 2022.

SINGH, J. N.; SRIVASTAVA, Nikhil; CHOUDHARY, Parul. Cloud Computing: Issues and Challenges. In: **2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)**. IEEE, 2023. p. 1080-1084.

JYOTI, Amrita; SHRIMALI, Manish; MISHRA, Rashmi. Cloud computing and load balancing in cloud computing-survey. In: **2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)**. IEEE, 2019. p. 51-55.

ODUN-AYO, Isaac; AJAYI, Olasupo; OKEREKE, Chinonso. Virtualization in cloud computing: Developments and trends. In: **2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)**. IEEE, 2017. p. 24-28.

ABBAS, Nasir et al. Mobile edge computing: A survey. **IEEE Internet of Things Journal**, v. 5, n. 1, p. 450-465, 2017.

MUNMUN, Farha Akhter; ANIS, Adeeba; HOSSAIN, Md Shohrab. A Comprehensive Comparison between Cloud Computing and Mobile Edge Computing. **International Journal of Research and Innovation in Applied Science (IJRIAS)**, v. 7, n. 7, p. 66-71, 2022.

WANG, Shangguang et al. A survey on service migration in mobile edge computing. **IEEE Access**, v. 6, p. 23511-23528, 2018.

HAIBEH, Lina A.; YAGOUB, Mustapha CE; JARRAY, Abdallah. A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches. **IEEE Access**, v. 10, p. 27591-27610, 2022.

SPINELLI, Francesco; MANCUSO, Vincenzo. Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility. **IEEE Communications Surveys & Tutorials**, v. 23, n. 1, p. 596-630, 2020.

KAUR, Kiranpreet; GUILLEMIN, Fabrice; SAILHAN, Francoise. Container placement and migration strategies for cloud, fog, and edge data centers: A survey. **International Journal of Network Management**, v. 32, n. 6, p. e2212, 2022.

HAIBEH, Lina A.; YAGOUB, Mustapha CE; JARRAY, Abdallah. A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches. **IEEE Access**, v. 10, p. 27591-27610, 2022.

XIE, Zhigang et al. Energy efficiency task scheduling for battery level-aware mobile edge computing in heterogeneous networks. **ETRI Journal**, v. 44, n. 5, p. 746-758, 2022.

CHEN, Luyao et al. Multi-mec collaboration for vr video transmission: Architecture and cache algorithm design. **Computer Networks**, v. 234, p. 109864, 2023.

RUI, LanLan et al. Service migration in multi-access edge computing: A joint state adaptation and reinforcement learning mechanism. **Journal of Network and Computer Applications**, v. 183, p. 103058, 2021.

O que é computação em nuvem? Um guia para iniciantes. **Azure Microsoft**, 2023. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-cloud-computing>. Acesso em 10 de set. de 2023.

O que é a computação em nuvem?. **Amazon AWS**, 2023. Disponível em: <https://aws.amazon.com/pt/what-is-cloud-computing>. Acesso em 10 de set. de 2023.

PADMANABHAN, Arvind. Multi-Access Edge Computing. **Devopedia**, 2022. Disponível em: <https://devopedia.org/multi-access-edge-computing>. Acesso em 10 de set. de 2023.

WHAT is multi-access edge computing (MEC)?. **RedHat**, 2022. Disponível em: <https://www.redhat.com/en/topics/edge-computing/what-is-multi-access-edge-computing>. Acesso em 10 de set. de 2023.

BASUMALLICK, Chiradeep. Top 10 Edge Computing Platforms in 2022. **Spiceworks**, 2022. Disponível em: <https://www.spiceworks.com/tech/edge-computing/articles/best-edge-computing-platforms/>. Acesso em 10 de set. de 2023.

ADIB, Dalia. What is mobile edge computing?. **STL Partners**, 2017. Disponível em: <https://stlpartners.com/articles/edge-computing/mobile-edge-computing/>. Acesso em 10 de set. de 2023.