Revista Eletrônica de Iniciação Científica em Computação, 2026, 24:1
ISSN: 3085-8461 • doi: 10.5753/reic.2026.7573

Artigo de Pesquisa/Research Paper

FCR: Uma Abordagem de Replicação Dinâmica de Dados para a Computação em Névoa

FCR: A Dynamic Data Replication Approach for Fog Computing

Camila Vieira Figueiredo [Universidade Tecnológica Federal do Paraná | *camilavieira@alunos.utfpr.edu.br*]
Ana Cristina Barreiras Kochem Vendramin [Universidade Tecnológica Federal do Paraná |*criskochem@utfpr.edu.br*]

Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Av. Sete de Setembro, 3165, Curitiba, PR, 80230-901, Brasil.

Resumo. A replicação de dados é um tema amplamente estudado em sistemas distribuídos e consiste na criação e no posicionamento de cópias de dados em diferentes locais da rede. No contexto da computação em névoa, esses estudos buscam atender aos princípios dessa arquitetura, como o aumento da disponibilidade e a redução do tempo de acesso e de transmissão de dados. Como a computação em névoa propõe a aproximação dos serviços da borda da rede, a replicação de dados torna-se uma técnica fundamental para apoiar esse modelo. Apesar da existência de diversas abordagens, nenhum método é capaz de solucionar de forma abrangente todos os desafios associados à replicação de dados. Este trabalho propõe a abordagem *Fog-Closeness Replication* (FCR), uma estratégia de replicação dinâmica baseada em métricas de centralidade de proximidade e quóruns de leitura e escrita. A abordagem é avaliada por meio de simulações no ambiente iFogSim2, considerando métricas como latência de leitura e escrita, número de réplicas e taxa de leituras bem-sucedidas. Os resultados demonstram que a FCR reduz as latências de escrita e leitura em até 52,1% e 9%, respectivamente, e mantém 100% de taxa de leituras bem-sucedidas nas réplicas, quando comparada a uma abordagem baseada em alocação aleatória, mesmo em cenários com topologias dinâmicas e padrões de acesso variáveis.

Abstract. Data replication is a widely studied topic in distributed systems and consists of creating and placing data copies at different locations across the network. In the context of fog computing, such approaches aim to meet key architectural requirements, including increased data availability and reduced access and transmission latency. As fog computing brings services closer to the network edge, data replication becomes a fundamental technique to support this paradigm. Despite the existence of several approaches, no single method comprehensively addresses all challenges associated with data replication. This work proposes the Fog-Closeness Replication (FCR) approach, a dynamic replication strategy based on proximity centrality metrics and read and write quorums. The approach is evaluated through simulations using the iFogSim2 environment, considering performance metrics such as read and write latency, number of replicas, and successful read rate. The results demonstrate that FCR reduces write and read latency by up to 52.1% and 9%, respectively, and maintains 100% successful read rate across replicas when compared to a random allocation-based approach, even under dynamic network topologies and varying access patterns.

Palavras-chave: computação em névoa, replicação de dados, disponibilidade de dados, centralidade de proximidade, sistemas de quórum.

Keywords: fog computing, data replication, data availability, proximity centrality, quorum systems.

Recebido/Received: 07 February 2026 • **Aceito/Accepted:** 17 April 2026 • **Publicado/Published:** 24 April 2026

1 Introdução

O crescimento contínuo da Internet das Coisas (em inglês, *Internet of Things*, IoT) tem impulsionado a adoção de um número cada vez maior de dispositivos conectados, resultando na geração de grandes volumes de dados [Torabi *et al.*, 2022]. O uso exclusivo da computação em nuvem para o processamento e o armazenamento desses dados pode acarretar elevados tempos de latência e congestionamento da rede, comprometendo o desempenho das aplicações [Naas *et al.*, 2021; Torabi *et al.*, 2022].

Nesse contexto, a computação em névoa surge como uma alternativa ao estender os serviços computacionais para uma camada intermediária entre a nuvem e os dispositivos de borda. Esse paradigma tem se mostrado especialmente adequado para aplicações de IoT, que frequentemente operam em tempo real e exigem respostas rápidas e processamento distribuído em larga escala, uma vez que atrasos podem ocasionar falhas ou impactos significativos [Torabi *et al.*, 2022; Naas *et al.*, 2021; Habibi *et al.*, 2020].

Diante da crescente geração de dados e da necessidade de acesso eficiente a eles, a replicação de dados destaca-se como uma estratégia fundamental para o gerenciamento desses ambientes. A replicação consiste na criação e distribuição de múltiplas cópias dos dados em locais estratégicos, visando reduzir o tempo de acesso, aumentar a disponibilidade e melhorar o desempenho do sistema [Naas *et al.*, 2021; Sabaghian *et al.*, 2023]. Entretanto, em ambientes de computação em névoa, a distribuição geográfica dos nós torna o posicionamento inadequado de réplicas um fator crítico, podendo resultar em elevada latência durante o acesso ou a sincronização dos dados, o que afeta diretamente a qualidade de serviço [Naas *et al.*, 2021]. Assim, o desenvolvimento de abordagens específicas de replicação de dados para esse paradigma torna-se essencial.

A replicação de dados é amplamente investigada em diferentes contextos de sistemas distribuídos. A revisão sistemática conduzida por Karamimirazizi *et al*. [2024], que analisou estudos publicados entre 2019 e 2024 sobre replicação em ambientes de computação em nuvem, névoa, borda e borda-nuvem, identificou um total de 51 trabalhos. Dentre esses, apenas 6 abordam especificamente a computação em névoa, enquanto 29, 9 e 7 estudos tratam, respectivamente, de computação em nuvem, borda e borda-nuvem. Embora esses resultados dependam dos critérios adotados, observa-se que a replicação de dados em computação em névoa ainda é menos explorada em comparação a outros paradigmas, além de os trabalhos existentes concentrarem-se majoritariamente no problema de posicionamento de réplicas, deixando outras questões relevantes em aberto.

Além disso, a literatura aponta que a replicação de dados constitui um problema de elevada complexidade. Segundo Miloudi *et al*. [2021], trata-se de uma área propícia à investigação de diferentes abordagens, sendo frequentemente caracterizada como NP-difícil por diversos autores [Sabaghian *et al*., 2023; Torabi *et al*., 2022]. Essa característica inviabiliza a obtenção de soluções ótimas em tempo polinomial para cenários reais, reforçando a necessidade de propostas que busquem soluções eficientes e aplicáveis a diferentes contextos.

No âmbito da computação em nuvem, Naganandhini and Shanthi [2023] destaca que nenhuma abordagem existente é capaz de resolver de forma abrangente todos os desafios associados à replicação de dados. Essa constatação é corroborada pela revisão sistemática apresentada por Karamimirazizi *et al*. [2024], que aponta a ausência de soluções capazes de tratar simultaneamente todos os aspectos críticos da replicação em ambientes de computação em nuvem, névoa, borda e borda-nuvem. Diante desse cenário, a análise, a comparação e a proposição de novas abordagens tornam-se essenciais para o avanço da área, especialmente no contexto da computação em névoa.

Dessa forma, esse artigo propõe uma abordagem de replicação dinâmica de dados em ambientes de computação em névoa. Para atingir esse objetivo, o trabalho busca projetar uma arquitetura de computação em névoa adequada à abordagem proposta, implementar a solução em um ambiente de simulação e avaliar seu desempenho por meio de diferentes métricas de desempenho.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os conceitos fundamentais relacionados à computação em névoa; a Seção 3 apresenta as estratégias de replicação de dados; a Seção 4 descreve os trabalhos relacionados; a Seção 5 detalha a FCR e seus mecanismos de funcionamento; a Seção 6 apresenta o ambiente de simulação; a Seção 7 demonstra os resultados obtidos; e, por fim, a Seção 8 apresenta as conclusões e os trabalhos futuros.

2 Computação em Névoa

A computação em névoa surge como uma extensão da computação em nuvem, com o objetivo de aproximar serviços computacionais dos dispositivos finais, especialmente em cenários de Internet das Coisas (IoT), reduzindo latência e tráfego de rede [Habibi *et al*., 2020; Naas *et al*., 2021]. Nesse modelo, os serviços da nuvem são estendidos para nós intermediários distribuídos ao longo da rede, posicionados entre a infraestrutura de nuvem e os dispositivos finais. Diferentemente da computação de borda (*edge computing*), que ocorre diretamente nos dispositivos ou muito próxima a eles, a computação em névoa utiliza uma camada intermediária responsável pelo armazenamento, gerenciamento e processamento de dados. A Figura 1 ilustra a arquitetura composta pelas camadas de nuvem, névoa e borda (dispositivos finais).

Essa abordagem permite o pré-processamento e a análise parcial dos dados mais próximo da borda da rede, enviando à nuvem apenas informações relevantes, o que contribui para maior eficiência e menor atraso [Guerrero *et al*., 2020]. O ciclo de vida dos dados na névoa, apresentado na Figura 2, evidencia as etapas de coleta, processamento, análise, *feedback* e execução de comandos, destacando o papel intermediário da camada de névoa. Apesar de suas vantagens, a computação em névoa enfrenta desafios relacionados ao acesso e à disponibilidade dos dados, o que torna a replicação de dados uma técnica fundamental nesse paradigma [Torabi *et al*., 2022].

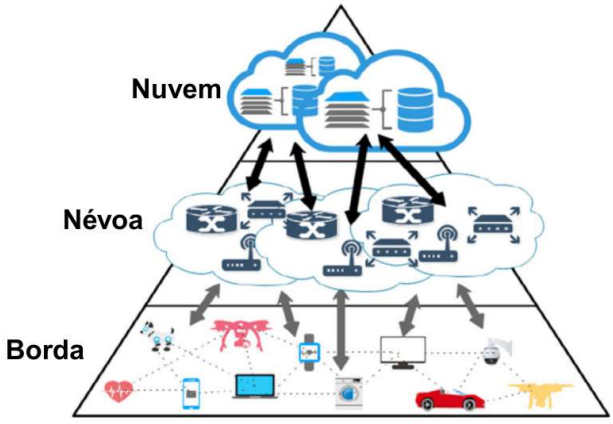

Figura 1. Arquitetura em camadas da computação em nuvem, névoa e borda.
Fonte: Adaptado de Naas *et al*. [2021].

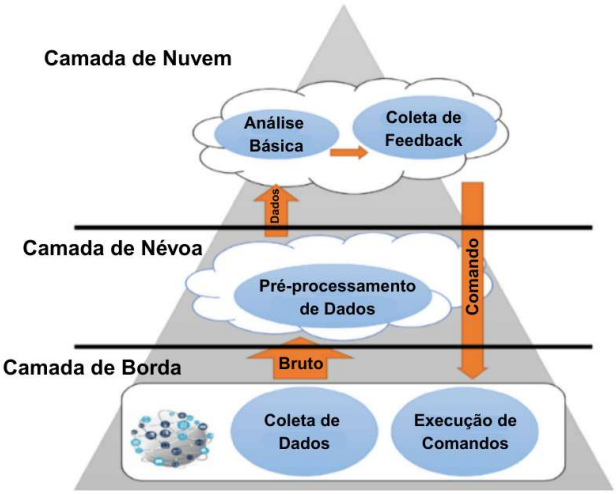

Figura 2. Ciclo de vida dos dados na névoa
Fonte: Adaptado de Torabi *et al*. [2022].

3 Replicação de Dados

A replicação de dados consiste na criação e manutenção de múltiplas cópias de dados em diferentes nós de um sistema distribuído, visando aumentar a disponibilidade, reduzir latência, melhorar a tolerância a falhas e balancear carga [Sabaghian *et al*., 2023; Mokadem and Hameurlain, 2020]. Em ambientes de computação em névoa, esse problema é considerado NP-difícil, devido à heterogeneidade dos nós e às restrições de recursos [Torabi *et al*., 2022].

As abordagens de replicação podem ser classificadas em estáticas e dinâmicas, sendo as dinâmicas mais adequadas a ambientes de névoa por se adaptarem às variações de acesso e às mudanças na topologia [Bouhouch *et al*., 2023]. De forma geral, a literatura identifica como principais problemas da replicação: seleção de réplicas, número de réplicas, tempo de replicação, posicionamento das réplicas, substituição de réplicas e consistência de dados [Karamimirazizi *et al*., 2024; Miloudi *et al*., 2021].

Além disso, as estratégias de replicação podem ser organizadas em não replicação, replicação completa e replicação parcial, sendo esta última a mais adotada em ambientes de névoa por equilibrar desempenho e consumo de recursos [Torabi *et al*., 2022].

4 Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados à replicação de dados em computação em névoa, considerando problemas como seleção, número, tempo e posicionamento de réplicas, substituição e consistência de dados. Em especial, considera-se que o problema do número de réplicas é abordado quando a quantidade não é fixa, sendo definida por critérios lógicos, de forma estática ou dinâmica.

Saleh *et al*. [2023] propõem uma abordagem dinâmica baseada em *publish-subscribe*, na qual cada dado possui um nó responsável por sua gestão. O número e o posicionamento das réplicas são definidos por meio de predição adaptativa e monitoramento dos padrões de acesso, considerando também caminhos mínimos na rede. A proposta aborda ainda tempo de replicação e consistência, sendo avaliada por métricas como latência, taxa de solicitações não atendidas e custo das réplicas.

A preservação da privacidade é o foco da abordagem de Sarwar *et al*. [2022], que utiliza o Nível de Privacidade (LoP) definido pelo proprietário do dado. Esse parâmetro orienta a seleção, o número e o posicionamento das réplicas, priorizando nós conforme capacidade e conectividade. A avaliação considera custos computacionais e de comunicação.

Já Taghizadeh *et al*. [2022] utilizam uma abordagem heurística baseada no *Non-dominated Sorting Genetic Algorithm II* (NSGA-II), combinada com o modelo de predição *Autoregressive Integrated Moving Average* (ARIMA), para definir o número e o posicionamento das réplicas a partir do histórico de acessos. O desempenho é avaliado considerando latência, custo, porcentagem da capacidade de acesso a pelo menos uma das réplicas e lucro estimado.

As abordagens iFogStorP e iFogStorS, propostas por Naas *et al*. [2021], diferem conforme o porte da infraestrutura, mas ambas definem o número de réplicas dentro de limites pré-estabelecidos, buscando minimizar a latência. O posicionamento é realizado considerando caminhos mínimos entre produtores e consumidores, e a consistência é tratada por meio da separação entre réplicas de leitura e escrita.

O posicionamento de réplicas por meio de particionamento de grafos é explorado por Guerrero *et al*. [2020], que define nós centrais em cada partição da rede. Já Huang *et al*. [2019] utilizam uma estratégia gulosa e recursiva para definir o número e o posicionamento das réplicas, visando à minimização da latência.

A popularidade dos dados, a energia dos nós e a mobilidade são consideradas por Vales *et al*. [2019], que tratam seleção, tempo e substituição de réplicas com base em limiares de demanda. Por sua vez, Naas *et al*. [2018] propõem uma abordagem de divisão e conquista baseada em particionamento de grafos para o posicionamento das réplicas.

Por fim, Mayer *et al*. [2017] abordam a replicação como um meio de integrar *Distributed Data Stores* (DDS) da nuvem à camada de névoa, tratando consistência por regiões e posicionamento por grupos de falha definidos por especialistas.

Um resumo dos problemas de replicação abordados por esses trabalhos é apresentado na Tabela 1. Em complemento dos trabalhos relacionados, este trabalho propõe o *Fog-Closeness Replication* (FCR), uma abordagem dinâmica que trata de forma integrada os principais problemas da replicação de dados. A FCR considera a seleção, o número, o tempo e o posicionamento das réplicas, bem como mecanismos de substituição e consistência.

5 *Fog-Closeness Replication* (FCR)

A presente seção descreve a *Fog-Closeness Replication* (FCR), uma abordagem dinâmica de replicação de dados aplicada à computação em névoa. A elaboração dessa abordagem levou em consideração o fluxo de trabalho dos processos, os problemas principais da replicação de dados e as abordagens já existentes que foram descritas na Seção 4.

5.1 Visão geral

A FCR fundamenta-se na métrica de centralidade de proximidade, em informações de acesso aos dados, como os *hits* de leitura, e nos quóruns de leitura e escrita, a fim de orientar o processo de replicação.

A centralidade de proximidade (*Closeness Centrality*) é uma métrica clássica da teoria de grafos que quantifica o quão próximo um nó está dos demais nós da rede, sendo definida como o inverso da soma das distâncias geodésicas mínimas entre esse nó e todos os outros nós do grafo [Freeman, 1979]. Dessa forma, nós com maior centralidade de proximidade apresentam menor distância média em relação aos demais, o que os torna mais adequados para funções que exigem baixa latência de comunicação.

Parte do funcionamento da FCR é centralizada na existência e disponibilidade de um nó líder, responsável por coordenar o processo de replicação, propagar atualizações da topologia da rede e direcionar as operações de escrita. A métrica de centralidade de proximidade é utilizada como critério de tomada de decisão para a seleção do nó líder e o posicionamento das réplicas, uma vez que nós mais centrais tendem a reduzir o tempo de acesso aos dados e o custo de comunicação na rede.

Tabela 1. Abordagens e Problemas de Replicação de Dados Abordados

Trabalhos Relacionados	Seleção	Número	Tempo	Posição	Substituição	Consistência
Saleh *et al*. [2023]		x	x	x		x
Sarwar *et al*. [2022]	x	x		x		
Taghizadeh *et al*. [2022]		x		x		
Naas *et al*. [2021]		x		x		x
Guerrero *et al*. [2020]				x		
Huang *et al*. [2019]		x		x		
Vales *et al*. [2019]	x	x	x	x	x	
Naas *et al*. [2018]				x		
Mayer *et al*. [2017]				x		x
FCR	x	x	x	x	x	x

A FCR considera dois tipos de réplicas: as permanentes e as temporárias. As réplicas permanentes são criadas na etapa de inicialização da topologia e obedecem ao fator mínimo de replicação. Por sua vez, as réplicas temporárias são criadas dinamicamente pelos mecanismos de replicação com o objetivo de explorar variações na demanda ao longo da topologia. Estas réplicas possuem um tempo de expiração, o qual pode ser estendido caso seja identificado que participam ativamente das operações de leitura. Esse conceito do tempo de expiração das réplicas temporárias tem inspiração no tempo de locação definido na abordagem de Vales *et al*. [2019].

Neste trabalho assume-se um ambiente livre de falhas, no qual os nós permanecem ativos durante a execução do sistema. Dessa forma, mecanismos explícitos de tolerância a falhas, como substituição do nó líder ou reconfiguração dinâmica de quórum, não são considerados no escopo desta proposta. Caso o nó líder se torne indisponível, o sistema deixa de coordenar novas operações de escrita, enquanto falhas em nós do quórum podem impedir que a maioria necessária seja alcançada.

Os mecanismos de replicação adotados pela FCR consideram tanto a frequência de acesso aos dados na topologia quanto a degradação da latência global do sistema. A frequência de acesso é avaliada a partir das operações de leitura, nas quais cada acesso a uma réplica gera um *hit* associado ao dado, servindo como um indicativo de sua popularidade. Quando o número de *hits* atinge um limiar predefinido, o processo de replicação é acionado. A degradação da latência global, por sua vez, é avaliada segundo um critério do tipo melhor–pior (*best–worst*). Para cada consumidor, considera-se a pior latência observada entre as réplicas que compõem seu quórum de leitura. Dentre esses valores, a maior latência é utilizada como referência global para a tomada de decisão, conforme está detalhado na próxima seção.

5.2 Funcionamento

Em relação ao funcionamento da FCR, podem ser identificados quatro elementos centrais: a inicialização do sistema, as operações de escrita, as operações de leitura e os mecanismos de replicação de dados. Esses elementos, juntamente com seus respectivos pseudocódigos, são descritos nas próximas subseções.

5.2.1 Inicialização do sistema

A inicialização do sistema corresponde à primeira etapa da FCR e requer um conjunto de informações iniciais, incluindo a topologia da rede, com as conexões e latências entre os nós, a identificação dos nós produtores e consumidores de cada dado, bem como o conjunto de dados que participarão do processo.

A partir da topologia inicial, calcula-se o caminho mínimo entre todos os pares de nós utilizando o algoritmo de Floyd-Warshall, considerando a latência como custo das arestas.

Em seguida, é calculada a centralidade de proximidade para cada nó. Para esse cálculo, soma-se, para cada nó, o custo de todos os seus caminhos mínimos até os demais nós da topologia, seguido de um processo de normalização. A centralidade de proximidade é definida conforme a Equação 1.

$$C(x) = \frac{N-1}{\sum\limits_{y \in V,\, y \neq x} d(x,y)}, \tag{1}$$

onde:

- $C(x)$ representa a centralidade de proximidade do nó x;
- N corresponde ao número total de nós da topologia que permitem armazenamento;
- V denota o conjunto de nós da topologia;
- y é um nó pertencente a V, distinto de x;
- $d(x, y)$ corresponde ao custo, em termos de latência, do caminho mínimo entre os nós x e y.

O nó com maior centralidade de proximidade é selecionado como o nó líder da topologia. Em seguida, são criadas as réplicas iniciais de cada dado, utilizando o P-Closeness para identificar os nós candidatos ao armazenamento dessas réplicas.

O P-Closeness adotado como mecanismo auxiliar é inspirado no P-Median empregado na abordagem proposta por Naas *et al*. [2021]. Esse método considera todos os nós pertencentes aos caminhos mínimos entre produtores e consumidores e seleciona os P nós que apresentam os maiores valores de centralidade de proximidade.

O pseudocódigo descrevendo o procedimento de inicialização é apresentado no Algoritmo 1. O parâmetro *minFatorRep* representa o fator mínimo de replicação necessário para garantir a disponibilidade dos dados no sistema, isto é, o número mínimo de réplicas permanentes que devem ser criadas para cada dado durante a inicialização da topologia.

Algorithm 1 Inicialização do sistema

Entrada: $topologia, produtores, consumidores, dados, minFatorRep$

```
caminhos ← calculaTodosCaminhosMinimos(topologia)
closeness ← calculaCentralidadeProximidade(topologia, caminhos)
lider ← escolhaLider(closeness)
para dado ∈ dados faça
    prodDado ← produtores(dado)
    consDado ← consumidores(dado)
    candidatos ← pCloseness(minFatorRep, caminhos, prodDado, consDado)
    criaReplicasIniciais(dado, candidatos)
fim para
```

5.2.2 Operação de escrita

A operação de escrita é iniciada a partir da solicitação de criação ou atualização de um dado por um sensor. Essa solicitação é encaminhada ao nó líder, que desempenha um papel central nesse processo, sendo responsável por coordenar o quórum de escrita e por decidir pela confirmação (*commit*) ou pelo aborto (*abort*) da atualização das réplicas.

Ao receber a solicitação de escrita, o líder verifica se já existe um processo de escrita em andamento para o mesmo dado. Em caso afirmativo, a solicitação é inserida em uma fila de espera. Quando a escrita pode ser realizada, o líder identifica os nós que armazenam réplicas do dado e lhes encaminha a solicitação de escrita contendo a versão atualizada.

Cada operação de escrita recebe um *timestamp* em milissegundos (*Unix epoch* – UTC), gerado pelo sensor responsável pela requisição e associado à versão do dado enviada às réplicas. Esse valor é utilizado para identificar a versão mais recente do dado durante operações de leitura. Embora essa abordagem dependa da sincronização aproximada dos relógios dos dispositivos, ela permite ordenar as versões dos dados de forma simples no cenário considerado.

Esses nós devem responder ao líder com uma confirmação de recebimento e permanecer aguardando a decisão final, que pode confirmar ou descartar a atualização. O sistema segue um modelo parcialmente síncrono, no qual não há limites rígidos conhecidos para o atraso das mensagens entre os nós. Para lidar com atrasos na comunicação, o nó líder utiliza temporizadores para aguardar confirmações de escrita e respostas dos nós do quórum. Caso o líder receba confirmações em número suficiente para atingir o quórum de escrita do dado, ele confirma a atualização; caso o tempo limite seja excedido, a operação é abortada. O quórum de escrita é definido conforme a Equação 2, variando de acordo com o número total de réplicas existentes para um determinado dado na rede em um dado momento.

$$QuorumEscrita(dado) = \frac{replicas}{2} + 1 \qquad (2)$$

O pseudocódigo descrevendo a operação de escrita realizada pelo líder é apresentado no Algoritmo 2.

Algorithm 2 Operação de escrita

Entrada: $nomeDado, dadoAtualizado$

```
se escrita de nomeDado não ocupada then
    ocuparEscrita(nomeDado)
    localReplicas ← obterLocalizacaoReplicas(nomeDado)
    confirmacoes ← 0
    para local ∈ localReplicas faça
        resposta ← enviarPedidoEscrita(dadoAtualizado, local)
        se resposta = confirmado then
            confirmacoes ← confirmacoes + 1
        fim se
    fim para
    quorum ← obterQuorumEscrita(nomeDado)
    se confirmacoes ≥ quorum then
        enviarConfirmarEscrita(localReplicas)
    senão
        enviarAbortarEscrita(localReplicas)
    fim se
    liberarEscrita(nomeDado)
    processarFilaEscrita(nomeDado)
senão
    adicionarFilaEscrita(nomeDado, dadoAtualizado)
fim se
```

5.2.3 Operação de leitura

A operação de leitura é iniciada a partir de uma solicitação gerada por um dispositivo de névoa, denominado leitor. O leitor encaminha a solicitação ao nó mais próximo, o qual assume a responsabilidade de coordenar a busca por uma réplica atualizada do dado, garantindo o atendimento ao quórum de leitura.

Ao receber a solicitação, o nó responsável notifica o líder sobre o *hit* de acesso ao dado e identifica, entre os nós que armazenam réplicas, aqueles mais próximos que podem compor o quórum de leitura. O quórum de leitura é definido conforme a Equação 3 e varia de acordo com o número total de réplicas existentes para o dado na rede no momento da operação.

$$QuorumLeitura(dado) = \frac{replicas}{2} + 1 \qquad (3)$$

Após o envio das solicitações de leitura às réplicas que compõem o quórum e o recebimento das respostas, verifica-se se o quórum de leitura foi atingido. Caso positivo, é identificada, entre as respostas recebidas, a réplica mais atualizada, a qual é então retornada ao leitor solicitante.

Caso o quórum não seja atingido nessa primeira tentativa, a solicitação de leitura é encaminhada aos demais nós que possuem réplicas do dado e que não integravam o quórum inicial. Em seguida, realiza-se uma verificação final para determinar se o quórum de leitura foi alcançado.

O pseudocódigo sobre a operação de leitura realizada pelo nó é apresentado no Algoritmo 3.

Algorithm 3 Operação de leitura

Entrada: $nomeDado$
1: $quorum \leftarrow obterQuorumLeitura(nomeDado)$
2: $enviarHitLider(nomeDado)$
3: $localReplicas \leftarrow obterLocalizacaoReplicas($ $nomeDado)$
4: $quorumInicial \leftarrow obterQuorumProximo($ $localReplicas)$
5: $listaReplicas \leftarrow \emptyset$
6: **para** $local \in quorumInicial$ **faça**
7: $resposta \leftarrow enviarPedidoLeitura(nomeDado,$ $local)$
8: **se** $resposta \neq \emptyset$ **then**
9: $listaReplicas \leftarrow listaReplicas \cup \{resposta\}$
10: **fim se**
11: **fim para**
12: **se** $|listaReplicas| \geq quorum$ **then**
13: $replicaAtualizada \leftarrow obterReplicaMaisAtual($ $listaReplicas)$
14: $retornarLeitura(replicaAtualizada)$
15: **senão**
16: **para** $local \in localReplicas$ **faça**
17: **se** $local \notin quorumInicial$ **then**
18: $resposta \leftarrow enviarPedidoLeitura($ $nomeDado, local)$
19: **se** $resposta \neq \emptyset$ **then**
20: $listaReplicas \leftarrow listaReplicas \cup$ $\{resposta\}$
21: **fim se**
22: **fim se**
23: **fim para**
24: **se** $|listaReplicas| \geq quorum$ **then**
25: $replicaAtualizada \leftarrow$ $obterReplicaMaisAtual(listaReplicas)$
26: $retornarLeitura(replicaAtualizada)$
27: **senão**
28: $retornarFalhaLeitura()$
29: **fim se**
30: **fim se**

5.2.4 Mecanismos de Replicação de dados

Os mecanismos de replicação de dados constituem elementos centrais da abordagem. Conforme mencionado anteriormente, a replicação é controlada por dois mecanismos complementares: a frequência de acesso aos dados e a degradação da latência da rede.

A replicação motivada pela frequência de acesso é dinâmica e reativa aos padrões de acesso observados na rede. Esse mecanismo avalia se há demanda suficiente para justificar a criação de novas réplicas, promovendo a exploração e o teste de possíveis localizações que possam reduzir a latência de acesso. Para esse fim, réplicas temporárias são criadas e posteriormente removidas caso não se mostrem efetivas.

Por outro lado, a replicação motivada pela degradação da latência, baseada no critério melhor–pior, desempenha um papel predominantemente corretivo. Esse mecanismo avalia se o estado atual da rede apresenta desempenho inferior quando comparado ao seu comportamento histórico, acionando a replicação como forma de mitigar aumentos significativos na latência.

O nó líder realiza periodicamente uma verificação para identificar se é necessária a criação de novas réplicas de um dado. Essa verificação inicia-se com o cálculo da latência global observada no momento, considerando o critério melhor–pior. Para cada consumidor do dado, é identificada a pior latência entre as réplicas que compõem o seu quórum de leitura. Em seguida, a pior dessas latências é adotada como a latência de referência atual do dado. Esse critério é caracterizado como melhor–pior por considerar, para cada consumidor, a pior latência dentro do melhor conjunto de nós capaz de atender a sua solicitação de leitura.

Em seguida, verifica-se se o número total de réplicas existentes é inferior ao fator máximo de replicação, o qual impõe um limite à criação excessiva de réplicas. Caso esse limite ainda não tenha sido atingido, os mecanismos de replicação são, então, considerados para a possível criação de uma nova réplica.

Primeiramente, verifica-se se o número total de *hits* de acesso de um dado ultrapassa o limiar predefinido. Em seguida, avalia-se se a pior latência atual, identificada no cálculo da latência global descrito anteriormente, é superior à menor pior latência registrada no histórico da rede. Caso qualquer uma dessas condições seja satisfeita, o processo de criação de uma nova réplica temporária é iniciado. Quando a criação é motivada pela frequência de acesso (*hits*), verifica-se inicialmente se mais de um nó reportou *hits* ao líder.

Se múltiplos nós tiverem reportado *hits*, o P-Closeness é empregado para identificar o melhor candidato à localização da nova réplica, considerando os caminhos mínimos entre os nós que registraram acessos e desconsiderando aqueles que já armazenam réplicas do dado. Caso apenas um nó tenha reportado *hits*, esse nó é selecionado como candidato ao armazenamento da nova réplica, desde que ainda não possua uma réplica do dado.

Como último recurso, também adotado quando a replicação é motivada pela degradação da latência global, o P-Closeness é aplicado considerando todos os nós produtores e consumidores do dado como referências.

Após a criação de uma nova réplica temporária, o líder atualiza as informações da topologia, incluindo os caminhos mínimos, as centralidades de proximidade e demais métricas relevantes, além de disseminar essas atualizações para os demais nós da rede.

Algorithm 4 Mecanismos de replicação

1: **para** $dado \in dados$ **faça**
2: $(atualPior, menorPior) \leftarrow$ $calculaLatenciaGlobal(dado)$
3: $totalReplicas \leftarrow obterTotalReplicas(dado)$
4: **se** $totalReplicas < fatorMaxRep$ **then**
5: **se** $obterHits(dado) > limiarReplicacao$ **then**
6: $criarNovaReplicaPorHits(dado)$
7: **senão**
8: **se** $atualPior > menorPior$ **then**
9: $criarNovaReplicaPorLatencia($ $dado)$
10: **fim se**
11: **fim se**
12: **fim se**
13: **fim para**

Algorithm 5 Método calculaLatenciaGlobal

Entrada: $dado$
$maiorLatencia \leftarrow 0$
$localReplicas \leftarrow obterLocalizacaoReplicas(dado)$
$consumidores \leftarrow obterConsumidores(dado)$
$quorum \leftarrow obterQuorumLeitura(dado)$
para $node \in consumidores$ **faça**
  $listaLatencias \leftarrow \emptyset$
  para $replica \in localReplicas$ **faça**
    $lat \leftarrow obterLatencia(node, replica)$
    $listaLatencias \leftarrow listaLatencias \cup \{lat\}$
  fim para
  $ordenar(listaLatencias)$
  $piorLatenciaQuorum \leftarrow listaLatencia[quorum - 1]$
  se $piorLatenciaQuorum > maiorLatencia$ **then**
    $maiorLatencia \leftarrow piorLatenciaQuorum$
  fim se
fim para
$setarAtualPiorLatenciaQuorum(maiorLatencia)$
se latencia demonstra estabilidade com historico **then**
  $setarMenorPiorLatenciaQuorum(maiorLatencia)$
fim se
$adicionarPiorLatenciaQuorumHistorico(maiorLatencia)$

Algorithm 6 Método criarNovaReplica

Entrada: $dado, repPorLimiar$
$produtores \leftarrow obterProdutores(dado)$
$consumidores \leftarrow obterConsumidores(dado)$
$localReplicas \leftarrow obterLocalizacaoReplicas(dado)$
$nosHit \leftarrow obterNosComHit(dado)$
se $repPorLimiar$ **then** ▷ Motivada por limiar
  se $|nosHit| > 1$ **then**
    $novaLocacao \leftarrow pClosenessExcecoes(1, nosHit, nosHit, localReplicas)$
    $enviaReplicaTemporaria(dado, novaLocacao)$
  senão se $|nosHit| = 1$ **then**
    se $nosHit[0] \notin localReplicas$ **then**
      $enviaReplicaTemporaria(dado, nosHit[0])$
    senão
      $novaLocacao \leftarrow pClosenessExcecoes(1, produtores, consumidores, localReplicas)$
      $enviaReplicaTemporaria(dado, novaLocacao)$
    fim se
  fim se
senão ▷ Motivada por latência
  $novaLocacao \leftarrow pClosenessExcecoes(1, produtores, consumidores, localReplicas)$
  $enviaReplicaTemporaria(dado, novaLocacao)$
fim se
$zerarHits(dado)$

5.3 Aspectos de Replicação Abordados pela FCR

Considerando os principais problemas relacionados à replicação de dados mencionados na Seção 3, a abordagem FCR atende a todos esses aspectos:

- **Seleção de réplicas:** realizada com base na relevância do dado e em sua demanda de acesso;
- **Número de réplicas:** definido a partir de um conjunto inicial fixo de réplicas permanentes e da criação dinâmica de réplicas temporárias com base na frequência de acesso, respeitando um fator máximo de replicação;
- **Tempo de replicação:** novas réplicas podem ser criadas, após o início da simulação, em duas situações: quando o acesso a um dado alcança o limiar estabelecido ou quando é identificada degradação da latência global da rede;
- **Posicionamento das réplicas:** determinado com base nos caminhos mínimos da topologia, na centralidade de proximidade dos nós e na localização dos produtores e consumidores de dados, de modo a identificar os nós mais adequados para armazenar novas réplicas temporárias;
- **Substituição de réplicas:** realizada considerando o tempo de expiração das réplicas temporárias, que são removidas caso não contribuam para o atendimento das solicitações de leitura, possibilitando a criação de novas réplicas em locais mais apropriados;
- **Consistência de dados:** garantida por meio do uso de quóruns de escrita e de leitura, assegurando que a atualização de uma réplica ocorra apenas quando a maioria dos nós envolvidos concorda e que as solicitações de leitura sejam atendidas, sempre que possível, por réplicas atualizadas. A FCR adota um modelo de consistência eventual com quóruns, não assegurando consistência forte, mas reduzindo a probabilidade de leituras desatualizadas.

6 Ambiente de Simulação

A abordagem FCR foi analisada através de um experimento controlado através de simulação. Esta seção apresenta o simulador utilizado, as principais implementações realizadas, a topologia selecionada, as parametrizações definidas e as métricas de desempenho escolhidas.

O ambiente de simulação empregado foi o iFogSim2, uma extensão do iFogSim, amplamente utilizada na literatura para modelagem e simulação de ambientes de computação em névoa, borda e IoT. O iFogSim2 foi apresentado por Mahmud *et al.* [2022] com o objetivo de superar limitações de simuladores anteriores, oferecendo suporte aprimorado a microsserviços, mobilidade e formação de *clusters*.

Embora o uso de simuladores implique simplificações na representação de ambientes reais, como abstrações da rede e dos dispositivos, eles oferecem um ambiente controlado e reprodutível, permitindo a avaliação sistemática de diferentes abordagens com menor custo e maior escalabilidade. Os experimentos foram executados em uma máquina equipada com processador Intel(R) Core(TM) i5-8250U e 8 GB de memória RAM, arquitetura de 64 bits, utilizando o sistema operacional Windows 10.

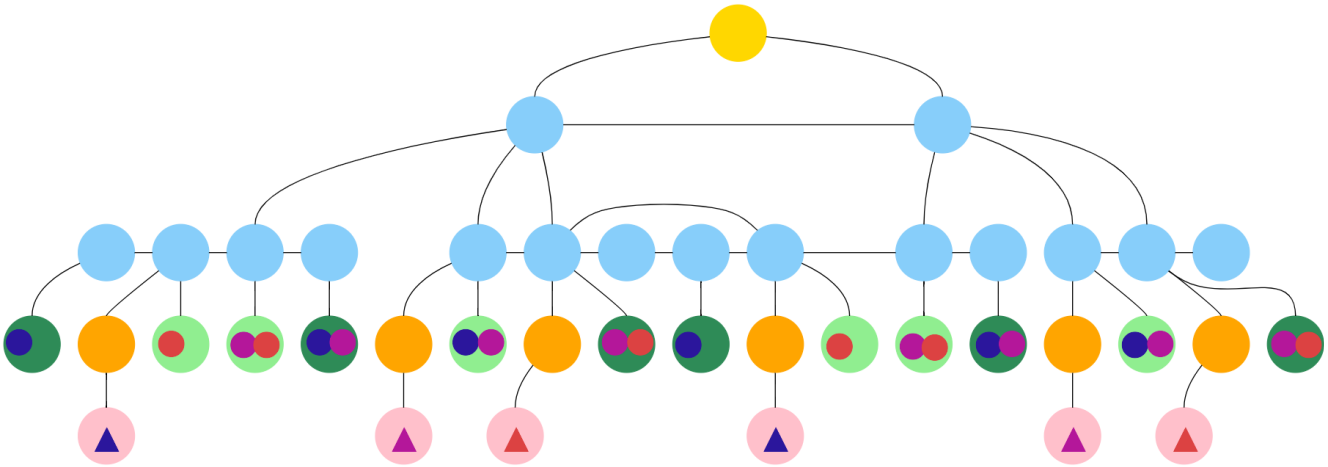

Figura 3. Topologia completa da simulação

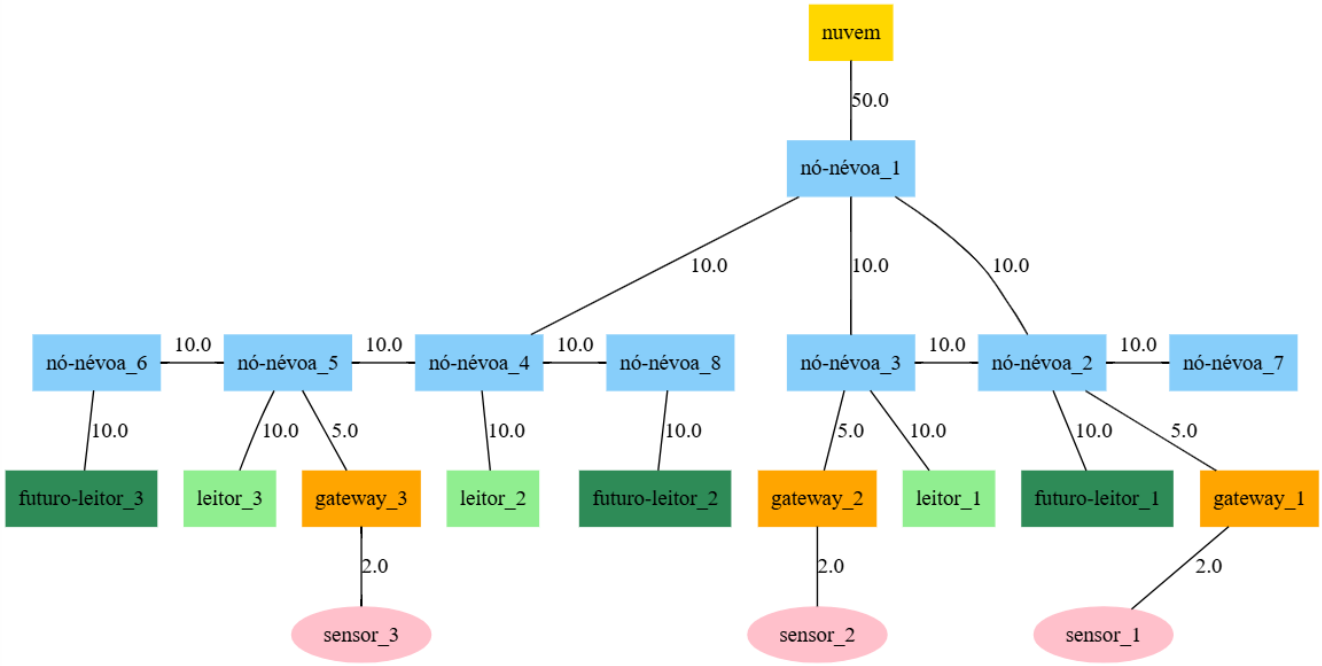

Figura 4. Topologia parcial da simulação

6.1 Implementações realizadas

Além dos mecanismos centrais da abordagem FCR, incluindo os processos de replicação e o método P-Closeness, foram necessárias extensões no iFogSim2 para viabilizar sua implementação, uma vez que a replicação de dados não é um foco principal do simulador.

Foram realizadas adaptações na modelagem das mensagens trocadas entre os nós, por meio da criação do objeto *OperationTuple*, bem como a implementação de um conceito de vizinhança entre dispositivos do mesmo nível da topologia, permitindo a comunicação baseada em caminhos mínimos. A classe *FogDevice* também foi estendida para suportar corretamente os papéis de leitores, futuros leitores e do nó líder. Adicionalmente, foram implementados fluxos específicos para operações de leitura e escrita, coordenação da replicação, gerenciamento de réplicas temporárias, entrada dinâmica de novos leitores na topologia e coleta das métricas necessárias para a avaliação do desempenho da abordagem.

6.2 Topologia

A topologia utilizada para a simulação da abordagem é representada nas Figuras 3 e 4. A Figura 3 apresenta a topologia completa da simulação, indicando os nós produtores e consumidores de cada dado, enquanto a Figura 4 apresenta um detalhamento de parte dessa topologia, em que os valores associados aos enlaces representam a latência, em milissegundos, considerada nas conexões entre os nós.

O cenário simulado representa um ambiente simplificado de computação em névoa, composto por sensores responsáveis pela geração de dados e por nós de névoa encarregados do armazenamento replicado dessas informações. Os sensores realizam solicitações de escrita encaminhadas aos *gateways*, que atuam como intermediários na comunicação, enquanto leitores executam requisições periódicas de leitura e leitores futuros tornam-se ativos em momentos posteriores da simulação, permitindo variações dinâmicas nos padrões de acesso. Os nós de névoa armazenam as réplicas e participam dos

Tabela 2. Parâmetros de configuração dos dispositivos de névoa

Parâmetro	Valor	Descrição
MIPS	1000	Capacidade de processamento do dispositivo em instruções por segundo
RAM (MB)	2048	Memória principal disponível
Uplink bandwidth (Mbps)	1000	Largura de banda de envio
Downlink bandwidth (Mbps)	1000	Largura de banda de recebimento
Busy power (W)	87.53	Consumo energético em estado ocupado
Idle power (W)	82.44	Consumo energético em estado ocioso

Tabela 3. Parâmetros de configuração dos *gateways* e leitores

Parâmetro	Valor	Descrição
MIPS	2800	Capacidade de processamento do dispositivo em instruções por segundo
RAM (MB)	4000	Memória principal disponível
Uplink bandwidth (Mbps)	10000	Largura de banda de envio
Downlink bandwidth (Mbps)	10000	Largura de banda de recebimento
Busy power (W)	107.339	Consumo energético em estado ocupado
Idle power (W)	83.433	Consumo energético em estado ocioso

quóruns de leitura e escrita, sendo que, no simulador, leitores, leitores futuros, *gateways* e nós de névoa são abstraídos como dispositivos de névoa.

Na Figura 3, os triângulos representam produtores, os círculos representam consumidores de dados, e a coloração dos símbolos identifica três itens de dados lógicos: azul para DATA1, roxo para DATA2 e vermelho para DATA3. Esses itens representam informações geradas por sensores e armazenadas nos nós de névoa. Eles não correspondem a conjuntos de dados distintos, mas sim a instâncias lógicas do mesmo tipo de informação, cada uma associada a um conjunto específico de produtores e consumidores na topologia. Essa configuração permite avaliar o comportamento das operações de leitura, escrita e replicação sob múltiplos fluxos simultâneos de dados.

6.3 Parâmetros de Simulação

A simulação demandou a definição de seus parâmetros de funcionamento. A Tabela 4 apresenta a parametrização referente ao custo de latência dos *links* de comunicação entre os diferentes tipos de nós considerados. Vale ressaltar que, nessa parametrização, os leitores são tratados como nós de névoa.

A configuração dos dispositivos de névoa na simulação é realizada por meio de um construtor que define os principais parâmetros computacionais, de comunicação e de consumo energético de cada nó. Esses parâmetros determinam a capacidade de processamento, memória disponível, largura de banda de comunicação e o comportamento energético dos dispositivos durante a execução das tarefas. A Tabela 2 apresenta os valores adotados para a criação dos dispositivos de névoa utilizados na simulação, enquanto a Tabela 3 apresenta os valores relacionados aos dispositivos leitores e *gatewayys*.

A Tabela 5 apresenta a parametrização referente ao número de dispositivos que fazem parte da topologia. Em complemento, a Tabela 6 apresenta quantos nós são produtores e consumidores de determinado dado da simulação.

Tabela 4. Parametrização da Latência entre Dispositivos

Relação entre dispositivos	Latência (ms)
Nuvem - Nó de névoa	50
Nó de névoa - Nó de névoa	10
Nó de névoa - *Gateway*	5
Gateway - Sensor	2

A Tabela 7 apresenta a parametrização referente ao fator de replicação, indicando o número de réplicas iniciais fixas e o número máximo de réplicas que cada dado pode atingir por meio do processo de replicação. Além disso, a tabela também apresenta o limiar utilizado para permitir a replicação motivada pela frequência de acesso. Já a Tabela 8 apresenta as parametrizações temporais relacionadas aos eventos e fluxos da abordagem FCR. O tempo máximo de simulação foi definido em 3000 ms, valor suficiente para permitir a execução completa das operações de leitura, escrita e replicação no cenário considerado.

Tabela 5. Parametrização do Total de Nós na Topologia

Dispositivo	Total
Nuvem	1
Nó de névoa	16
Leitor	6
Futuro Leitor	6
Gateway	6
Sensor	6

Tabela 6. Parametrização dos Produtores e Consumidores

Dado	Produtores	Cons. iniciais	Cons. futuros
DATA1	2	2	4
DATA2	2	4	4
DATA3	2	4	2

Tabela 7. Parametrização dos Fatores de Replicação

Fator	Valor
Fator mínimo	3
Fator máximo	7
Limiar de replicação	100

Tabela 8. Parametrização dos Tempos dos Eventos

Evento	Tempo (ms)
Expiração de réplica	100
Timeout verificação de replicação	100
Timeout confirmação de criação de nova réplica	100
Timeout quórum de escrita	100
Timeout quórum de leitura	100
Adição de um novo leitor	150

6.4 Métricas de avaliação

A avaliação da abordagem FCR considerou métricas essenciais para replicação de dados em ambientes de computação em névoa, incluindo a latência das operações de leitura e escrita, o número de réplicas por dado ao longo do tempo e a taxa de leituras bem-sucedidas nas réplicas.

7 Resultados

Os experimentos foram conduzidos com o objetivo de avaliar o desempenho da abordagem FCR considerando três métricas principais: latência das operações de escrita e leitura, número de réplicas por dado e taxa de leituras bem-sucedidas nas réplicas. Cada cenário foi executado dez vezes de forma independente, sendo analisados os valores médios obtidos, a fim de reduzir o impacto de variações inerentes ao ambiente de simulação e garantir maior estabilidade nas métricas observadas. Observou-se baixa variabilidade entre as execuções, indicando comportamento consistente dos resultados, o que tornou as dez repetições suficientes para a avaliação do modelo proposto no cenário considerado. Para fins de comparação, foi utilizada uma abordagem baseada em seleção aleatória de líderes e posicionamento de réplicas.

A latência média das operações de escrita é apresentada na Figura 5. Observa-se que a FCR manteve um comportamento estável, com baixo crescimento da latência mesmo com o aumento do quórum de escrita decorrente da criação de novas réplicas. Na média global das simulações, a abordagem proposta apresentou latência de escrita até 52,1% menor quando comparada à abordagem aleatória, a qual exibiu valores significativamente mais elevados ao longo da simulação.

Esse resultado indica que a FCR é capaz de realizar a escrita e atualização das réplicas de forma eficiente, mantendo a consistência dos dados por meio do posicionamento estratégico das réplicas na topologia da rede, próximas ao nó líder definido com base na centralidade de proximidade.

A latência das operações de leitura é apresentada na Figura 6. De forma semelhante ao observado nas operações de escrita, a FCR apresentou tendência à constância, mantendo baixos valores de latência ao longo do tempo. Embora a abordagem aleatória tenha alcançado valores próximos em determinados períodos, seu comportamento geral foi mais instável, refletindo decisões menos eficientes quanto ao posicionamento das réplicas.

A evolução do número de réplicas para cada dado é apresentada nas Figuras 7, 8 e 9. Observa-se que a FCR atinge o fator máximo de replicação de forma mais rápida e, após esse ponto, apresenta comportamento estável. Esse resultado

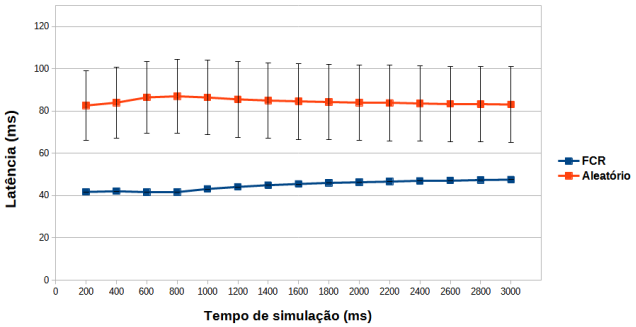

Figura 5. Latência média das operações de escrita.

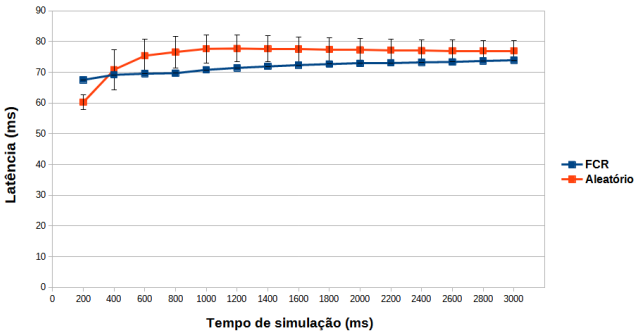

Figura 6. Latência média das operações de leitura.

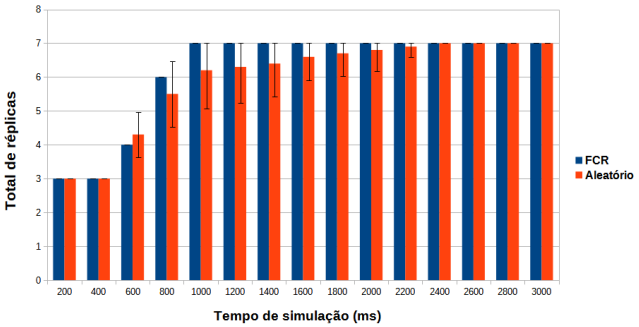

Figura 7. Número de réplicas do dado DATA1.

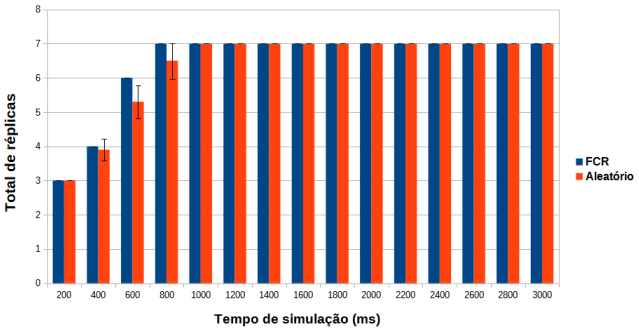

Figura 8. Número de réplicas do dado DATA2.

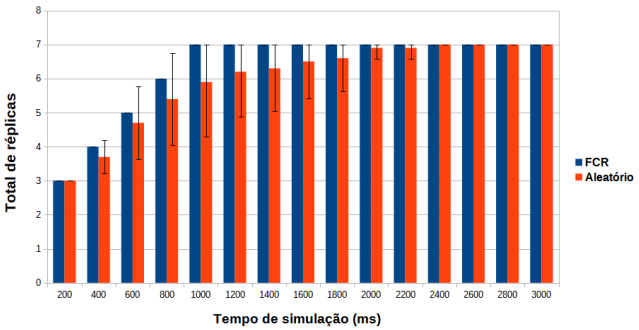

Figura 9. Número de réplicas do dado DATA3.

decorre do uso do mecanismo *P-Closeness*, que permite a criação de réplicas em localizações adequadas, garantindo sua participação contínua nos quóruns de leitura. Em contraste, a abordagem aleatória apresentou crescimento mais lento e instável no número de réplicas, impactando o atendimento às requisições.

A taxa de leituras bem-sucedidas foi avaliada como a proporção de solicitações de leitura atendidas com sucesso, conforme apresentado na Figura 10. A FCR apresentou desempenho elevado, sendo capaz de atender integralmente às

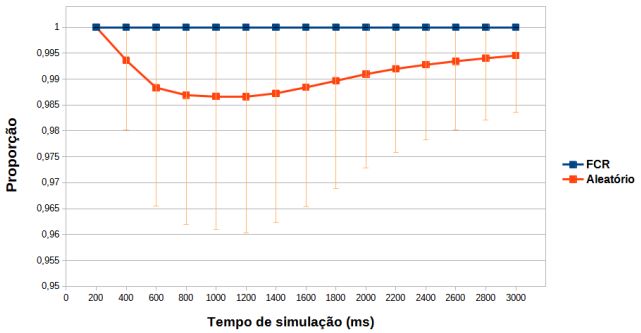

Figura 10. Taxa de leituras bem-sucedidas nas réplicas.

solicitações de leitura realizadas. Por outro lado, a abordagem aleatória apresentou uma redução média de 1,3% na taxa de leituras bem-sucedidas, associada a decisões menos eficientes de replicação.

De forma geral, os resultados demonstram que a FCR proporciona melhorias significativas em relação à abordagem aleatória, com reduções de até 52,1% na latência de escrita e até 9% na latência de leitura, além de prover uma taxa de leituras bem-sucedidas de 100%, mantendo a consistência dos dados. Esses resultados evidenciam a eficácia da abordagem proposta para ambientes de computação em névoa.

8 Conclusão e trabalhos futuros

A abordagem FCR apresentada neste trabalho demonstrou ser eficaz na replicação dinâmica de dados em ambientes de computação em névoa, atendendo aos principais problemas relacionados à replicação, incluindo seleção de réplicas, número de réplicas, tempo de replicação, posicionamento, substituição de réplicas e consistência de dados. A utilização de métricas como centralidade de proximidade, quóruns de leitura e escrita, e o mecanismo P-Closeness permitiu decisões dinâmicas e balanceadas quanto à criação e manutenção de réplicas.

Os resultados experimentais evidenciaram que a FCR é capaz de manter baixa latência nas operações de leitura e escrita, ao mesmo tempo em que assegura alta taxa de leituras bem-sucedidas nas réplicas, uma métrica fundamental para a efetividade de sistemas distribuídos em névoa.

Apesar dos resultados positivos, ainda existem desafios a serem explorados. A estratégia de seleção de réplicas pode ser aprimorada por meio da incorporação de novos critérios, como custo de armazenamento, capacidade dos nós e consumo de recursos. Além disso, há oportunidade de investigar alternativas para a definição do tempo de expiração das réplicas temporárias e mecanismos que avaliem a adequação do atendimento a novos leitores em cenários já estabilizados. A dependência de um nó líder para a coordenação da replicação também pode introduzir pontos únicos de falha, motivando a investigação de mecanismos de tolerância a falhas, estratégias de eleição de líderes e reconfiguração dinâmica de quórum. Adicionalmente, pretende-se avaliar o uso de identificadores incrementais de escrita em substituição aos *timestamps*, reduzindo a dependência da sincronização de relógios entre os dispositivos, bem como explorar cenários de simulação mais complexos, com diferentes cargas de trabalho, tamanhos de infraestrutura e tempos de execução dos experimentos. Por fim, trabalhos futuros incluem comparações mais abrangentes com outras abordagens de replicação em ambientes de névoa e borda, considerando métricas adicionais como escalabilidade, consumo de energia e desempenho em cenários de maior porte. Dessa forma, este trabalho evidencia que a FCR constitui uma solução promissora para a replicação dinâmica de dados em computação em névoa, abrindo espaço para extensões voltadas ao aumento da robustez, da escalabilidade e da eficiência do sistema.

Declarações complementares

Contribuições dos autores

Camila Vieira Figueiredo foi responsável pela conceitualização, investigação, metodologia, software e visualização. Ana Cristina Vendramin foi responsável pela supervisão e contribuiu com a revisão e edição do manuscrito. Todos os autores leram e aprovaram a versão final do artigo.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

O código do sistema estará disponível mediante solicitação.

Referências

Bouhouch, L., Zbakh, M., and Tadonki, C. (2023). A new classification for data placement techniques in cloud computing. In *2023 International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications, CloudTech 2023*. Institute of Electrical and Electronics Engineers Inc.. DOI: 10.1109/CloudTech58737.2023.10366156.

Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3):215–239. Disponível em: `https://scholar.google.com/scholar?cluster=4188774183582503557`.

Guerrero, C., Lera, I., and Juiz, C. (2020). Optimization policy for file replica placement in fog domains. *Concurrency and Computation: Practice and Experience*, 32(21):e5343. e5343 cpe.5343. DOI: https://doi.org/10.1002/cpe.5343.

Habibi, P., Farhoudi, M., Kazemian, S., Khorsandi, S., and Leon-Garcia, A. (2020). Fog computing: A comprehensive architectural survey. *IEEE Access*, 8:69105–69133. DOI: 10.1109/ACCESS.2020.2983253.

Huang, T., Lin, W., Li, Y., He, L., and Peng, S. (2019). A latency-aware multiple data replicas placement strategy for fog computing. *Journal of Signal Processing Systems*, 91:1191–1204. DOI: 10.1007/s11265-019-1444-5.

Karamimirazizi, F., Jameii, S. M., and Rahmani, A. M. (2024). Data replication methods in cloud, fog, and edge computing: A systematic literature review. *Wireless Personal Communications*, 135:531–561. DOI: 10.1007/s11277-024-11082-7.

Mahmud, R., Pallewatta, S., Goudarzi, M., and Buyya, R. (2022). ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, 190:111351. DOI: https://doi.org/10.1016/j.jss.2022.111351.

Mayer, R., Gupta, H., Saurez, E., and Ramachandran, U. (2017). Fogstore: Toward a distributed data store for fog

computing. In *2017 IEEE Fog World Congress (FWC)*, pages 1–6. DOI: 10.1109/FWC.2017.8368524.

Miloudi, I. E., Yagoubi, B., and Bellounar, F. Z. (2021). A survey of dynamic replication strategies in cloud computing systems. In *5th International Conference on Networking and Advanced Systems, ICNAS 2021*. Institute of Electrical and Electronics Engineers Inc.. DOI: 10.1109/ICNAS53565.2021.9628963.

Mokadem, R. and Hameurlain, A. (2020). A data replication strategy with tenant performance and provider economic profit guarantees in cloud data centers. *Journal of Systems and Software*, 159. DOI: 10.1016/j.jss.2019.110447.

Naas, M. I., Lemarchand, L., Boukhobza, J., and Raipin, P. (2018). A graph partitioning-based heuristic for runtime iot data placement strategies in a fog infrastructure. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 767–774, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3167132.3167217.

Naas, M. I., Lemarchand, L., Raipin, P., and Boukhobza, J. (2021). Iot data replication and consistency management in fog computing. *Journal of Grid Computing*, 19. DOI: 10.1007/s10723-021-09571-1.

Naganandhini, S. and Shanthi, D. (2023). Optimizing replication of data for distributed cloud computing environments: Techniques, challenges, and research gap. In *Proceedings of the 2nd International Conference on Edge Computing and Applications, ICECAA 2023*, pages 35–41. Institute of Electrical and Electronics Engineers Inc.. DOI: 10.1109/ICECAA58104.2023.10212287.

Sabaghian, K., Khamforoosh, K., and Ghaderzadeh, A. (2023). Data replication and placement strategies in distributed systems: A state of the art survey. *Wireless Personal Communications*, 129:2419–2453. DOI: 10.1007/s11277-023-10240-7.

Saleh, S. S., Alansari, I., Hamiaz, M. K., Ead, W., Tarabishi, R. A., and Khater, H. (2023). ifogrep: An intelligent consistent approach for replication and placement of iot based on fog computing. *Egyptian Informatics Journal*, 24(2):327–339. DOI: https://doi.org/10.1016/j.eij.2023.05.003.

Sarwar, K., Yongchareon, S., Yu, J., and ur Rehman, S. (2022). Efficient privacy-preserving data replication in fog-enabled iot. *Future Generation Computer Systems*, 128:538–551. DOI: https://doi.org/10.1016/j.future.2021.10.024.

Taghizadeh, J., Ghobaei-Arani, M., and Shahidinejad, A. (2022). A metaheuristic-based data replica placement approach for data-intensive iot applications in the fog computing environment. *Software: Practice and Experience*, 52(2):482–505. DOI: https://doi.org/10.1002/spe.3032.

Torabi, E., Ghobaei-Arani, M., and Shahidinejad, A. (2022). Data replica placement approaches in fog computing: a review. *Cluster Computing*, 25:3561–3589. DOI: 10.1007/s10586-022-03575-6.

Vales, R., Moura, J., and Marinheiro, R. (2019). Energy-aware and adaptive fog storage mechanism with data replication ruled by spatio-temporal content popularity. *Journal of Network and Computer Applications*, 135:84–96. DOI: 10.1016/j.jnca.2019.03.001.