Monetizando recursos de computação ociosos na borda para execução de serviços nativos da nuvem em regiões dinâmicas

Alex F R Trajano[1,2], **José Neuman de Souza**[2]

[1]Instituto Atlântico
Fortaleza – CE – Brazil

[2]Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil

alex_ferreira@atlantico.com.br, neuman@ufc.br

Abstract. *This article introduces a system that utilizes idle resources from Personal Computers (PCs) at the edge, aiming to execute cloud-native microservices. The system encourages PC owners to voluntarily share resources to profit through a resource pricing mechanism that considers electricity cost and PC performance. Simultaneously, the system allows cloud-native microservices developers to define their maximum valuation for PC resources, intending to execute such services at the edge. The system identifies real-time idle resources and strategically organizes them across multiple geographical regions, enabling the deployment of microservices closer to end-users, thereby reducing communication latency. Utilizing double auctions, the system efficiently matches PCs to microservices in their respective regions. Detailed measurements in comprehensive simulations substantiate the effectiveness of uServ, ensuring gains for PC owners and, simultaneously, a reduction in operational costs for microservices developers.*

Resumo. *Este artigo apresenta um sistema que utiliza recursos ociosos de Computadores Pessoais (PCs) na borda, visando executar microsserviços nativos de nuvem. O sistema incentiva os proprietários de PCs a compartilharem recursos voluntariamente a fim de obter lucro por meio de um mecanismo de precificação de recursos que considera o custo de eletricidade e o desempenho do PC. Simultaneamente o sistema permite que desenvolvedores de microsserviços nativos da nuvem definam sua valoração máxima para os recursos de PCs, no intuito de executar tais serviços na borda. O sistema identifica recursos ociosos em tempo real e os organiza estrategicamente em várias regiões geográficas, possibilitando a implantação de microsserviços mais próximo dos usuários finais, reduzindo a latência de comunicação. Utilizando leilões duplos, o sistema combina eficientemente PCs a microsserviços em suas respectivas regiões. Medições detalhadas em simulações abrangentes substanciam a eficácia do uServ, assegurando ganhos para os proprietários de PCs e, simultaneamente, uma redução nos custos operacionais para os desenvolvedores de Microsserviços.*

1. Introdução

No âmbito da computação distribuída, que engloba paradigmas como Computação em Nuvem, Computação em Borda, Computação em Névoa, e outros modelos, reconhecemos

a diversidade de recursos computacionais, desde data centers centralizados até dispositivos de borda descentralizados. Essa perspectiva destaca a necessidade por uma integração coordenada desses recursos para possibilitar o processamento eficiente de dados em diferentes pontos da rede, atendendo às demandas de ambientes computacionais distribuídos.

Apesar da atenção dada à Computação de Borda e à Computação em Névoa, a maioria dos usuários continua a acessar aplicativos executados principalmente na Nuvem. Isso se deve à preferência dos desenvolvedores por acordos de nível de serviço (SLAs) robustos oferecidos por provedores de nuvem, que garantem serviços estáveis e escaláveis. Com a adoção crescente da abordagem *Cloud-Native* no desenvolvimento de software, a arquitetura de microsserviços tornou-se dominante, juntamente com a rápida adoção de contêineres como tecnologia de virtualização [Mavridis and Karatza 2023]. Ao optar por provedores de nuvem, os desenvolvedores assumem a responsabilidade financeira, pois os custos estão diretamente relacionados à utilização de recursos em nuvem.

Por outro lado, existe uma quantidade considerável de Computadores Pessoais (PCs) globalmente dispersos que estão ociosos por uma parcela de tempo considerável, visto que tais dispositivos costumam ser superdimensionados [Nedevschi et al. 2009], apresentando uma oportunidade para desenvolver mecanismos que aproveitem e monetizem esses recursos abundantes, o que poderia trazer alternativas além dos recursos de nuvem tradicional para os desenvolvedores. Com a transição para um paradigma que integra Computação em Nuvem, em Borda e em Névoa, o apelo dos recursos ociosos nos PCs aumenta. Apesar de muitos trabalhos explorarem a utilização de recursos ociosos provenientes de veículos [Hamdi et al. 2022], servidores de borda e névoa dedicados [Luo et al. 2021, Tabatabaee Malazi et al. 2022], e dispositivos IoT genéricos [Gasmi et al. 2022], uma lacuna de pesquisa notável persiste na investigação da execução direta de microsserviços nativos de nuvem em PCs distribuídos ao longo da borda da rede. Além do mais, uma parcela considerável desses trabalhos dão ênfase a estratégias de *offloading* de tarefas entre dispositivos IoT, borda, névoa e nuvem, o que é um problema diferente da execução de microsserviços. Notavelmente, os microsserviços costumam ser aplicações que expõem interfaces de comunicação externas, atuando como servidores na arquitetura cliente/servidor.

Considerando os desafios e lacunas de pesquisa mencionados, este trabalho visa apresentar o uServ, um sistema que aproveita recursos ociosos de PCs para a execução de microsserviços nativos de nuvem. O uServ identifica e prevê o uso de recursos em tempo real nos PCs, disponibilizando unidades de recurso para a execução de microsserviços de terceiros. Para incentivar os proprietários de PCs a compartilharem recursos ociosos, o uServ determina o preço de uma unidade de recurso por meio de um leilão duplo entre provedores e consumidores de recursos. Além disso, o uServ gerencia implantações em regiões dinâmicas que são geograficamente distribuídas, buscando minimizar a latência de comunicação.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta alguns trabalhos relacionados a este. Na Seção 3 é apresentada a arquitetura proposta, assim como os algoritmos críticos para o uServ. Por sua vez, a Seção 4 descreve o conjunto de simulações executadas para avaliar a proposta deste artigo, e, finalmente a Seção 5 conclui o trabalho resumindo as principais descobertas e direções futuras de pesquisa.

2. Trabalhos relacionados

A seguir são apresentados alguns trabalhos relacionados que abordam o aproveitamento de recursos computacionais subutilizados. Estas pesquisas, embora valiosas, destacam lacunas específicas que orientam o presente trabalho.

O conceito de uma rede de serviços apoiada por todos os usuários da internet foi introduzido no trabalho [Hosseini et al. 2019], onde é apresentado uma visão na qual os usuários da internet colaboram para estabelecer o *CrowdCloud*, um sistema totalmente distribuído que agrega recursos computacionais de vários dispositivos, permitindo que cada usuário atue como consumidor e provedor de serviços. Os autores propõem um modelo no qual toda a infraestrutura é proveniente do público, adquirida por meio de *crowdsourcing*. A estrutura do CrowdCloud envolve a multidão como a infraestrutura fundamental, composta por usuários e dispositivos que contribuem com seu poder computacional, tendo todos os recursos gerenciados por uma plataforma específica. Embora o trabalho introduza um modelo baseado nos recursos dos usuários, uma lacuna de pesquisa significativa permanece, pois os autores apresentam um arcabouço conceitual, mas não realizaram qualquer forma de implementação ou conduziram uma exploração completa das tecnologias fundamentais para sua proposta.

O trabalho de [Ali et al. 2021] propõe um novo paradigma computacional chamado *Computação em Névoa Apoiada por Voluntários (VSFC)* para aplicações de IoT que combina Computação em Névoa e Computação Voluntária (VC) para aumentar as capacidades computacionais na borda da rede. Os autores estendem a ferramenta iFogSim incorporando uma camada de VC para viabilizar o VSFC e realizam extensas simulações para fornecer uma análise comparativa entre a Névoa conceitual e VSFC. O artigo destaca os benefícios da combinação de computação em névoa e computação voluntária para minimizar atrasos de comunicação, consumo de energia e uso de rede. O método proposto mostra-se eficaz na execução de trabalhos sensíveis a atrasos em dispositivos voluntários próximos disponíveis. Apesar dos benefícios, o artigo não aprofunda a discussão sobre incentivos para usuários que compartilham recursos ociosos, nem aborda os desafios relacionados à conectividade de ponta a ponta, gerenciamento de recursos e alocação de serviços dentro da proposta.

Conforme os autores em [Nguyen et al. 2020], a maioria dos veículos passa mais de 95% do tempo estacionado, apresentando uma oportunidade para utilizar eficientemente seus recursos computacionais. Assim, o trabalho introduz o *EdgePV*, um sistema construído na colaboração entre veículos estacionados e servidores de borda. O *EdgePV* é projetado para capacitar dispositivos IoT a consumirem serviços fornecidos por um *cluster* Kubernetes onde veículos atuam como nós trabalhadores, juntamente com um servidor de borda dedicado servindo como nó mestre. À medida que os veículos estacionam próximos a um nó mestre, eles se integram de maneira transparente ao *cluster* Kubernetes. Esse modelo está alinhado com a abordagem *Cloud Native*, fundamentada na conteinerização de microsserviços, frequentemente implantados dentro de *clusters* Kubernetes, conforme observado em [Toffetti et al. 2017]. Apesar do avanço na área introduzido pelo trabalho, ainda carecem evoluções acerca das recompensas oferecidas aos proprietários dos veículos. Além do mais, o trabalho propõe uma abordagem localizada, focando em melhorar a oferta de recursos apenas onde houverem *clusters* Kubernetes disponíveis.

3. Arquitetura e algoritmos do uServ

O uServ habilita os proprietários de PCs (provedores) a aproveitarem seus recursos computacionais subutilizados. Ao facilitar a execução de microsserviços de terceiros em seus PCs, os provedores podem obter compensação financeira por seus ativos computacionais ociosos. Vale ressaltar que os desenvolvedores de microsserviços (consumidores) já estão pagando por serviços em nuvem, portanto o uServ seria uma plataforma alternativa para implantar microsserviços diretamente na borda, aproximando-os de seus usuários finais. O sistema coleta pagamentos dos consumidores e os redistribui aos provedores que efetivamente hospedaram os microsserviços implantados. Esse modelo é projetado para incentivar os provedores a compartilharem consistentemente seus recursos, pois a compensação financeira está alinhada à duração e ao volume de recursos efetivamente utilizados pelos consumidores. Portanto, o uServ é estruturado por vários componentes que residem tanto na borda quanto na nuvem, cada um com tarefas específicas e complementares entre si. A Figura 1 apresenta uma visão geral da arquitetura e das relações entre os principais componentes.

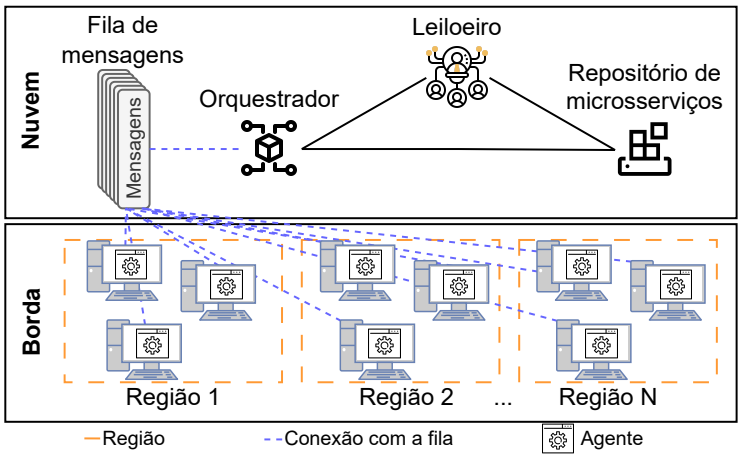


Figura 1. Visão geral da arquitetura do uServ

A arquitetura é dividida em duas camadas: a camada em nuvem e a camada em borda. Na nuvem, o Orquestrador é responsável por subdividir os PCs em regiões, além de gerenciar os recursos ociosos dos PCs e a alocação de microsserviços. Essa tarefa conta com suporte de um leiloeiro, responsável por conduzir leilões para determinar os preços dos recursos, além do repositório de microsserviços que mantém as imagens dos contêineres e seus parâmetros operacionais. Na camada de borda, cada PC opera um Agente, encarregado de coletar métricas de uso, gerenciar os contêineres dos microsserviços.

Os Agentes realizam o monitoramento em tempo real das métricas de uso dos PCs, possibilitando ao sistema identificar períodos de ociosidade. Ao concentrar-se exclusivamente em recursos ociosos, o uServ busca que seus usuários possam utilizar seus PCs sem afetar a sua produtividade. Dessa forma, os usuários podem monetizar seus recursos sem maiores esforços, tendendo a incentivar os provedores a participarem voluntariamente do sistema, aumentando a disponibilidade de recursos para a execução de microsserviços. Na perspectiva do consumidor, são fornecidas interfaces para registro de seus microsserviços para execução na plataforma, por meio do Repositório de microsserviços. Para isso, o

uServ requer imagens de contêineres e requisitos de hardware específicos para garantir uma alocação adequada de recursos para cada serviço. Essa prática está alinhada com os padrões da indústria observados em provedores de nuvem, onde os consumidores geralmente especificam os requisitos mínimos de recursos necessários para suas operações.

3.1. Estabelecimento de regiões

Todos os microsserviços implantados no uServ podem ser acessados globalmente pela Internet, se requisitado pelo consumidor. Isso apresenta um desafio para a alocação de microsserviços em PCs, visto que idealmente o serviço deve estar próximo de seus usuários finais a fim de minimizar latência. Por esta razão, o uServ divide os PCs em regiões, uma prática semelhante à dos provedores de nuvem. Dado a possível flutuação nos níveis de ociosidade e disponibilidade de PCs, o Orquestrador determina as regiões dinamicamente em intervalos de tempo predeterminados. A gestão de regiões no uServ envolve a formação de agrupamentos de PCs com base em sua proximidade geográfica.

A incorporação de *Affinity Propagation* no uServ desempenha um papel fundamental no estabelecimento de regiões. Affinity Propagation [Dueck 2009] é uma técnica que identifica *clusters* considerando similaridades e a disponibilidade de dados representativos. Dentro do uServ, esse método identifica grupos de PCs com base em sua proximidade geográfica, formando *clusters* que representam regiões específicas. Assim, quando um provedor disponibiliza seu PC no uServ, este deve registrar as coordenadas geográficas do seu dispositivo, servindo de entrada do algoritmo de agrupamento. O algoritmo funciona por meio da simulação iterativa da troca de mensagens entre PCs, identificando exemplares-chave ou representantes dentro do conjunto de dados. Inicialmente, cada PC considera todos os outros PCs como exemplares potenciais e troca mensagens para determinar os exemplares mais adequados. Essas mensagens são baseadas na similaridade entre as coordenadas, ajudando a avaliar quais PCs são mais adequados para servir como exemplares para os demais. Ao longo de iterações sucessivas, o algoritmo identifica PCs específicos como exemplares, resultando na formação de *clusters* distintos. Uma das vantagens desse método é sua capacidade de determinar automaticamente o número de *clusters* sem exigir um valor predefinido, o que é especialmente útil, uma vez que o número de *clusters* é desconhecido devido à dinâmica de uso da rede de PCs.

3.2. Gerenciamento de recursos ociosos

O Agente desempenha um papel crucial ao monitorar continuamente as estatísticas de uso do PC em tempo real. Ele tem a função de prever a disponibilidade (ou ociosidade) de recursos em um curto período de tempo, alguns segundos no futuro. Essas previsões são transmitidas para a fila de mensagens na nuvem a intervalos regulares, permitindo que o orquestrador determine o volume de recursos disponíveis. Para tanto, o orquestrador organiza os recursos em blocos que serão oferecidos aos consumidores para a execução de microsserviços.

A predição de utilização de recursos realizada pelo Agente é feita através da coleta minuto a minuto das seguintes métricas: percentual de uso e carga de 1 minuto da CPU, memória RAM em uso, vazão de entrada e saída de rede, e espaço de armazenamento em uso. Cada uma dessas métricas é coletada e tem seu valor futuro estimado através de um

método baseado em médias móveis e estimativa de tendência, dado por:

$$\hat{y}_{T+h|T} = \max\left\{\frac{(y_{T-w} + \ldots + y_T)}{w}, y_T + h\left(\frac{y_T - y_{T-d}}{d}\right)\right\} \quad (1)$$

onde y_T é o valor da métrica no tempo T, h é o número de passos no futuro a serem previstos, $\hat{y}_{T+h|T}$ é a previsão para o tempo $T + h$ dada a informação até o tempo T, w é o comprimento da janela da média móvel, e d é o comprimento da janela de tendência. A lógica por trás da Equação 1 é garantir uma abordagem cautelosa na previsão do uso de recursos, não antecipando excessivamente períodos de inatividade. Assim, se os recursos apresentarem uma queda de uso repentina, os valores previstos respondem lentamente. Inversamente, quando os recursos se tornam escassos, os valores previstos se ajustam rapidamente. Isso é feito calculando a média móvel das últimas w medições e a tendência das últimas d amostras. Quando $w > d$, a média móvel responde lentamente a valores extremos que surgem no tempo T, enquanto a tendência captura imediatamente mudanças abruptas nas medições.

Um desafio no contexto de PCs é que seus recursos são heterogêneos. Por esta razão, o uServ homogeneíza os recursos oferecendo blocos com quantidades predeterminadas de recursos, assim como fazem os provedores de nuvem tradicionais. O **Bloco de Recursos** (RB) é a unidade de alocação do uServ. Uma configuração n-RB equivale a n núcleos de CPU, n GB de RAM, n GB de armazenamento e n Mbps de largura de banda de rede. Essa padronização de RB permite que os consumidores solicitem uma configuração de RB alinhada com os requisitos específicos para a execução de seus microsserviços. Por exemplo, um 3-RB equivale a uma máquina de 3 núcleos de CPU, 3 GB de RAM, 3 GB de armazenamento e 3 Mbps de largura de banda.

No entanto, um n-RB não pode ser considerado completamente homogêneo, dadas as possíveis variações de hardware entre diferentes PCs. Nesse sentido, o agente uServ instalado num PC realiza um *benchmark* para avaliar o poder de processamento real de um 1-RB alocado nesse PC específico. Esse processo pode ser realizado por meio de ferramentas conhecidas de *benchmarks* sintéticos, como o *PassMark*[1] e o *PCMark10*[2]. Como os testes são padronizados e possuem pontuações lineares, cada PC recebe uma pontuação que é usada para comparar seu desempenho com outros. Com o ranqueamento das pontuações, o uServ remunera os PCs proporcionalmente ao seu desempenho.

A medida que as predições de recursos utilizados são enviadas para o Orquestrador, o sistema vai automaticamente identificando a quantidade de RBs disponíveis em cada PC. O Algoritmo 1 é aplicado para realizar esse processo. O algoritmo calcula o número de RBs para cada possível n-RB, iterando de 1 até o máximo permitido n, que é igual ao menor valor no conjunto Q. Em seguida, o número de n-RBs é calculado com base nos recursos disponíveis, capturando o número mínimo da divisão de cada recurso disponível por n. Os n-RBs calculados são adicionados ao conjunto de RBs. Finalmente, o algoritmo retorna o conjunto de blocos de recursos, onde cada entrada compreende uma tupla, sendo o primeiro valor representando n em n-RB, e o segundo valor indicando o número de n-RBs disponíveis.

[1]Disponível em https://www.passmark.com/

[2]Disponível em https://benchmarks.ul.com/pcmark10

Algoritmo 1 Algoritmo para definir os blocos de recursos disponíveis num PC

$Q_{cpu} \leftarrow CPU_{total} - \lceil \max(CPU_{load}, CPU_{total} \times CPU_{usage}) \rceil$
$Q_{memory} \leftarrow \lfloor Memory_{total} - Memory_{used} \rfloor$
$Q_{storage} \leftarrow \lfloor Storage_{total} - S_{used} \rfloor$
$Q_{network} \leftarrow Network_{total} - \lceil max(Network_{rx}, Network_{tx}) \rceil$
$Q \leftarrow (Q_{cpu}, Q_{memory}, Q_{storage}, Q_{network})$
$RBs \leftarrow \emptyset$
for $n \leftarrow 1$ **to** $min(Q)$ **do**
    $ra \leftarrow \left\lfloor \frac{q}{n} \right\rfloor, \forall q \in Q$
    $RB \leftarrow ((n, min(ra)))$
    $RBs.add(RB)$
end for
return RBs

3.3. Implantação de microsserviços em regiões

Devido à variação na quantidade de n-RBs nos PCs ao longo do tempo, influenciada pela utilização dos dispositivos, o uServ adota uma abordagem dinâmica para alocar essas unidades, considerando que a alocação de n-RBs tem uma duração limitada de apenas 1 minuto. A cada minuto, o sistema reavalia a quantidade disponível de n-RBs e, em seguida, usa o *Affinity Propagation* para definir dinamicamente as regiões com base na proximidade geográfica dos PCs. Após esse processo, o sistema determina em qual região alocar um microsserviço, buscando minimizar a distância entre a região selecionada e os usuários ativos do microsserviço.

O processo adotado pelo uServ consiste em alocar os microsserviços em regiões aleatórias, quando estes são registrados no sistema pelo consumidor. A partir desse posicionamento inicial, o sistema coleta os endereços IP de cada um dos consumidores dos microsserviços, permitindo inferir a localização geográfica dessas requisições, que podem ser tanto de usuários finais, como de outros microsserviços. A partir daí, o sistema resolve o problema de otimização linear a seguir. Seja N o número de regiões, $R = r_1, r_2, \ldots, r_N$ representando o conjunto de regiões, M o número de IPs que acessaram o microsserviço, $IP = ip_1, ip_2, \ldots, ip_M$ representando o conjunto de IPs, e a matriz de alocação A de tamanho $N \times M$ onde:

$$A_{n,m} = \begin{cases} 1 & \text{se } r_n \text{ deve atender } ip_m \ 0 \\ \text{caso contrário} \end{cases} \tag{2}$$

A função objetivo pode ser expressa como:

$$\text{Minimize: } \sum_{n=1}^{N} \sum_{m=1}^{M} d(r_n, ip_m) A_{n,m} \tag{3}$$

onde $d(r_n, ip_m)$ denota a distância euclidiana entre o centro da região r_n e as coordenadas aproximadas do endereço IP ip_m. A otimização está sujeita a:

$$\sum_{n=1}^{N} A_{n,m} = 1 \quad \text{para cada coluna } m, \text{ e } 1 < \sum_{n=1}^{N} \sum_{m=1}^{M} A_{n,m} \leq X \tag{4}$$

onde X é o número máximo de instâncias que o microsserviço pode ter, conforme especificado pelo consumidor durante o registro. A Equação 4 garante que cada IP seja atendido por exatamente uma região, ou seja, a mais próxima, enquanto assegura que um microsserviço seja implantado em até X regiões diferentes.

3.4. Definição de preço e leilões

O método de definição de preços é de extrema importância para que provedores garantam um lucro mínimo e que consumidores paguem um valor justo e competitivo pelos recursos efetivamente utilizados. No uServ é adotado um mecanismo de leilão, onde tanto provedores quanto consumidores podem realizar lances pelos n-RBs disponíveis. No momento que um consumidor registra um microsserviço, ele especifica o preço máximo que está disposto a pagar pelo n-RB requisitado. Já o provedor tem o seu lance mínimo calculado de acordo com o custo de energia, medido durante a avaliação de desempenho de sua 1-RB, expresso pelo consumo em Watts do seu PC durante o *benchmark* multiplicado pelo preço da energia elétrica em sua região. O lance mínimo também considera o seu ranqueamento no desempenho de sua 1-RB. Dessa forma, o Algoritmo 2 define o lance mínimo de uma 1-RB de cada PC numa determinada região.

Algoritmo 2 Algoritmo de cálculo do preço de uma 1-RB numa região

Require: O número n de PCs na região
Require: As pontuações S e os custos C, ambos de tamanho n
1: $\bar{s} = \frac{1}{n}\sum_{i=1}^{n} s_i$
2: $\bar{c} = \frac{1}{n}\sum_{i=1}^{n} c_i$
3: $F \leftarrow (0, 0, 0, \ldots, 0)$
4: **for** $i \leftarrow 1$ **to** n **do**
5: **if** $s_i > \bar{s}$ **then**
6: $sdev \leftarrow (s_i - \bar{s}) / \bar{s}$
7: $cdev \leftarrow (c_i - \bar{c}) / \bar{c}$
8: $F[i] \leftarrow (1 + \max(0, sdev - cdev)) \times c_i$
9: **else**
10: $F[i] \leftarrow c_i$
11: **end if**
12: **end for**
13: **return** F

O algoritmo visa aumentar o custo atribuído aos PCs que superam a pontuação média da região, garantindo assim um lucro mínimo para esses PCs, gerando a lista $F = (f_1, f_2, \ldots, f_n)$, onde f_i representa o preço mínimo do PC i, $i : 1 \leq i \leq n$. Finalmente, com cada PC tendo o preço mínimo para um 1-RB, o custo de um n-RB multiplicando o custo unitário por n.

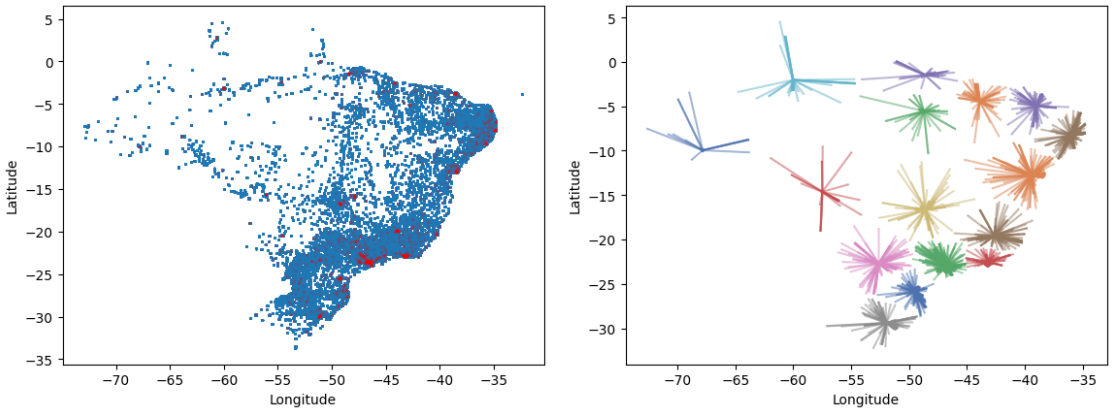
Após a obtenção dos preços mínimos dos n-RBs, o orquestrador coleta os microsserviços alocados àquela região e seus respectivos preços máximos e repassa-os para o leiloeiro. Nesse momento, o leiloeiro irá executar um leilão duplo que irá definir em qual PC um microsserviço será alocado. O mecanismo empregado pelo leiloeiro é um **leilão duplo de segundo preço com lance selado de rodada única**. Dado um conjunto de preços de compra $b = (b_1, \ldots, b_n)$ e um conjunto de preços de venda $s = (s_1, \ldots, s_n)$,

o sistema calcula o par de vetores de alocação, x e $y \in \{0, 1\}$, junto com os vetores de pagamento p e $q \in \mathbb{R}$, de modo que o número de vencedores seja igual entre compradores e vendedores. Para cada x_i tal que $x_i = 1$, o sistema garante que o pagamento correspondente p_i esteja dentro do intervalo de $0 \leq p_i \leq b_i$. Por outro lado, para cada y_i tal que $y_i = 1$, o pagamento correspondente q_i deve satisfazer a condição de $0 \leq s_i \leq q_i$. Estas condições são cruciais para manter a justiça e a viabilidade dentro do mecanismo de leilão, uma vez que restringe os pagamentos dentro dos limites de preço de compra/venda para cada vencedor. Além disso, por se tratar de um leilão de segundo preço, o participante vencedor paga ou recebe um valor equivalente ao segundo maior valor de lance ou venda. Estas condições tornam o leilão *truthful* e com propriedades onde o Equilíbrio de Nash é alcançado quando os participantes dão lances correspondentes a suas valorações reais [Deshmukh et al. 2002].

4. Experimentos e resultados

A fim de verificar o desempenho da proposta, simulações foram conduzidas com auxílio de dados reais de monitoramento coletados durante um período de 42 dias. Nesta seção serão apresentados os detalhes do estudo.

Dados demográficos obtidos do Instituto Brasileiro de Geografia e Estatística (IBGE) foram utilizados para criar um conjunto aleatório de PCs simulados que reflete a demografia da população brasileira[3]. Foram gerados 21 milhões de PCs simulados, iterando por cada cidade brasileira e gerando um número de PCs equivalente a 10% da população local. Os PCs foram distribuídos aleatoriamente nas coordenadas da cidade, dentro de um raio máximo equivalente à área da cidade. Após esse processo inicial, uma amostra de 2000 PCs foi selecionada para atuar como provedores. A Figura 2a mostra todos os PCs aleatórios em azul, bem como a amostra de provedores em vermelho.



(a) Distribuição dos PCs simulados. **(b) Regiões definidas dinamicamente.**

Figura 2. PCs simulados e a distribuição das regiões dos PCs selecionados.

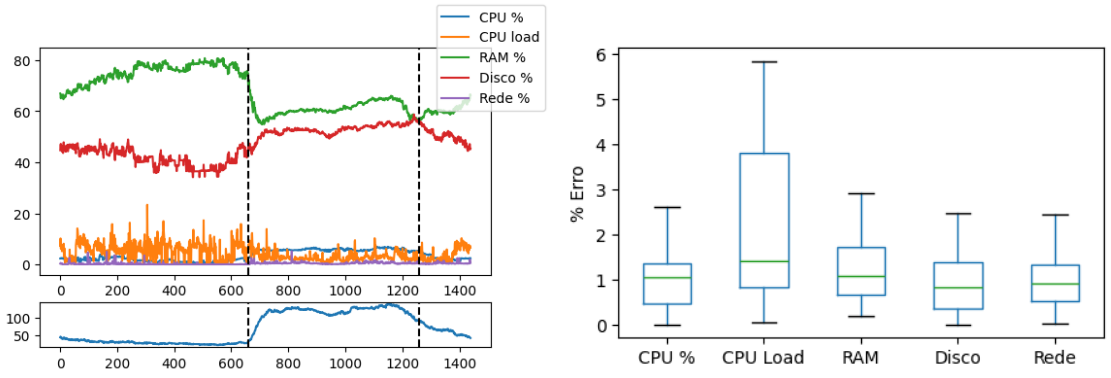
Após selecionar os 2000 provedores, o processo de definição de regiões foi realizado. A implementação do Affinity Propagation do *scikit-learn* foi utilizada para essa tarefa. O parâmetro *damping* foi definido como 0,95 e o parâmetro *preference* em -500, garantindo rápida convergência do algoritmo na definição de regiões. Ao término da

[3]Disponíveis em https://www.ibge.gov.br/estatisticas/downloads-estatisticas.html

execução do algoritmo, temos um total de 16 regiões, cada uma composta por aproximadamente 125 PCs. A distribuição geográfica resultante é mostrada na Figura 2b.

A simulação também considerou os preços atuais da eletricidade, estabelecidos pela Agência Nacional de Energia Elétrica (ANEEL)[4]. A simulação incorporou a suposição de que o consumo de energia de cada PC, em Watts, ao usar um 1-RB, segue uma distribuição de Pareto Tipo II com parâmetros $\mu = 65$ e $\alpha = 6$. Além disso, a simulação opera sob a suposição de que as especificações de hardware do PC são atribuídas aleatoriamente, abrangendo uma variedade de configurações: núcleos de CPU variando de 2 a 16, RAM de 4 a 32 GB, armazenamento de disco de 250 a 1000 GB e vazão de rede de 100 a 500 Mbps. Também se assume que a pontuação de um PC é calculada como dez vezes seu consumo de energia, aumentado por um valor aleatório retirado de uma distribuição normal caracterizada por $\mu = 0$ e $\sigma^2 = 100$.

Para aprimorar o realismo da simulação, um conjunto de 15 PCs reais foi monitorado ativamente ao longo de 42 dias para coletar métricas de consumo de recursos minuto a minuto. Essas métricas serviram como base para gerar dados sintéticos de monitoramento para todos os 2000 PCs. Para cada PC, um conjunto de séries temporais foi construído para cada métrica de consumo de recursos, calculando a média de todas as observações no mesmo minuto do dia. Isso resultou em 1440 pontos de dados por métrica por PC. A Figura 3a mostra as séries temporais resultantes a partir da média dos 15 PCs reais.



(a) Média das métricas de consumo. **(b) Erro de predição de métricas.**

Figura 3. Médias das métricas de consumo de recursos e suas margens de erro.

Durante a simulação, as métricas dos PCs simulados são definidas atribuindo uma das 15 séries temporais dos PCs reais a um PC simulado. Para evitar que vários PCs tenham métricas exatas durante a simulação, uma série temporal sintética é obtida adicionando um deslocamento médio e variância à série original. O deslocamento médio e a variância são valores aleatórios de uma distribuição normal com valores μ e σ^2 aleatórios, distribuídos uniformemente entre $[0 - 10]$ e $[0 - 5]$, respectivamente.

Como cada PC simulado possui sua própria série temporal de monitoramento sintética, é possível avaliar o método de predição da Equação 1. Para isso, é definida

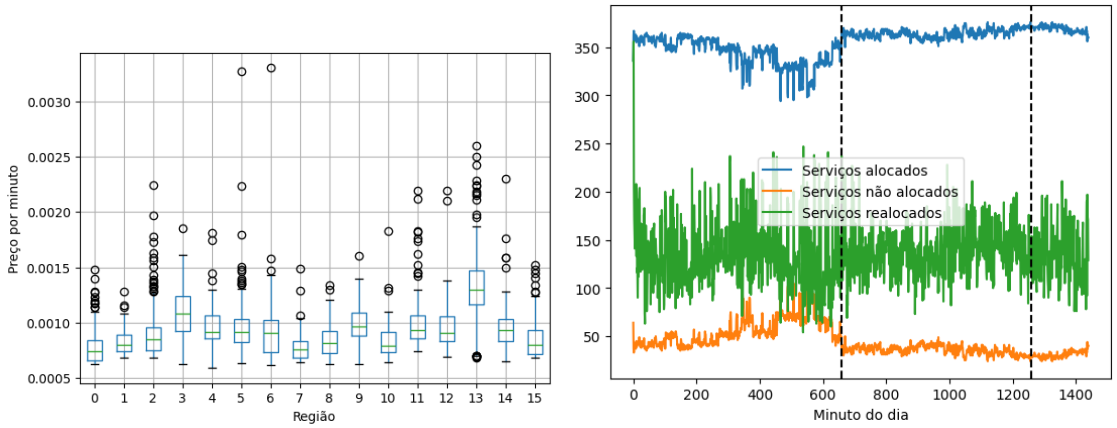
[4]Disponíveis em https://portalrelatorios.aneel.gov.br/luznatarifa/rankingtarifas

uma métrica com base no Erro Médio Absoluto (MAE):

$$\text{Erro} = \frac{\sum_{i=1}^{n} \max(0, y_i - x_i)}{n} \tag{5}$$

onde n é o número de amostras na série temporal, y_i é o valor previsto e x_i é o valor real para o tempo i. Esta métrica ignora erros nos quais o valor previsto fica abaixo do valor real. Como o objetivo do algoritmo é evitar a alocação de microsserviços durante períodos não ociosos nos PCs, é aceitável prever um consumo de recursos mais alto, pois isso reduziria a quantidade de recursos disponíveis para microsserviços. A Figura 3b mostra os valores de erro para os 2000 PCs utilizados na simulação. É importante mencionar que foram utilizados os valores $d = 5$ e $w = 10$ na Equação 1 durante a simulação.

O cálculo do preço mínimo por minuto que um provedor deve receber por um bloco de recursos 1-RB é determinado pela execução do Algoritmo 2. Dado que a simulação envolveu 16 regiões, todos os provedores foram agregados para calcular preços por região. Os resultados derivados do algoritmo, considerando despesas de energia e localização do provedor, são apresentados na Figura 4a. Os preços foram convertidos de real para dólar na proporção R$ 5,2 para US$ 1.



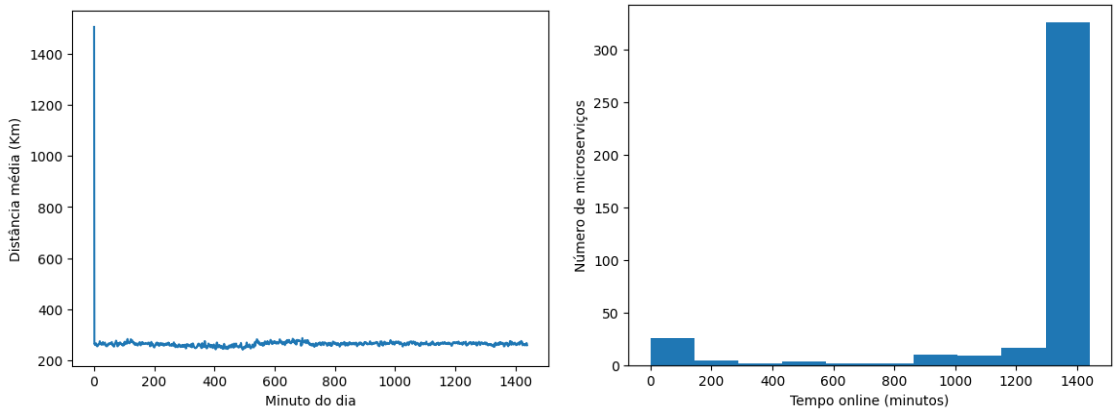
(a) Preços por minuto por região. **(b) Número de alocações durante o dia.**

Figura 4. Preços por minuto por região e número de alocações ao longo do dia.

Os microsserviços que vão consumir os recursos ociosos desses provedores também foram simulados. Nesse sentido, três conjuntos de microsserviços foram gerados, com 200, 400 e 600 microsserviços aleatórios. Cada microsserviço foi gerado com o número blocos de recursos (n-RBs) selecionando n aleatoriamente de 1 a 10. O preço máximo relativo a uma 1-RB foi definido seguindo uma distribuição normal, com μ igual ao preço da instância *t2.micro* da *Amazon Web Services* (AWS). O σ^2 da distribuição foi definido como $\mu \times 0,3$. O preço máximo das demais n-RBs foi definido proporcionalmente a 1-RB. Cada microsserviço tem um conjunto de 300 usuários selecionados aleatoriamente do conjunto de dados inicial (Figura 2a), todos posicionados num único estado brasileiro.

A Figura 4b ilustra o comportamento dos algoritmos de alocação ao longo de um dia simulado com o conjunto de 400 microsserviços. É evidente que a maioria

dos microsserviços foi implantada com sucesso. Notavelmente, durante o horário comercial, há um ligeiro aumento no número de alocações bem-sucedidas. O número de realocações de microsserviços apresenta uma volatilidade considerável ao longo do dia. Uma realocação ocorre quando há uma mudança no PC que executa um microsserviço entre a duração da alocação atual e a seguinte. Essa volatilidade pode ser atribuída a dois fatores: variações na disponibilidade de recursos entre provedores; e a necessidade de um microsserviço migrar de uma região para outra, impulsionada pela distribuição geográfica de seus usuários.



(a) Distâncias. **(b) *Uptimes*.**

Figura 5. Distância entre usuários finais e microsserviços e seus *uptimes*.

A Figura 5a exibe a média das distâncias entre os usuários finais de um microsserviço e a localização do PC responsável pela execução. No primeiro minuto do dia, que marca a primeira alocação do sistema, observa-se uma média de aproximadamente 1500 km. Essa média tem uma queda abrupta para cerca de 300 km, permanecendo estável até o final do dia. Esse padrão é resultado da estratégia inicial de alocação regional, que atribui regiões aleatórias aos microsserviços para registrar as localizações dos usuários. Posteriormente, as melhores regiões são designadas para os microsserviços com base nesses registros, reduzindo assim as distâncias médias e induzindo uma menor latência de propagação para as comunicações do microsserviço.

Adicionalmente, a Figura 5b exibe o histograma do *uptime* dos microsserviços após o término do dia simulado, no cenário de 400 microsserviços. É perceptível que a grande maioria dos microsserviços tiveram seu tempo acumulado próximo aos 1440 minutos do dia, indicando que ficaram disponíveis durante quase todo o dia. Por outro lado, aproximadamente 50 microsserviços tiveram *uptime* inferior a 1000 minutos. Isso pode ser explicado por duas razões: indisponibilidade de n-RBs de n elevado no mercado; ou custo da n-RB solicitada superior ao preço máximo estipulado pelo consumidor. Em ambos os casos, o microsserviço não é agendado a nenhum PC, tornando-o indisponível.

A Tabela 1 apresenta vários números importantes para a compreensão dos resultados obtidos. Analisando os números na perspectiva do número de microsserviços simulados, percebe-se que a medida que o número aumenta, aumenta-se também o número de realocações, o que tende a afetar negativamente os consumidores. Por outro lado, é perceptível que o lucro total do uServ aumenta a medida que o número de microsserviços aumenta. Esse lucro é obtido no *spread* existente entre o valor recebido do consumidor e

Número de microsserviços simulados	200	400	600
Percentual médio do número de serviços alocados	92%	88%	85%
Percentual médio do número de realocações de serviços	28%	34%	36%
Lucro total do uServ em R$	147	284	416
Lucro total do uServ em US$	28	54	80
Número de PCs com algum microsserviço alocado	456	708	882
Lucro médio de um provedor em R$	0.96	1.24	1.41
Correlação do lucro do provedor com o preço de 1-RB	-0.19	-0.16	-0.20
Correlação do lucro do provedor com sua disponibilidade	-0.06	-0.06	-0.07
Correlação do lucro do provedor com o *score* do PC	-0.01	-0.06	-0.11
Correlação do lucro com a distância a outros PCs da região	-0.02	-0.01	-0.02

Tabela 1. Resultados observados durante a simulação de 1 dia de operação

o valor pago ao provedor. Os provedores também são capazes de obter lucro líquido (descontado de despesas com energia) de forma proporcional à quantidade de microsserviços implantados no sistema, o que apresenta um incentivo real a participação do sistema. Uma análise de correlação de Pearson entre o lucro dos provedores e outras variáveis do sistema também é apresentada na tabela. É interessante notar que a disponibilidade dos PCs, o seus *scores* e a distância para outros PCs na mesma região não apresentam correlação significante com o lucro obtido. Por outro lado, o preço mínimo de uma 1-RB já apresenta uma correlação ligeiramente negativa.

5. Conclusão

O artigo apresentou o uServ, um sistema que monetiza recursos ociosos de PCs, permitindo a execução de microsserviços na borda. Ele oferece aos proprietários de PCs uma oportunidade de rentabilizar seus recursos, ao mesmo tempo em que fornece aos desenvolvedores de microsserviços uma plataforma de implantação mais próxima dos usuários finais. Com uma arquitetura que abrange tanto a nuvem quanto a borda, o sistema aborda desafios na gestão de recursos ociosos de PCs e alocação de microsserviços em regiões dinâmicas. O modelo de precificação, baseado em leilões regionalizados, considera os custos de energia e garantem a existência de mercados localizados onde blocos de recursos podem ser alocados aos microsserviços.

Os resultados da simulação validam o uServ como uma solução promissora para a execução de microsserviços nativos da nuvem na borda. Foi possível observar que o sistema não apenas facilita a implantação de microsserviços próximos aos usuários finais, mas também oferece uma oportunidade lucrativa para os proprietários de PCs aproveitarem efetivamente seus recursos subutilizados. No cenário simulado envolvendo a distribuição de 600 microsserviços em 882 PCs com recursos ociosos, foi alcançado um lucro diário médio de R$1,41, já descontando os gastos com eletricidade. Além disso, os desenvolvedores de microsserviços se beneficiaram, em média, pagando menos do que as despesas equivalentes incorridas ao utilizar hardware semelhante do provedor de nuvem AWS, pois a simulação considerou os preços da AWS como limite máximo de preço. Portanto, o uServ se mostra mutuamente vantajoso para consumidores e provedores.

Áreas não exploradas que serão abordadas em trabalhos futuros incluem segurança, minimização de realocações de microsserviços e refinamento da estratégia de

precificação das n-RBs, que pode se beneficiar de atributos dinâmicos ao invés de apenas estáticos. É de suma importância a abordagem de riscos de segurança, visto que a aplicação do uServ em ambientes públicos, não permissionados, incorre em riscos para os provedores, consumidores e usuários finais. Em ambientes permissionados o risco de segurança é mitigado dado a confiança pré-existente entre as partes, embora ainda exista possíveis tópicos a serem explorados em modelos *Zero Trust*.

Referências

Ali, B., Adeel Pasha, M., Islam, S. u., Song, H., and Buyya, R. (2021). A volunteer-supported fog computing environment for delay-sensitive iot applications. *IEEE Internet of Things Journal*, 8(5):3822–3830.

Deshmukh, K., Goldberg, A. V., Hartline, J. D., and Karlin, A. R. (2002). Truthful and competitive double auctions. In Möhring, R. and Raman, R., editors, *Algorithms — ESA 2002*, pages 361–373, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dueck, D. (2009). *Affinity propagation: clustering data by passing messages*. University of Toronto Toronto, ON, Canada.

Gasmi, K., Dilek, S., Tosun, S., and Ozdemir, S. (2022). A survey on computation offloading and service placement in fog computing-based iot. *The Journal of Supercomputing*, 78(2):1983–2014.

Hamdi, A. M. A., Hussain, F. K., and Hussain, O. K. (2022). Task offloading in vehicular fog computing: State-of-the-art and open issues. *Future Generation Computer Systems*, 133:201–212.

Hosseini, M., Angelopoulos, C. M., Chai, W. K., and Kundig, S. (2019). Crowdcloud: a crowdsourced system for cloud infrastructure. *Cluster Computing*, 22.0(2).

Luo, Q., Hu, S., Li, C., Li, G., and Shi, W. (2021). Resource scheduling in edge computing: A survey. *IEEE Communications Surveys & Tutorials*, 23(4):2131–2165.

Mavridis, I. and Karatza, H. (2023). Orchestrated sandboxed containers, unikernels, and virtual machines for isolation-enhanced multitenant workloads and serverless computing in cloud. *Concurrency and Computation: Practice and Experience*, 35(11):e6365.

Nedevschi, S., Chandrashekar, J., Liu, J., Nordman, B., Ratnasamy, S., and Taft, N. (2009). Skilled in the art of being idle: Reducing energy waste in networked systems. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, page 381–394, USA. USENIX Association.

Nguyen, K., Drew, S., Huang, C., and Zhou, J. (2020). Collaborative container-based parked vehicle edge computing framework for online task offloading. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–6.

Tabatabaee Malazi, H., Chaudhry, S. R., Kazmi, A., Palade, A., Cabrera, C., White, G., and Clarke, S. (2022). Dynamic service placement in multi-access edge computing: A systematic literature review. *IEEE Access*, 10:32639–32688.

Toffetti, G., Brunner, S., Blöchlinger, M., Spillner, J., and Bohnert, T. M. (2017). Self-managing cloud-native applications: Design, implementation, and experience. *Future Generation Computer Systems*, 72:165–179.