Agendamento de Contêineres Ciente da QoE

Marcos Carvalho[1] **Daniel F. Macedo**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

{marcoscarvalho,damacedo}@dcc.ufmg.br

Abstract. *The cloud provider shares its computing resources among different clients, co-locating the applications on the same server. However, this can cause application degradation. In addition, cloud providers use Quality of Service (QoS) metrics as a way to measure the quality of service delivered to their customers. These metrics are pre-established and specified in the Service Level Objective (SLO). However, the SLO based on QoS is insufficient to guarantee the users of the applications a good Quality of Experience (QoE). The dissertation addresses this problem by proposing a QoE-aware container scheduler in a cloud environment where applications suffer interference caused by co-location. We propose a new approach that uses machine learning methods to estimate the QoE that the cloud can offer, considering cloud attributes. Experiments have shown that QoE-aware scheduling can improve users' QoE as well as reduce resource usage.*

Resumo. *O provedor de nuvem compartilha seus recursos computacionais entre diferentes clientes, co-localizando as aplicações no mesmo servidor. No entanto, isso pode causar degradação nas aplicações. Além disso, os provedores de nuvem utilizam métricas de Qualidade de Serviço (QoS) como forma de medir a qualidade do serviço entregue aos seus clientes. Essas métricas são pré-estabelecidas e especificadas no Service Level Objective (SLO). Contudo, o SLO baseado em QoS é insuficiente para garantir aos usuários das aplicações uma boa Qualidade de Experiência (QoE). A dissertação lida com esse problema, propondo um agendador de contêiner ciente da QoE em um ambiente de nuvem onde as aplicações sofrem interferência causada pela co-localização. Propomos uma nova abordagem que utiliza métodos de aprendizado de máquina para estimar a QoE que a nuvem pode oferecer, considerando atributos da nuvem. Experimentos mostraram que o agendamento com reconhecimento da QoE pode melhorar a QoE dos usuários, assim como reduzir o uso de recursos.*

1. Introdução

A computação em nuvem tem sido bem-sucedida em fornecer recursos computacionais para implantação de aplicações altamente disponíveis. De forma geral, o ambiente de nuvem é composto por dois agentes. O primeiro são os provedores de nuvem tais como a AWS e Google Cloud. O segundo, são os clientes da nuvem, os quais utilizam recursos computacionais/rede da nuvem para prover suas aplicações aos usuários finais.

O provedor de nuvem utiliza um modelo conhecido como Infraestrutura como serviço (do inglês, *Infrastructure As a Service* - IaaS). Esse modelo tem como característica o compartilhamento de recursos, o que é feito através de tecnologias de virtualização,

como máquinas virtuais e, recentemente, o uso de contêineres. Outro aspecto importante da nuvem é o escalonamento horizontal. No caso de aplicações baseadas em contêineres, a IaaS provê novas réplicas em tempo real para lidar com a variação da carga de trabalho. Neste processo, o agendamento de contêiner é um fator primordial, na qual consiste em escolher em qual servidor cada novo contêiner será alocado. Esse processo coloca as aplicações de forma co-localizadas no mesmo servidor, o que pode melhorar a utilização dos recursos. No entanto, essa prática pode gerar interferência entre as aplicações e, consequentemente, prejudicar o serviço prestado.

Para mitigar isso, os provedores da nuvem utilizam métricas de Qualidade de Serviço (QoS), tais como tempo de resposta, vazão e largura de banda. Essas métricas são, portanto, estabelecidas e acordadas entre as partes no *Service Level Objectives (SLO)*. Contudo, SLO baseado em QoS não é suficiente para aquelas aplicações onde a Qualidade de Experiência dos Usuários (QoE) é essencial e fundamental para os lucros dos provedores de conteúdo [Haouari et al. 2019]. Isso porque as métricas de QoS não refletem bem a experiência dos usuários finais [Juluri et al. 2015]. Ao contrário, a medição direta de QoE tornou-se um método mais robusto para entender a experiência e o engajamento do usuário final [De Cicco et al. 2019]. O método mais comum para representar a experiência do usuário é o *Mean Opinion Score (MOS)*.

No contexto da dissertação, o ambiente utilizado foi formalmente definido como se segue. $C = \{c_1, c_2, c_3 \dots c_k\}$ representa uma fila (FIFO) de novos contêineres a serem alocados. Cada novo contêiner será agendado para um conjunto de servidores $W = \{w_1, w_2, w_3 \dots w_m\}$ (processo de agendamento). Além disso, executamos o processo de reagendamento. Isso significa que, contêineres já alocados podem ser realocado para outro servidor devido a falhas de sistema ou hardware. Isso implica que o contêiner é excluído do servidor original e um novo contêiner é enfileirado para ser agendado para um novo servidor.

As conexões dos usuários são distribuídas entre esses contêineres por meio de um serviço de balanceamento de carga. Isso cria uma quantidade diferente de usuários por contêiner, representado como $U = \{u_1, u_2, u_3 \dots u_n\}$, onde U é o conjunto de todos os usuários para todos os contêineres e u_x representa o número de usuários em cada contêiner. Além disso, os usuários de cada contêiner experimentam uma QoE diferente, expressa como $Q_x = \{q_{x,1}, q_{x,2}, q_{x,3} \dots q_{x,n}\}$, onde $q_{x,i}$ é a QoE do usuário i para o contêiner x.

O trabalho proposto resolve dois problemas no cenário acima descrito. (i) *Qual é o melhor servidor W_i para alocar um novo contêiner visando maximizar a QoE e garantir o SLO acordado?* Além disso, (ii) *Dado um contêiner c_x já alocado no qual os usuários experimentam um QoE degradado $q_{x,n}$, como esse QoE pode ser melhorado para que não haja violação de SLO?*

Existem diversos algoritmos e técnicas para realizar o agendamento/reagendamento. No entanto, escolhemos utilizar técnicas de aprendizado de máquina (AM) para mapear a relação entre o uso de recursos da nuvem e a QoE dos usuários. Nesse caso, cada modelo de AM serve como um oráculo que responde a seguinte pergunta: *Para um novo contêiner, qual será a QoE dos usuários se o contêiner for implantado em um servidor específico?*

Dada a problematização acima citada, a dissertação propõe um agendando de contêiner com reconhecimento da QoE para aplicações co-localizadas na nuvem. Para alcançar esse objetivo geral, dividimos-o em outros objetivos específicos, listados abaixo.

- Propor uma arquitetura para agendamento de contêiner usando estimadores de QoE.
- Avaliar e desenvolver modelos de aprendizado de máquina para estimar a QoE do usuário para diferentes aplicações. Até onde sabemos, este é o primeiro trabalho que usa um conjunto de recursos de computação em nuvem (CPU, memória, disco, rede) para prever QoE dentro da nuvem, ou seja, a QoE que a nuvem pode oferecer.
- Desenvolver algoritmos para melhorar a QoE dos usuários escolhendo o melhor servidor para o alocação do conteiner.

O restante do texto está organizado como se segue. Apresentamos a motivação para o desenvolvimento da dissertação na seção 2. Em seguida, apresentamos alguns trabalhos relacionados na Seção 3. Já na Seção 4, apresentamos, de forma geral, os principais componentes da arquitetura proposta no trabalho. Apresentamos a avaliação e os resultados experimentais na Seção 5. Por fim, concluímos mostrando as principais contribuições do trabalho na Seção 6, o que inclui um artigo publicado na conferência IFIP/IEEE e uma submissão para o periódico *IEEE Transactions on Network and Service Management*.

2. Motivação

Diversos provedores de aplicações onlines estão migrando suas aplicações para o ambiente em nuvem. Isso graças às vantagens que a nuvem oferece, como redução de custo e a garantia de recursos computacionais/rede disponíveis [Abdallah et al. 2018].

Além da problematização entre a relação de QoE e QoE, os trabalhos apresentados na literatura que se propõem a melhorar a QoE não consideram a degradação causada pelas aplicações co-localizados na nuvem. Em geral, esses trabalhos medem a QoE e atuam em outra parte do caminho fim-a-fim dos usuários, como em redes Wi-Fi.

Portanto, a motivação da dissertação baseia-se na falta de soluções para agendamento de contêiner ciente da QoE dos usuários. Além disso, o gerenciamento automático para melhorar a QoE pode ser de interesse dos provedores de nuvem. Isso porque uma boa QoE pode melhorar a retenção dos clientes, o que aumentaria a margem dos lucros, além de ser um diferencial ao oferecer seus serviços.

3. Trabalhos Relacionados

Nossa metodologia para encontrar os trabalhos relacionados está disponível na dissertação. Em resumo, utilizamos a ferramenta Google Scholar e aplicamos algumas palavras-chaves na busca, como *"QoE-Aware cloud resource management"*,*"Container scheduling and "cloud"* e *"QoS-aware/drive"and container scheduling*.

Por fim, os resultados consideram aquelas propostas que utilizam contêineres, porque: (1) nosso trabalho aborda especificamente problemas de agendamento de conteineres. (2) Consideramos a crescente aplicação de conteineres em ambiente da nuvem, bem como um tópico de pesquisa emergente [Masdari and Khoshnevis 2020]. Abaixo apresentamos alguns dos trabalhos relacionados ao nosso.

Os autores [Liu et al. 2018] propõem um algoritmo de escalonamento de contêiner multiobjetivo considerando recursos do lado do servidor, sendo (1) a CPU do servidor, (2) o uso de memória e (3) o tempo necessário para transmitir a imagem do contêiner pela rede da nuvem. Além disso, (4) a proposta do autor também precisa classificar a demanda de recursos da aplicação. Com isso, o agendador leva em consideração quais recursos a aplicação demanda e os servidores disponíveis. O último fator é (5) o agrupamento de conteineres, o que significa que os autores consideram as características da aplicação para reduzir o consumo de transmissão da rede entre conteineres inter-relacionados. No entanto, os autores ignoraram o efeito da co-localização das aplicações, que pode sobrecarregar o servidor. Além disso, a proposta não utiliza nenhum mecanismo para mitigar sobrecargas, por exemplo, reescalonamento de conteineres. Por outro lado, nosso trabalho reagenda os containers quando a QoE do usuário é degradada.

Os autores [Mao et al. 2017] propõem um agendamento consciente de recursos computacionais chamado DRAPS (Dynamic and Resource-Aware Placement Scheme). O DRAPS executa o agendamento com base nos recursos dos servidores disponíveis no momento e identifica o tipo de recurso dominante usado por cada conjunto de conteineres. O agendamento equilibra o uso de recursos entre os servidores, evitando que os conteineres que usam o mesmo recurso de forma intensiva estejam no mesmo servidor. Além disso, a proposta dos autores realiza a migração de conteineres com base nos gargalos de recursos dos servidores. Essa migração pode evitar a degradação da QoE do usuário após a migração de conteineres. No entanto, os autores não especificaram o valor limite que determinaria um gargalo, o que pode ser difícil de decidir para cada tipo de aplicação. Por outro lado, nossa proposta usa aprendizado de máquina para determinar quando ocorre a degradação da QoE. Além disso, em vez de reprogramar aqueles containers com intensivo uso de recursos, decidimos priorizar os conteineres em que a QoE pode ser medida enquanto outras aplicações/serviços são executados em *background* com o melhor esforço.

Uma estratégia de agendamento de contêiner no Kubernetes chamada KCSS foi proposta em [Menouer 2021]. O KCSS pode usar um conjunto de critérios para selecionar um servidor com base no algoritmo Ordem de prioridade para semelhança com a solução ideal (TOPSIS)[Lai et al. 1994]. O objetivo é compactar os conteineres para usar o máximo de recursos possível do servidor. Portanto, o TOPSIS escolhe o servidor cuja distância das melhores e piores soluções é mínima com base na distância euclidiana n-dimensional. Isso significa que o TOPSIS agrega todos os critérios em uma única classificação e seleciona o servidor com a classificação mais alta. No entanto, os autores não consideram a interferência causada pelas aplicações co-localizados. Apesar disso, o KCSS é extensível, permitindo a utilização e combinação de novos critérios e recursos. Devido a essa generalização, usamos esse algoritmo como baseline.

Os trabalhos aqui apresentados, assim como os demais detalhados na dissertação mostram que nenhum dos trabalhos visam melhorar a QoE dos usuários. Em vez disso, o objetivo final é maximizar a utilização de recursos da nuvem ou alguma métrica de QoS, como largura de banda da rede. Além disso, diferentemente de outros trabalhos encontrados na literatura, nossa proposta estima a QoE considerando apenas o estado da nuvem, o que significa que estamos preocupados em como a nuvem pode melhorar a QoE dos usuários de ponta-a-ponta.

4. Agendador de contêineres Ciente da QoE

A arquitetura de sistema proposta na dissertação é dividida em dois planos. Primeiro, o Plano de Dados é formado pela plataforma de computação em nuvem (Kubernetes Engine). Segundo, o Plano de Controle é composto por três módulos que serão descritos abaixo. Os dois últimos módulos são as principais contribuições do trabalho (destacados em preto).

Monitor de Recursos de Nuvem: Este módulo utiliza ferramentas existentes para coletar métricas da nuvem. No trabalho, utilizou-se o *cAdivsor* devido à facilidade de uso e ampla implantação no Kubernetes. As métricas coletadas são utilizadas como **atributos de entradas** para os modelos. Separamos as métricas de uso em 5 categorias, sendo: CPU, Memória, Disco, Sistema de arquivos e Redes. O detalhamento das métricas estão disponíveis na dissertação.

Monitor de QoE Baseado em AM: Este módulo armazena e opera os preditores baseados em AM que fornecem estimativas em tempo real da QoE oferecida para cada aplicação. Cada modelo de cada aplicação leva em consideração o uso de recursos dos nós de trabalho e contêineres.

A saída do modelo é o QoE dos usuários, na qual foi modelada considerando a Recomendação ITU-T P.1203 [ITU Telecommunication Standardization Sector 2017]. Neste caso, a QoE é estabelecida através do MOS, que é determinado entre 0 e 5, sendo 0 uma QoE ruim e 5 extremamente bom. De forma geral, consideramos metadados do vídeo (resolução,taxa de bits) para determinar a QoE dos usuários da aplicação de vídeo sob demanda. Por outro lado, para a aplicação de sala de aula ao vivo consideramos também eventos de travamento do vídeo. Na dissertação detalhamos formalmente as entradas gerais dos modelos propostos, que podem ser usadas para qualquer estimador. Além disso, apresentamos também as saídas dos modelos para as duas aplicações utilizadas no trabalho.

O Módulo de Decisão do Agendador (SD): O módulo SD emprega uma abordagem gulosa para otimizar a QoE no agendamento e reagendamento. Ambos os procedimentos (agendamento/reagendamento) usam os preditores como um oráculo para estimar a QoE dos usuários do contêiner, se implementado em um determinado nó de trabalho.

5. Avaliação Experimental

Nossa avaliação experimental foi realizada no testbed Virtual Wall [Municio et al. 2021]. Utilizamos cinco máquinas físicas para a instalação do Kubernetes, sendo uma para o nó mestre e quatro para os nós de trabalhos. Uma outra máquina física foi utilizada para implementar nossa solução, o que inclui o módulo para coleta dos dados da nuvem, o algoritmo proposto e os estimadores de QoE. Alocamos também 14 máquinas virtuais (VMs) para serem os clientes, sendo 7 para cada aplicação.

Para simular diferentes aplicações co-localizados, configuramos uma carga de trabalho diferente em cada nó do trabalhador, utilizando a ferramenta *stress-ng* e *Iperf3*. Além disso, o algoritmo proposto na dissertação realiza o escalonamento de contêiner considerando um valor limite que é definido no SLO e um intervalo de tempo T. O SLO é definido para que a $QoE \geq 3$ e o intervalo para 10 segundos. Portanto, consideramos a degradação da QoE quando a QoE está a baixo do valor do SLO durante o intervalo T.

Como *baselines* consideramos o agendador padrão do Kubernetes e duas versões do KCSS. A primeira com o objetivo de maximizar o uso de recursos (KCSS Max) e a segunda com o objetivo de minimizar o uso de recursos (KCSS Min).

5.1. Resultados Experimentais

Nesta seção apresentamos os principais resultados da dissertação. Primeiro vamos discutir sobre os resultados do processo de treinamento dos modelos. Por fim, mostramos como a QoE foi melhorada utilizando nossa proposta em relação aos demais agendadores.

Aprendizagem de Máquina: Consideramos a medição da QoE como uma série temporal, assim implementamos algoritmos de aprendizado de máquina tais como: *Short-Term Long Memory* (LSTM) e *Gated Recurrent Unit* (GRU). O treinamento dos modelos (um para cada aplicação) segue a mesma metodologia. Separamos o conjunto de dados entre treinamento (70%), validação (20%) e teste (10%). Além disso, executamos várias técnicas que visam evitar o *overfitting*. Por exemplo, realizamos o ajuste de hiperparâmetros através da técnica *Grid Search*.

Tabela 1. RMSE obtido no GRU e LSTM

	LSTM		**GRU**	
	VoD	Live Classroom	VoD	Live Classroom
Validação (20%)	0.044	0.039	0.084	0.099
Teste (10%)	0.032	0.035	0.072	0.085

Usamos a Raiz do Erro Quadrático Médio (RMSE) para medir a qualidade da previsão. A Tabela 1 mostra o RMSE obtido pelo LSTM e GRU. Avaliamos o RMSE sobre os conjuntos de dados de validação e teste de cada aplicação. Como resultado, ambos os algoritmos alcançam uma qualidade de previsão satisfatória em ambas as aplicações. Embora o RMSE dos algoritmos tenham obtido um resultado satisfatório, o LSTM se destacou e, consequentemente, o escolhemos para ser utilizado no nosso módulo **Monitor de QoE Baseado em ML** para ambas as aplicações.

Por fim, realizamos uma avaliação experimental para verificar a capacidade dos modelos de generalizar em um ambiente diferente de onde foi treinado. Realizamos essa análise medindo o erro dos modelos em ambos os ambientes. Para isso, criamos dois novos conjuntos de dados (um para cada aplicação) com 10 sessões de vídeo. O RMSE nesses novos conjuntos de testes foi, em média, de 0.043 com um desvio padrão 0.015 para a aplicação vídeo sob demanda e de 0.033 com desvio padrão de 0.016 para a aplicação de sala de aula ao vídeo. Isso mostra que os modelos foram capazes de generalizar para um novo ambiente, dado que o RMSE ficou próximo daquele obtido na validação no ambiente original (Tabela 1). Isso pode ser explicado pelas técnicas empregadas durante o treinamento do modelo que evitaram o overfit.

Melhoria da QoE e Utilização de Recursos: Embora os *baselines* não executem o agendamento de contêiner ciente da QoE, medimos os efeitos desses agendamentos na QoE dos usuários e comparamos com nossa proposta. Consideramos diversos cenários na avaliação, na qual, em todas eles, a nossa proposta obteve uma melhora na QoE dos usuários. Por exemplo, em um dos cenários, nossa proposta obteve uma QoE de 4.1 para aplicação VoD, o que significa uma melhora de 95,2%, 86,3% e 78,2%, comparado com

2,1, 2,2 e 2,3 obtidos no Kubernetes-Scheduler, KCSS Max e KCSS Min, respectivamente. Da mesma forma para sala de aula ao vivo, nossa solução obteve uma QoE de 4.2, representando uma melhora de 61,5%, 82,6% e 75%, comparado com 2,6, 2,3 e 2,4 obtidos respectivamente no Kubernetes-Scheduler, KCSS Max e KCSS Min.

Além das melhorias na QoE, nossa solução diminuiu a quantidade de recursos utilizados. Por exemplo, em um dos experimentos, o agendador do Kubernetes atingiu o sobre-provisionamento de contêiner em 47%, enquanto a nossa solução obteve apenas 26.3%. Isso é devido à forma como cada agendador realiza a escolha do servidor. Em nossa abordagem, escolhemos aquele servidor com melhores condições o que, além de influenciar a QoE, provou-se ser impactante também na quantidade de recursos utilizados.

6. Contribuições

Desenvolvemos estimadores de QoE utilizando métodos de aprendizado de máquina profundo para realizar o agendamento e reagendamento de containers na nuvem. A proposta foi avaliada em um ambiente experimental considerando duas diferentes aplicações de vídeo: vídeo sob demanda e uma aplicação que simula uma sala de aula ao vivo (live streaming). Embora tenhamos utilizados aplicações de vídeo, a arquitetura proposta no trabalho é capaz de suportar outras aplicações, desde que haja um preditor de QoE para as aplicações, por exemplo, transmissão de áudio e sites.

As principais contribuições do trabalho são: 1) Um agendador e reagendador de contêineres que levam em consideração a QoE dos usuários para aplicações co-localizados; 2) Um sistema de agendamento/reagendamento de QoE que utilizam valores de QoE como métrica objetiva configurada no SLO e estende o Kubernetes Scheduler; 3) Dois novos estimadores de QoE que utiliza métodos de AM profundo para prever a QoE dos usuários em vídeo sob demanda e aplicação de sala de aula virtual ao vivo. Modelamos a QoE seguindo a Recomendação ITU-P P.1203; 4) A avaliação quantitativa da nossas propostas foi realizada em um ambiente experimental. 5) Uma publicação [Carvalho and Macedo 2021]. 6) Uma submissão para o periódico *IEEE Transactions on Network and Service Management (IEEE TNSM)*

Referências

Abdallah, M., Griwodz, C., Chen, K.-T., Simon, G., Wang, P.-C., and Hsu, C.-H. (2018). Delay-sensitive video computing in the cloud: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(3s):1–29.

Carvalho, M. and Macedo, D. F. (2021). Qoe-aware container scheduler for co-located cloud environments. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 286–294.

De Cicco, L., Mascolo, S., and Palmisano, V. (2019). QoE-driven resource allocation for massive video distribution. *Ad Hoc Networks*, 89:170–176.

Haouari, F., Baccour, E., Erbad, A., Mohamed, A., and Guizani, M. (2019). Qoe-aware resource allocation for crowdsourced live streaming: A machine learning approach. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

ITU Telecommunication Standardization Sector (2017). ITU-T Rec P.1203: Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport.

Juluri, P., Tamarapalli, V., and Medhi, D. (2015). Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):401–418.

Lai, Y.-J., Liu, T.-Y., and Hwang, C.-L. (1994). Topsis for modm. *European journal of operational research*, 76(3):486–500.

Liu, B., Li, P., Lin, W., Shu, N., Li, Y., and Chang, V. (2018). A new container scheduling algorithm based on multi-objective optimization. *Soft Computing*, 22(23):7741–7752.

Mao, Y., Oak, J., Pompili, A., Beer, D., Han, T., and Hu, P. (2017). Draps: Dynamic and resource-aware placement scheme for docker containers in a heterogeneous cluster. In *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE.

Masdari, M. and Khoshnevis, A. (2020). A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–2424.

Menouer, T. (2021). Kcss: Kubernetes container scheduling strategy. *The Journal of Supercomputing*, 77(5):4267–4293.

Municio, E., Cevik, M., Ruth, P., and Marquez-Barja, J. M. (2021). Experimenting in a global multi-domain testbed. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE.