**UNIVERSIDADE FEDERAL DO PAMPA**

**RONALDO AMATO DE SOUZA**

# ANÁLISE COMPARATIVA DA EFICIÊNCIA DE ALGORITMOS PARA CACHEAMENTO DE VÍDEO DO TIPO LOOK AHEAD PROJETADOS PARA PROXIES DE VÍDEO SOB DEMANDA

**Bagé**
**2026**

**RONALDO AMATO DE SOUZA**

# ANÁLISE COMPARATIVA DA EFICIÊNCIA DE ALGORITMOS PARA CACHEAMENTO DE VÍDEO DO TIPO LOOK AHEAD PROJETADOS PARA PROXIES DE VÍDEO SOB DEMANDA

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Bruno Silveira Neves

**Bagé**
**2026**

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

S729a Souza, Ronaldo Amato de

ANÁLISE COMPARATIVA DA EFICIÊNCIA DE ALGORITMOS PARA CACHEAMENTO DE VÍDEO DO TIPO LOOK AHEAD PROJETADOS PARA PROXIES DE VÍDEO SOB DEMANDA / Ronaldo Amato de Souza.

72 f.: il.

Orientador: Bruno Silveira Neves

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Pampa, Engenharia de Computação, 2026.

1. Video On Demand. 2. Proxy. 3. Caching. 4. Streaming. 5. Look Ahead. I. Título.

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
Universidade Federal do Pampa

**RONALDO AMATO DE SOUZA**

# ANÁLISE COMPARATIVA DA EFICIÊNCIA DE ALGORITMOS PARA CACHEAMENTO DE VÍDEO DO TIPO LOOK AHEAD PROJETADOS PARA PROXIES DE VÍDEO SOB DEMANDA

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 06 de Julho de 2026.

Banca examinadora:

Prof. Dr. Bruno Silveira Neves
Orientador
(UNIPAMPA)

Prof. Dr. Carlos Michel Betemps
(UNIPAMPA)

Prof. Dr. Milton Roberto Heinen
(UNIPAMPA)

Assinado eletronicamente por **CARLOS MICHEL BETEMPS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 13/07/2026, às 14:31, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **BRUNO SILVEIRA NEVES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 13/07/2026, às 22:17, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **MILTON ROBERTO HEINEN**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/07/2026, às 16:51, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2085597** e o código CRC **361A8313**.

Referência: Processo nº 23100.010883/2026-81 SEI nº 2085597

## RESUMO

O *Video on Demand* (VoD) é uma tecnologia que oferece conteúdos em vídeo para visualização conforme a preferência do usuário, a qualquer momento e em qualquer lugar. Sua popularidade advém principalmente da variedade de conteúdo disponível, possibilidade de controle do que é assistido e, principalmente, preços acessíveis. Tais facilidades, agregaram milhares de novos clientes para as plataformas, sendo que este volume tem criado impactos sobre a utilização de recursos da infraestrutura de rede global. Segundo a Decoded Magazine (2023), a Netflix foi responsável por consumir, em 2022, cerca de 15% do tráfego *downstream* global, montante significativamente superior em comparação com os dados de YouTube e TikTok. Índices desta proporção podem sobrecarregar servidores de vídeo facilmente, no que diz respeito à largura de banda e operações de input/output por segundo (*iops*). Uma alternativa em uso para aumento da eficiência dos serviços de VoD é a implementação de Redes de Distribuição de Conteúdo *(Content Distribution Network - CDN)*, que implementam, entre outros aspectos, uma rede de servidores substitutos (proxies de VoD) que atuam como caches para reduzir o fluxo sobre a espinha dorsal da rede e sobre os servidores primários do serviço VoD. Entretanto, a eficiência dos algoritmos de cacheamento utilizados nestes proxies é decisiva para o alcance dos objetivos destes sistemas. Neste âmbito, os algoritmos $CARTE_{BB}$ e $RTBC$ despontam como as duas principais soluções algorítmicas na perspectiva de uma nova abordagem ao cacheamento, focando na interação efetiva com clientes conectados ao servidor para a transmissão de vídeos, especialmente visando cenários de alta carga de clientes e dados. O presente projeto tem como propósito realizar uma análise comparativa destes dois algoritmos, suas vantagens e desvantagens, analisando parâmetros como vazão e taxa de acertos em um ambiente simulado e sob variadas circunstâncias de carga de trabalho. Posteriormente à análise comparativa, foi proposta uma solução híbrida que integra os mecanismos de reutilização temporal do $RTBC$ e de popularidade local do $CARTE_{BB}$. Os resultados obtidos indicam que essa abordagem apresentou desempenho superior na maior parte dos cenários analisados, mantendo resultados consistentes sob diferentes distribuições de carga e condições de operação.



**Palavras-chave:** Video On Demand. Proxy. Caching. Streaming. Look Ahead.

## ABSTRACT

*Video on Demand* (VoD) is a technology that provides video content according to user preferences, allowing access at any time and from any location. Its popularity is mainly driven by the wide variety of available content, the flexibility to control what is watched, and the affordability of subscription services. These characteristics have attracted a large number of new users to streaming platforms, increasing the demand for global network infrastructure resources. According to Decoded Magazine (2023), Netflix accounted for approximately 15% of global downstream Internet traffic in 2022, a substantially higher proportion than that observed for platforms such as YouTube and TikTok. Traffic volumes of this magnitude can place considerable pressure on video servers, particularly regarding bandwidth consumption and input/output operations per second (*IOPS*). One of the approaches adopted to improve the efficiency of VoD services is the deployment of *Content Distribution Networks* (CDNs). These architectures employ distributed proxy servers that operate as caches, reducing traffic over the network backbone and the load imposed on the origin servers responsible for content delivery. However, the effectiveness of these systems depends directly on the cache replacement algorithms implemented by the proxy servers. In this context, the $CARTE_{BB}$ and *RTBC* algorithms emerge as two representative cache management policies, focusing on the interaction between connected users and the video content being streamed, particularly under scenarios characterized by high client concurrency and data volume. The objective of this work is to perform a comparative analysis of these algorithms by investigating their strengths and limitations through performance metrics such as throughput and cache hit ratio under different workload conditions in a simulation environment. Based on the results of this analysis, a hybrid approach integrating the temporal reuse mechanism of *RTBC* with the local popularity strategy of $CARTE_{BB}$ was proposed. The experimental results indicate that the proposed algorithm achieved superior performance in most of the evaluated scenarios, maintaining consistent results across different workload distributions and operating conditions.



**Keywords:** Video On Demand, Proxy, Caching, Streaming, Look Ahead.

## LISTA DE FIGURAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ACM | Association for Computing Machinery |
| CARTE | Current demAnd Rather Than futurE |
| CARTE$_{BB}$ | Current demAnd Rather Than futurE - Block Based |
| CC | Chunk-based Cache |
| CCVC | Collapsed Cooperative Video Caching |
| CDN | Content Delivery Network |
| DC | Densidade de Clientes |
| FIFO | First In / First Out |
| ID | Identifier |
| IEEE | Institute of Electrical and Electronics Engineers |
| iops | input/output por segundo |
| HAL | Hyper Articles en Ligne |
| HTTP | HyperText Transfer Protocol |
| LFU | Least Frequently Used |
| LRU | Least Recently Used |
| PC | Prioridade de Cacheamento |
| PRT | Predictible Reuse Time |
| RTBC | Reuse Time Based Cache |
| SRAM | Static Random Access Memory |
| TCC | Trabalho de Conclusão de Curso |
| TER | Tempo Exato de Reuso |
| TS | Tamanho da Sequência |
| TV | TeleVisão |
| TTL | Time To Live |
| VoD | Video on Demand |

**SUMÁRIO**

# 1 INTRODUÇÃO

Atualmente, entregar e sustentar um sistema computacional tornou-se um desafio significativo quando se avalia a consistência e resiliência de serviços, visto que a tecnologia sofre frequentes e significativas mudanças em um intervalo curto de tempo. Observando a magnitude da Internet, os sistemas mais populares na Web sofrem gargalos devido às grandes demandas feitas em seus serviços. Este cenário têm como resultado o incremento do fluxo de dados a níveis que sobrecarregam a infraestrutura de rede, acarretando na perda de muitas solicitações e/ou redução da qualidade do serviço entregue aos usuários. No contexto da distribuição de conteúdo por *Video on Demand* (VoD), a replicação de conteúdos em servidores substitutos (proxies de VoD), distribuídos estrategicamente na infraestrutura da rede, constitui uma das principais estratégias para ampliar a capacidade de atendimento e reduzir a sobrecarga dos servidores de origem. Essa organização permite distribuir as requisições entre diferentes pontos da rede, reduzindo o tráfego sobre a infraestrutura principal e tornando a entrega do conteúdo mais eficiente. Essa estratégia fundamenta a utilização das Redes de Distribuição de Conteúdo (*Content Distribution Networks* – CDNs), amplamente empregadas pelos provedores de serviços de streaming. As demandas dos usuários, desde modo, são realocados para o servidor mais próximo, assim esta abordagem permite a redução de fluxo de rede e do tempo de resposta das solicitações, impactando em ganhos no que diz respeito às respostas das solicitações dos usuários (PATHAN; BUYYA; COMPUTING, 2006). Segundo a Cisco (2020), até o fim de 2023, quase dois terços da população global, ou seja, 66%, terão acesso à Internet. Foi estimado que o número total de usuários da Internet alcançou 5,3 bilhões, em comparação com os 3,9 bilhões (51% da população mundial) registrados em 2018.

O volume de dados gerados por vídeos na internet tem se mostrado significativamente elevado, sendo responsável por uma parcela dominante do tráfego global. Segundo estimativas do International Telecommunication Union (2024), o tráfego de vídeos no ano de 2022 foi responsável por 65% do tráfego global total da internet, com um aumento de 24% em relação ao ano anterior, abrangendo conteúdos sob demanda, transmissões ao vivo, plataformas de streaming, redes sociais e chamadas de vídeo. Esse tráfego associado aos vídeos destaca a crescente predominância desse formato de dados no uso global da rede, expondo desafios à infraestrutura das redes e exigindo soluções cada vez mais avançadas para suportar a distribuição de conteúdo para atender à demanda

crescente.

No que diz respeito à implementação de um sistema de *Video on Demand* (VoD), como um primeiro aspecto da organização do sistema, tem-se o conteúdo de vídeo que é armazenado em um servidor centralizado na forma de arquivos digitais compactados. Os usuários exploram um menu de opções através de uma aplicação e fazem suas seleções, que podem estar disponíveis gratuitamente ou mediante pagamento, onde o servidor inicia imediatamente a transmissão do vídeo escolhido.

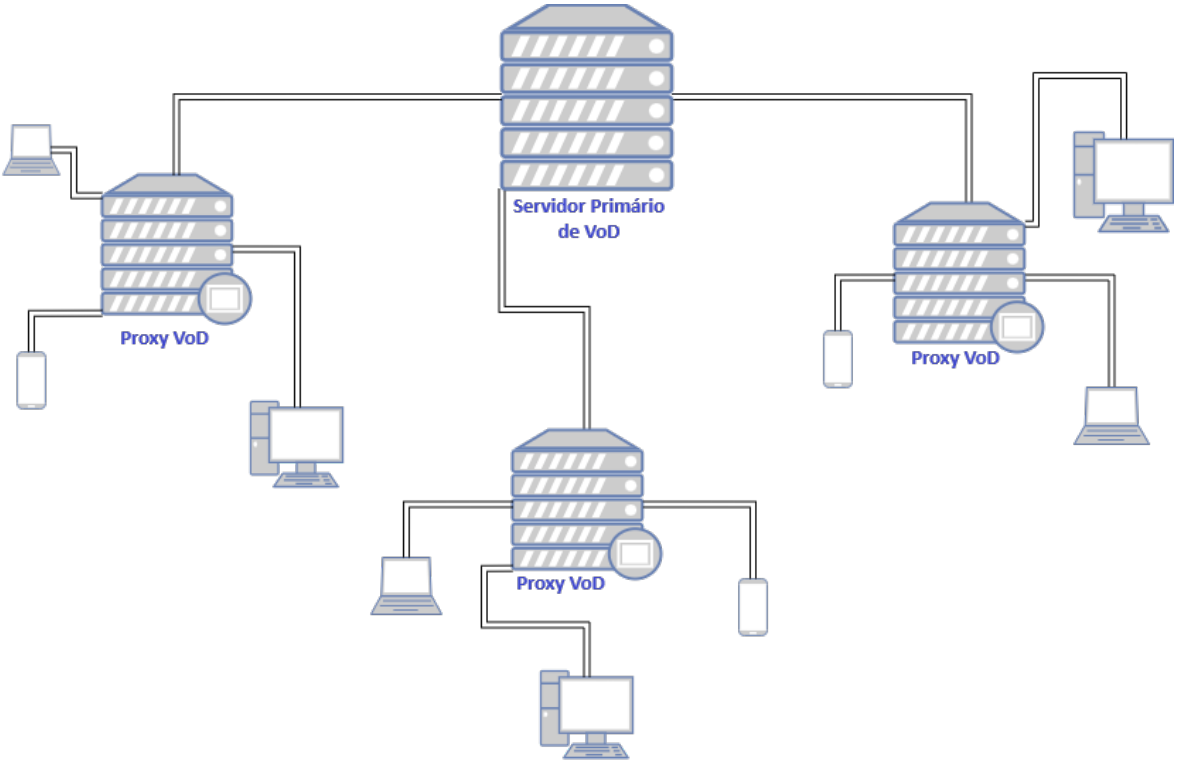
Estes espectadores têm a capacidade de pausar, avançar, retroceder ou até mesmo parar a exibição do vídeo, podendo posteriormente retomar a reprodução do arquivo de vídeo, sendo este sistema classificado em uma dentre duas principais abordagens: A abordagem tradicional cliente/servidor ou a abordagem de computação distribuída, como explica Kalan (2017).

Na abordagem cliente/servidor, os arquivos de vídeo são armazenados em um ou mais servidores centralizados. Cada cliente se conecta de forma independente ao servidor para reproduzir o vídeo solicitado. Embora o modelo cliente/servidor seja o mais simples, um aumento no número de clientes pode resultar em problemas de gargalo, afetando o desempenho do servidor.

Já a abordagem distribuída utiliza uma técnica de comunicação descentralizada e de múltiplos pontos, dividindo e balanceando as cargas sobre os servidores, buscando evitar problemas de gargalo do lado do servidor, economizando largura de banda.

Partindo deste princípio, segundo CloudFlare, Inc (2023), uma *Content Delivery Network* (CDN) é uma rede de servidores interconectados e fisicamente distribuídos, muitos dos quais atuam como caches para armazenamento de porções do conteúdo (acervo VoD) mais populares entre os clientes. Uma vez que CDNs estão globalmente distribuídas, elas possibilitam uma distribuição mais eficiente do conteúdo para um público amplo em comparação a um único servidor de origem. Conforme ilustra a Figura 1, sempre que um usuário realiza uma solicitação de conteúdo a um serviço de streaming que roda sobre uma CDN, este conteúdo pode, com base em suas regras de priorização para armazenamento dos dados, ficar temporariamente replicado ou *"cacheado"* em um servidor substituto, denominado proxy VoD, mais próximo ao cliente, permitindo que o mesmo seja aproveitado por demais clientes que tenham interesse pelo mesmo item de acervo, resultando em menor tráfego sobre a rede, e em um volume menor de processamento no servidor principal, sendo esta parte assumida pelo servidor substituto onde o conteúdo está replicado.

Figura 1 – Rede de Distribuição de Conteúdo.




Fonte: Do Autor (2026)

Como o espaço de armazenamento em um proxy de VoD corresponde a apenas uma fração daquele disponível no servidor primário de VoD, os algoritmos de cacheamento, que definem quais conteúdos devem ou não ser replicados no espaço de memória dos proxies VoD, desempenham um papel fundamental para otimizar o desempenho de armazenamento visando a maior parcela possível da demanda recebida pelo proxy. Desta forma, o proxy se comporta perante ao sistema VoD de forma análoga ao comportamento de uma cache na hierarquia de memória de um processador atuando como uma memória temporária, onde os dados mais populares (muitas vezes aqueles mais frequentemente ou recentemente acessados) são mantidos para acelerar o tempo de resposta para os clientes. No entanto, como o espaço no cache é limitado, é necessário utilizar algoritmos de cacheamento para gerenciar eficientemente o que deve ser mantido e o que pode ser descartado. Dentre os algoritmos mais conhecidos, o LRU *(Least Recently Used)* é um dos mais comuns, onde sua rotina elimina o arquivo de vídeo que não foi utilizado há mais tempo dentro da cache, assumindo que os mesmos não serão acessados novamente. Outro algoritmo muito conhecido é o LFU *(Least Frenquently Used)* que remove o conteúdo que foi acessado com menor frequência da cache. Também amplamente conhecido, o algoritmo FIFO *(First In, First Out)* remove o arquivo de

vídeo mais antigo, ou seja, o que entrou no cache primeiro, independentemente de sua frequência de uso ou do tempo em que permaneceu na cache.

Por outro lado, neste trabalho serão abordados algoritmos pertencentes a um paradigma emergente no âmbito dos sistemas de streaming, denominado por Neves (2015) de *"Look Ahead (Olhar para a frente)"*. O comportamento dos algoritmos que atuam sob este paradigma apresenta características de antecipação, buscando prever quais dados serão necessários no futuro com base em padrões de acesso dos clientes ativos (conectados ao sistema) do proxy, permitindo que o algoritmo tome decisões mais assertivas sobre quais conteúdos devem ser mantidos na cache, a fim de otimizar o desempenho futuro do sistema. Estes algoritmos não se baseiam apenas no histórico de acessos passados, mas também fazem previsões sobre os próximos acessos esperados (BELADY, 1966). Por consequência, a realização de uma avaliação comparativa do comportamento de algoritmos do tipo *"Look Ahead"* torna-se de grande relevância como estratégia para definir os próximos passos para a evolução, buscando o surgimento de novos e mais avançados algoritmos nesta categoria. Desta forma, são abordados nesta pesquisa, o algoritmo RTBC, ou "*Reuse Time Based Caching*"proposto por Wu et al. (2012) em comparação aos algoritmos $CARTE_{BB}$, ou "*Current demAnd Rather Than futurE - Block Based*", uma versão simplificada derivada do algoritmo CARTE original, desenvolvido por Neves (2015), e CC (*Chunk-based Cache*) criado por Hong, Vleeschauwer e Baccelli (2010). Todos estes algoritmos são classificados por Neves (2015) como algoritmos *"Look Ahead"*.

Uma vez que o foco do referido paradigma algorítmico é tratar situações de alta carga de trabalho, buscar-se-á comparar os seus desempenhos em cenários de carga mais extremos. Cenários extremos buscam contemplar, por exemplo, flutuações no número de clientes, mudanças de comportamento destes clientes na linha de tempo, alterações na velocidade dos links de internet disponíveis para replicação dos conteúdos de proxy, assim como outros fatores, esperando contrastar os desempenhos obtidos com aspectos comportamentais dos algoritmos, estabelecendo os pontos fortes para uma futura proposição de um algoritmo híbrido, modelado a partir destes modelos de referência.

Para viabilizar a análise comparativa desejada neste trabalho, por meio da execução e mensuração do desempenho destes algoritmos, pretende-se usar o simulador para proxies de vídeo sob demanda desenvolvido por Koch (2022). Este simulador atualmente dispõe de suporte para execução dos algoritmos $CARTE_{BB}$ e *CC*, sendo uma das propostas deste trabalho realizar dentro deste a implementação do algoritmo *RTBC*,

a fim de prover ao simulador a capacidade de realizar uma comparação mais ampla do conjunto de alternativas disponíveis para cacheamento de vídeo em cenários de alta carga.

## 1.1 Justificativa

Este trabalho justifica-se pela necessidade de se identificar, frente ao conjunto de alternativas mais promissoras para implementação da lógica de cacheamento dos proxies VoD, aquela capaz de prover mais alto desempenho em cenários operacionais que exigem alta escalabilidade e adaptação do proxy para atendimento da sua demanda, procurando identificar caminhos para a futura evolução do paradigma *"Look Ahead"*, tanto por meio de aprimoramento de um dos algoritmos já concebidos sob esta ótica ou até mesmo através da proposição de soluções híbridas que contemplem os pontos fortes de cada estratégia.

## 1.2 Objetivo Geral

Expandir um simulador de proxies *Video on Demand* (VoD) por meio da implementação de novas políticas de cacheamento do tipo *Look-Ahead* e avaliar seu desempenho em diferentes cenários de carga, incluindo condições representativas de ambientes reais de operação.

## 1.3 Objetivos Específicos

Considera-se que são objetivos específicos desse trabalho, os seguintes:

- Construir um novo módulo no simulador de Koch (2022) para suporte ao algoritmo RTBC e outro(s) que possam advir da análise pretendida neste estudo;
- Estudar e desenvolver um mecanismo que viabilize a realização de simulações em cenários realistas;
- Analisar a eficiência dos algoritmos de cacheamento simulados;
- Conhecer mais profundamente o funcionamento de um sistema de vídeo sob demanda, incluindo aspectos relativos a sua carga de trabalho;
- Compreender diferentes estratégias para cacheamento de vídeo, bem como suas

complexidades e seus impactos sobre a eficiência em um sistema de VoD.

## 1.4 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte forma: no Capítulo 2, são descritos a metodologia utilizada, os tipos de pesquisa empregados, os critérios de inclusão e exclusão adotados, bem como o processo de coleta de dados. No Capítulo 3, apresenta-se a fundamentação teórica que embasa o estudo, com foco nos algoritmos de cacheamento pertencentes ao paradigma look ahead. O Capítulo 4 apresenta os trabalhos relacionados e os dados coletados considerados relevantes para o desenvolvimento da pesquisa. O capítulo 5 dedica-se ao desenvolvimento do algoritmo selecionado, levando em consideração as condições estabelecidas pelo simulador de proxy VoD escolhido. No Capítulo 6, são expostos os resultados obtidos neste trabalho, apontando as principais observações obtidas até o momento. Por fim, o Capítulo 7 apresenta as conclusões decorrentes deste trabalho, destacando as principais contribuições obtidas e apontando possíveis direções para pesquisas futuras a serem exploradas a partir dos resultados apresentados.

# 2 MATERIAIS E MÉTODOS

Neste capítulo, serão apresentados os aspectos metodológicos adotados nesta pesquisa, que tem base em uma análise quantitativa. A metodologia envolve a coleta e análise de dados numéricos para avaliar o desempenho dos diferentes algoritmos de cacheamento tratados neste estudo. Serão detalhados os critérios de inclusão e exclusão de materiais relevantes, especialmente aqueles relacionados a algoritmos preditivos de cacheamento, buscando identificar as abordagens mais adequadas e alinhadas aos objetivos do estudo. Nas seções seguintes, esses aspectos serão aprofundados, em uma perspectiva de amparar o processo de avaliação e comparação dos algoritmos, almejado por este trabalho.

## 2.1 Tipo de pesquisa

De acordo com materiais dispostos por Prodanov e Freitas (2013), no que se refere ao tipo, a presente pesquisa contempla as seguintes modalidades:

- Enquadra-se como pesquisa bibliográfica, pois se baseia na consulta e análise de materiais já publicados, como livros, artigos científicos, teses, dissertações, e outros documentos relevantes, com o objetivo de obter uma compreensão abrangente e ao mesmo tempo aprofundada sobre os tópicos relevantes relativos ao tema estudado.
- Enquadra-se como pesquisa experimental, pois envolve a realização de experimentos controlados, pois serão realizadas simulações de diferentes opções algoritmicas para implementação do cacheamento de vídeo em proxies de video sob demanda, a fim de avaliar comparativamente a eficiência destas alternativas e explicar as diferenças entre os resultados por meio do apontamento de aspectos comportamentais presentes nestas implementações, seja para testar hipóteses ou avaliar o desempenho de variáveis em condições específicas.
- Enquadra-se como pesquisa hipotética dedutiva, pois as hipóteses serão baseadas na realização de uma análise comparativa da eficiência dos principais algoritmos de cacheamento, correlacionando esses dados com seus aspectos comportamentais. Com estes dados, será possível identificar os fatores determinantes para alcançar escalabilidade em cenários de variação de carga de trabalho ao sistema de streaming. Outra hipotese possível a estas simulações surge na possibilidade de

melhorar a qualidade do serviço ao reduzir a latência inicial para a transmissão de vídeos, que pode melhorar significativamente a percepção de fluência da execução do serviço. Os dados analisados são obtidos a partir de experimentos realizados com os algoritmos implementados no simulador de proxy VoD.

- Enquadra-se como pesquisa quantitativa, pois busca medir e quantificar as variáveis de interesse, como largura de banda utilizada, taxa de acertos no cache e sua capacidade, além da taxa de chegada de solicitações, que permitem a análise estatística dos dados.

Neste trabalho, a coleta de informações segue uma abordagem metodológica mesclada, integrando diferentes técnicas para garantir a robustez e a profundidade da análise. A principal estratégia adotada combina métodos experimentais e quantitativos de forma integrada. Os métodos experimentais, ou seja, as simulações, permitem analisar as variáveis de interesse de forma controlada, observando seu comportamento ao longo do tempo, oferecendo uma análise detalhada e dinâmica das hipóteses citadas anteriormente. Essa abordagem possibilita identificar relações causais e compreender as interações entre as variáveis em diferentes condições experimentais. Simultaneamente, o uso de métodos quantitativos fornece suporte estatístico, permitindo mensurar e analisar os resultados de forma objetiva e precisa, ampliando a confiabilidade e a profundidade da análise.

Em paralelo, utiliza-se o método quantitativo, que oferece uma base objetiva para analisar os dados coletados. Esse método permite comparar as amostras e realizar cálculos estatísticos que ajudam a identificar padrões, tendências e relações entre as variáveis investigadas, que podem surgir ao analisar hipóteses como eficiência do cacheamento, utilizando os resultados obtidos no tratamento da fila de espera e o tempo necessário para que o segmento de vídeo esteja disponível para cada algoritmo, visto que ambos gerenciam o espaço disponível em cache. A combinação dos dados experimentais com a análise quantitativa visa não apenas a validação dos resultados, mas também a ampliação da precisão das conclusões.

Adicionalmente, o trabalho faz uso do método hipotético-dedutivo, que desempenha um papel crucial na formulação e verificação de hipóteses. Por meio dessa abordagem, é possível estabelecer suposições fundamentadas sobre os estados da simulação investigados e consequentemente suas cargas de trabalho aplicadas e, a partir delas, derivar previsões que serão testadas. A relação de causa e efeito é, portanto, central nesse processo, permitindo elaborar explicações teóricas que possam ser validadas ou refutadas com base nos dados coletados durante as simulações.

## 2.2 Etapas para desenvolvimento deste TCC

Para alcançar os objetivos previstos e apresentados no Capítulo 1 deste texto, prevê-se, de forma alinhada à proposta de desenvolvimento do TCC, a execução das seguintes etapas:

### 2.2.1 Seleção dos trabalhos

A seleção dos artigos utilizados neste trabalho foi realizada com o objetivo de garantir que as fontes fossem as mais recentes possíveis, dentro de um cenário atual no que diz respeito aos algoritmos de cacheamento e métodos para simulação de proxies VoD. Para isso, estabeleceu-se como critério principal o recorte temporal de no máximo 10 anos, buscando refletir as tendências e avanços mais contemporâneos na área. Contudo, ao aprofundar-se em temas fundamentais para a pesquisa, como a arquitetura de cache, memória e redes, constatou-se que grande parte do material disponível data de períodos superiores a esse limite temporal. Esse fator é, no entanto, compreensível, pois muitas das bases teóricas e técnicas dessas áreas são consolidadas ao longo do tempo, o que justifica a utilização de publicações mais antigas. Do mesmo modo, uma revisão literária focada na possível identificação de algoritmos mais recentes e avançados foi realizada, com o objetivo de avaliar inovações e avanços nesse nicho, a fim de contribuir para uma análise mais completa e atualizada do estado da arte.

A seleção inicial dos materiais bibliográficos foi conduzida por meio de um processo de filtragem baseado na análise dos títulos e resumos dos documentos identificados durante a etapa de busca. Como critério de inclusão, considerou-se a aderência dos trabalhos aos objetivos da pesquisa e sua contribuição para os temas abordados ao longo deste estudo. Esse procedimento permitiu reduzir o conjunto inicial de resultados e direcionar a análise para os trabalhos mais relevantes ao contexto investigado. Ao final da triagem, foram selecionados 16 artigos que apresentaram maior alinhamento com os aspectos teóricos e metodológicos relacionados ao problema de pesquisa. Esses trabalhos forneceram subsídios para a fundamentação conceitual, para a compreensão das abordagens empregadas na literatura e para a análise dos resultados obtidos neste estudo. A relação completa dos artigos é apresentada na Tabela 1. As palavras utilizadas nos repositórios oficiais, foram *"Video On Demand, Proxy, Caching, Streaming, Look Ahead"*

Tabela 1 – Títulos Analisados

| **Titulo** | **Autor(es)** | **Utilizado** |
|---|---|---|
| Proxy Caching for Video-on-Demand Using Flexible Starting Point Selection | Tu et al. (2009) | Resultados contribuem: **NÃO** |
| Energy-Efficient Proactive Content Caching For On-Demand Video Streaming Under Uncertainty | Krishna, Nayak e Tumuluru (2022) | Resultados contribuem: **NÃO** |
| An Adaptive TTL Allocation Scheme for Real-time Live and On-Demand Personal Broadcasting Service | Kim et al. (2017) | Resultados contribuem: **SIM** |
| A study of replacement algorithms for a virtual-storage computer | Belady (1966) | Resultados contribuem: **SIM** |
| Warehouse-Scale Video Acceleration: Co-design and Deployment in the Wild | Ranganathan et al. (2021) | Resultados contribuem: **NÃO** |
| FSpot: Fast and Efficient Video Encoding Workloads Over Amazon Spot Instances | Zabrovskiy et al. (2022) | Resultados contribuem: **NÃO** |
| Video on Demand Streaming Using RL-based Edge Caching in 5G Networks | Nikbakht, Kahvazadeh e Mangues-Bafalluy (2022) | Resultados contribuem: **NÃO** |
| Reuse time based caching policy for video streaming | Wu et al. (2012) | Resultados contribuem: **SIM** |
| Development of Distributed Multimedia System for Ensuring Quality of Experience (QoE) on Video-on- Demand (VOD) Streaming Service | Amansyah e Bandung (2022) | Resultados contribuem: **NÃO** |
| Cooperative Announcement-based Caching for Video-on-Demand Streaming | Claeys et al. (2016) | Resultados contribuem: **SIM** |
| An Announcement-based Caching Approach for Video-on-Demand Streaming | Claeys et al. (2015) | Resultados contribuem: **SIM** |
| Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming | Rejaie e Kangasharju (2001) | Resultados contribuem: **NÃO** |
| A chunk-based caching algorithm for streaming video | Hong, Vleeschauwer e Baccelli (2010) | Resultados contribuem: **SIM** |
| Statistical characterization of a real video on demand service: User behaviour and streaming-media workload analysis | García et al. (2007) | Resultados contribuem: **SIM** |
| Methodologies for generating http streaming video workloads to evaluate web server performance | Summers et al. (2012) | Resultados contribuem: **SIM** |
| Characterizing the workload of a netflix streaming video server | Summers et al. (2016) | Resultados contribuem: **SIM** |

Fonte: Do Autor (2026)

### 2.2.2 Procedimento de coleta de dados

Este trabalho busca compreender o fluxo de distribuição de vídeos em ambientes digitais, com foco em estratégias de cacheamento de vídeo baseadas no paradigma *Look Ahead*. A pesquisa envolve uma revisão da literatura existente, análise comparativa de algoritmos de referência e identificação de avanços recentes no campo. Para isso, utiliza-se palavras-chave como *"Video on Demand"*, *"Proxy"*, *"Caching"*, *"Streaming"* e *"Look Ahead"* em bases de dados renomadas, como *IEEE Explore*, *ACM Digital Library* e *HAL Open Science*. Além disso, conceitos de arquitetura de memória são explorados a partir de bibliografias consagradas, como Patterson e Hennessy (2011). A pesquisa também investiga ambientes experimentais de simulação de proxies VoD, com ênfase na modelagem das condições de carga. As referências coletadas servirão de base para experimentos realizados com o simulador proposto por Koch (2022).

### 2.2.3 Procedimento de análise de dados

O processo de levantamento de dados para análise será conduzido a partir da execução de algoritmos em um ambiente simulado, onde testes serão repetidamente realizados conforme necessário, com a carga sendo progressivamente aumentada a cada nova execução. O objetivo dessa abordagem é avaliar o desempenho de cada algoritmo com um nível de confiança adequado, permitindo uma análise detalhada sobre como eles se comportam sob diferentes condições de carga.

Após a coleta e organização dos dados obtidos, é realizada uma tabulação cuidadosa, e para cada cenário testado, serão calculados os valores médios, os quais representarão o desempenho individual de cada algoritmo. Esses resultados serão então analisados de forma comparativa, permitindo identificar quais algoritmos se destacam em termos de eficiência e eficácia à medida que a carga aumenta.

A mensuração do desempenho será feita com base em critérios comparativos entre os algoritmos, focando especialmente na identificação dos que apresentam os melhores resultados frente ao aumento da carga. Para isso, usaremos como parâmetro de referência a taxa de acertos obtida pelo simulador, que servirá como o padrão para avaliar a precisão e a eficácia de cada algoritmo.

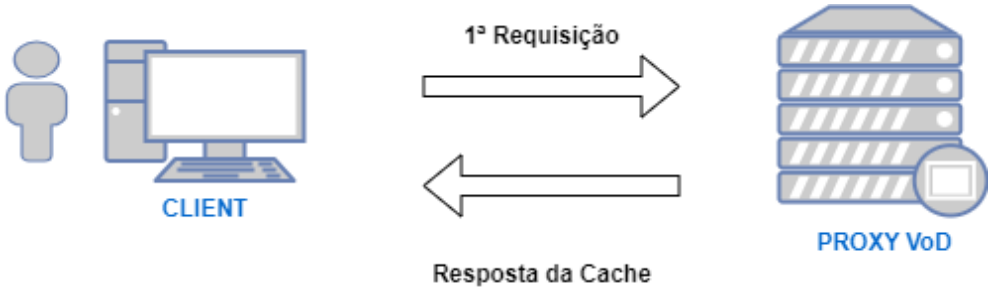
# 3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão abordados os conceitos teóricos fundamentais que sustentam a presente pesquisa. Inicialmente, serão apresentados os princípios básicos de cache, destacando seu papel essencial na otimização de sistemas de armazenamento e distribuição de dados. Em seguida, serão descritos os algoritmos de cacheamento que constituem o foco deste trabalho, explorando suas características e estratégias. Por fim, será detalhado o simulador que será utilizado para a execução dos algoritmos, fornecendo informações sobre suas funcionalidades e justificando sua escolha como ferramenta de análise neste estudo.

## 3.1 Princípios básicos de cache

Conforme pode ser visto em Patterson e Hennessy (2005), a palavra cache é utilizada para referenciar um local de armazenamento de acesso mais rápido devido a maior proximidade com a unidade de processamento dos dados utilizados, o que permite obter um aprimoramento do desempenho por meio das exploração das localidades de acesso. Em um sistema de proxy de vídeo, a cache funciona de modo similar. Segundo Wu et al. (2012), quando um cliente submete uma requisição, o proxy de vídeo atua como uma cache, verificando se o conteúdo desejado já está presente em seu espaço de armazenamento. Em caso afirmativo, o conteúdo é prontamente direcionado ao cliente, resultando em um *hit* na cache, ou seja, um acerto, como pode ser visto na Figura 2. Em situações em que o conteúdo não se encontra na cache, o proxy se

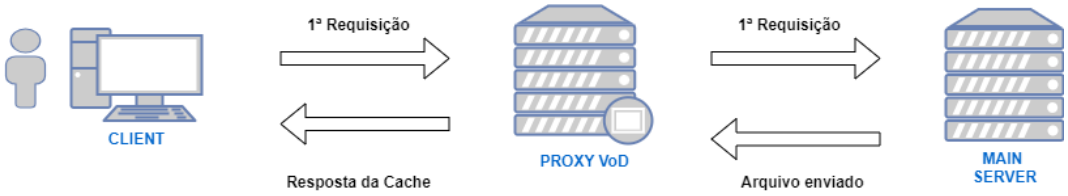
Figura 2 – Cache Hit.




Fonte: do Autor (2026)

encarrega de requisitá-lo ao servidor primário e, subsequentemente, encaminhá-lo ao cliente, resultando em um *miss*, ou falha, como pode ser visto na Figura 3. Neste caso,

Figura 3 – Cache Miss.




Fonte: do Autor (2026)

o sistema de cache precisa ter em sua unidade de controle, métodos de tratamentos de falhas de cache, que utilizam um algoritmo de cacheamento para determinar se o conteúdo solicitado e não localizado na memória do proxy deve ser armazenado e, caso deva, quais são conteúdos até então armazenados que precisam ser deletados do armazenamento do proxy, caso este armazenamento já esteja completamente cheio.

Segundo Patterson e Hennessy (2005), a taxa de acertos é a fração do total de acessos realizados ao cache onde o item buscado está presente. Complementarmente, a taxa de falhas corresponde a proporção de acessos realizados ao cache, onde os itens buscados não foram encontrados.

No âmbito de vídeo sob demanda, essa abordagem também acontece, onde os proxies VoD distribuídos geograficamente funcionam como caches entre os servidores principais e seus clientes, implementando parte da estrutura das CDNs (*Content Distribution Network*) (CloudFlare, Inc, 2023).

## 3.2 Exploração do simulador

Para a realização desta pesquisa o simulador implementado no Trabalho de Conclusão de Curso de Koch (2022) serve como base para as avaliações dos algoritmos propostas para este trabalho. Dois dos principais algoritmos *Look Ahead* já se encontram implementados, o que em parte se justifica seu uso neste TCC, e um terceiro foi implementado no escopo do presente trabalho, estando os três algoritmos brevemente descritos na Tabela 2, sendo sua lógica de funcionamento mais detalhadamente apresentada no próximo capítulo deste texto.

Um dos fatores determinantes para selecionar o algoritmo RTBC é que o mesmo utiliza como critério para cacheamento uma estimativa de distância entre cada segmento e seu cliente mais próximo, ou seja, o tempo exato de reuso para o segmento,

Tabela 2 – Características dos Algoritmos

| **Algoritmo** | **Características** | **Status da implementação** |
|---|---|---|
| CC | Baseia-se na contagem de quantos clientes ativos acessarão determinadas porções do acervo. | Já implementado |
| $CARTE_{BB}$ | Baseia-se na contagem de quantos clientes ativos acessarão determinadas porções do acervo, implementando uma janela temporal para limitar o escopo de contagem. | Já implementado |
| RTBC | Estimativa de distância entre cada segmento e seu cliente mais próximo. | Implementado durante este trabalho. |

Fonte: Do Autor (2026)

diferenciando-se, portanto, da atuação dos demais algoritmos.

Além disso, o simulador está com seu código fonte disponibilizado para edição, o que foi um dos critérios de seleção do mesmo para execução desta pesquisa, assim como possui linguagem de código aberto, acesso a documentação e se necessário, ao desenvolvedor, para esclarecimentos acerca de seu uso.

Também é possível selecionar o algoritmo de cacheamento que será executado através de um menu disponível e, após finalizar sua execução, compará-los, com a finalidade de analisar o código e identificar pontos a serem modificados para inclusão do novo algoritmo.

Durante o processo de busca por artigos relevantes sobre algoritmos de cacheamento de vídeo, constatou-se que grande parte dos serviços de streaming atuais utiliza algoritmos proprietários, cujas especificações e comportamentos diante de diferentes cargas de trabalho não são acessíveis ou disponibilizados publicamente.

## 3.3 Algoritmo CARTE - Current demAnd Rather Than futurE

Neves (2015) propõe em sua abordagem cenários que sugerem situações de maior demanda, prevendo que o proxy adote decisões criteriosas ao alocar seus recursos, priorizando as demandas imediatas em um horizonte de tempo de alguns segundos no futuro. Isso visa maximizar o uso do conteúdo armazenado no cache do proxy, priorizando o uso de recursos do proxy para atendimento das demandas mais urgentes de curto prazo.

Suas características fundamentais podem ser delineadas através dos seguintes elementos:

- Acompanhar e priorizar o cacheamento de conteúdos para os quais observa-se uma alta da densidade de clientes ativos posicionados em um horizonte de tempo futuro e de curto prazo, de modo que estes clientes tendem a assistir a estas porções do acervo.
- Responder às atuais condições de carga por meio da atualização do conteúdo na memória.
- Utilizar a capacidade de processamento disponível no hardware para aprimorar a escalabilidade do proxy, sobretudo alocando o hardware disponível de forma a priorizar a entrega de conteúdos com grande demanda a curto prazo.

A estratégia de gerenciamento de cache prioriza a permanência, na memória, dos segmentos que estão em uso ou que ainda serão requisitados por um ou mais clientes, evitando sua remoção enquanto permanecem necessários para a continuidade da reprodução dos vídeos. Quando existe espaço disponível, novos segmentos são armazenados no cache à medida que são requisitados. Caso a capacidade de armazenamento seja insuficiente, o proxy inicia o processo de substituição, liberando espaço por meio da remoção de conjuntos de blocos previamente entregues aos clientes e classificados com menor prioridade de reutilização. Dessa forma, a política de cache busca preservar os segmentos com maior probabilidade de acesso futuro, reduzindo a ocorrência de substituições que possam comprometer o atendimento de novas requisições.

Para aprimorar o gerenciamento de prioridades, o algoritmo vincula prioridades de cacheamento a trechos de vídeo de tamanhos variáveis denominados *Sequências*, onde cada sequência é formada por um conjunto de blocos de vídeo compreendidos entre dois clientes adjacentes que assistem a um mesmo vídeo.

A decisão sobre quais sequências de dados devem ser excluídas será tomada com base na prioridade de cacheamento das sequências disponíveis. Isso será determinado pela combinação dos seguintes fatores:

- A posição inicial, que corresponde à localização atual do primeiro bloco da sequência do filme.
- O tamanho da sequência, isto é, o número de segmentos pertencentes à sequência.
- A densidade de clientes: o total de clientes presentes em uma janela de $n$ blocos posicionados a frente de cada sequência e que possivelmente acessarão a referida sequência.

Em resumo, o cálculo da prioridade de cacheamento *(PC)* é a métrica que visa

determinar a ordem de armazenamento de sequências de dados na cache, levando em consideração tanto a Densidade de Clientes *(DC)* quanto o Tamanho da Sequência *(TS)*. A prioridade é calculada pela razão entre a *DC* e o *TS*, multiplicada por um fator adicional *X*, conforme a Equação 1.

$$PC = (DC/TS) * X \tag{1}$$

onde:

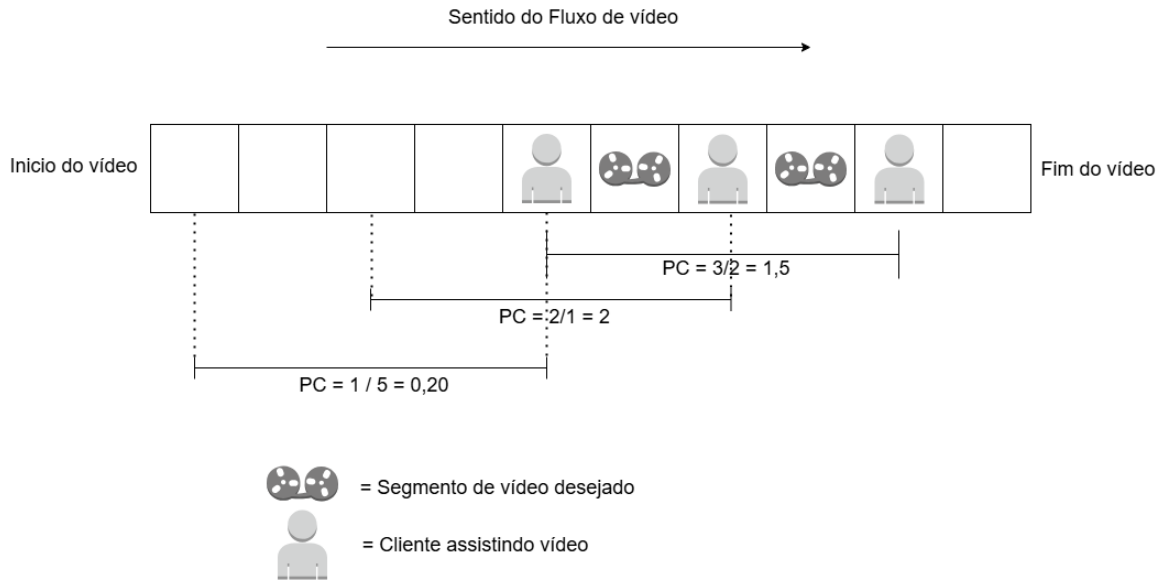
- *DC* representa o número de clientes que estão acessando ou requisitando dados daquela sequência específica. Sequências com maior densidade de clientes têm maior prioridade, pois a alta demanda sugere que esses dados são mais relevantes e precisarão ser rapidamente entregues a múltiplos usuários.
- *TS* é o tamanho da sequência de blocos de segmentos em questão. Sequências menores tendem a ser priorizadas, pois ocupam menos espaço de memória e podem ser manipuladas mais rapidamente. Sequências maiores, por outro lado, podem ocupar mais memória e exigir mais tempo para serem transferidas ou processadas.
- *X* é um fator adicional que atua como um delimitador para resolver empates no cálculo da prioridade. Nesse caso, a sequência mais próxima ao final de cada vídeo recebe uma prioridade maior.

Como pode ser observado na Figura 4, foram ilustradas janelas temporais de tamanho 5, considerando cada cliente assistindo ao vídeo. Observando-se o cliente mais próximo ao fim do vídeo, verifica-se que o bloco de vídeo predecessor a ele possui uma prioridade maior, com valor de $1,5$. Em comparação, na última janela temporal ilustrada, onde está presente o último cliente ativo no intervalo, o bloco correspondente apresenta uma prioridade de apenas $0,20$.

Em sua formulação original, o algoritmo busca manter armazenadas as sequências que apresentam maior densidade de demanda, medida pela concentração de clientes na janela que antecede cada sequência, e, simultaneamente, prioriza sequências menores, reduzindo a quantidade de memória necessária para seu armazenamento. Essa estratégia contribui para aumentar a quantidade de conteúdo mantida em cache e reduzir a pressão sobre os recursos disponíveis do proxy. Além disso, em situações de empate entre sequências com a mesma prioridade, o *CARTE* atribui maior prioridade às sequências localizadas mais próximas do final do vídeo. No $CARTE_{BB}$, versão simplificada do algoritmo derivada por Koch (2022) por outro lado, as prioridades passam a ser associadas

diretamente aos blocos de vídeo, considerando que todos possuem tamanho uniforme e representam um segundo de vídeo. Nessa adaptação, não há uso de sequências de blocos, os dados de densidade de clientes são trabalhados dentro de uma janela. O mecanismo de desempate foi simplificado, mantendo os princípios gerais de priorização, mas alterando parte da lógica originalmente utilizada para ordenação dos candidatos à substituição.

Figura 4 – Cálculo de prioridade de cacheamento no algoritmo CARTE




Fonte: do Autor(2026)

### 3.4 Algoritmo RTBC - Reuse Time Based Caching

Trata-se de um algoritmo projetado por Wu et al. (2012) para estimar o tempo até que ocorra o acesso a cada segmento dos vídeos cacheados na memória do proxy. O funcionamento baseia-se na medição da distância entre cada segmento e a posição de leitura do cliente ativo temporalmente mais próximo a ele. Essa análise permite identificar quais segmentos de vídeo tem maior urgência de cacheamento com base na localidade temporal entre acessos.

Quando o cliente mais próximo a um determinado segmento efetua o acesso a este segmento, o sistema automaticamente realiza uma nova análise, recalculando a distância entre o segmento acessado e o cliente mais próximo seguinte, ajustando as estimativas de tempo com base na nova configuração do sistema. Esse processo dinâmico garante que as estimativas permaneçam atualizadas, otimizando o desempenho e a eficiência no

gerenciamento de acessos ao vídeo.

Com base nos princípios básicos apresentados, Wu et al. (2012) utiliza em sua pesquisa as seguintes premissas, a respeito do tempo de reutilização do segmento, ou seja, da sua estratégia para obter as prioridades de cacheamento para cada segmento. As situações previstas são:

- Transferência imediata entre servidor, proxy e clientes: o autor assume em seu trabalho que o tempo de transmissão do conteúdo entre as três entidades é desconsiderado. Além disso, ele também assume que há cenários onde os clientes não são servidos pelo proxy, mas sim diretamente pelo servidor principal. Logo, do ponto de vista de proxy, as requisições feitas por esses clientes são tratados como cache miss.
- Continuidade ininterrupta: Supôe-se que os clientes não interrompem a visualização do filme desde que solicitem o primeiro segmento; eles assistem ao filme do início ao fim, segmento por segmento.
- Uniformidade de duração e tamanho dos filmes: Partindo da premissa de que todos os filmes têm uma duração de 5.400 segundos (equivalente a 1,5 horas) e que cada segmento é presumivelmente de tamanho igual.

O cálculo do intervalo entre cada segmento e seu cliente correspondente mais próximo chama-se Tempo Exato de Reuso (TER) para o segmento $\sigma$(sigma).

Quando não há clientes ativos, o algoritmo estima um valor apropriado para o *TER($\sigma$)*, garantindo que o sistema continue a operar de maneira eficiente mesmo em situações de baixa demanda. Essa definição também é essencial para o cálculo da popularidade do primeiro segmento de cada vídeo, uma vez que não há clientes ativos situados temporariamente em ponto anterior a estes segmentos, sendo este calculo realizado em duas etapas complementares:

- Cálculo Arbitrário: Nessa etapa, é atribuído um valor elevado aos primeiros segmentos de cada vídeo. Isso assegura que esses segmentos recebem prioridade de cacheamento maior do que qualquer outro, permitindo que o sistema obtenha uma estimativa inicial do tempo de substituição para segmentos iniciais de vídeos recém-introduzidos.
- Cálculo de Frequência: A frequência de chegada das requisições para cada vídeo é armazenada pelo algoritmo. Esse valor é utilizado como base para determinar o tempo inicial de substituição dos vídeos. No entanto, se não houver

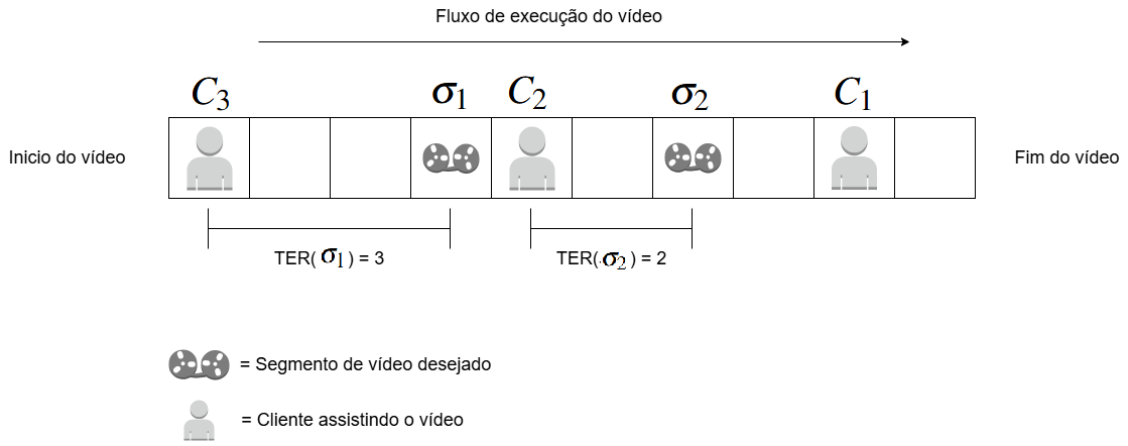
novas requisições para um vídeo dentro de um intervalo de tempo predefinido, a prioridade de substituição será ajustada de forma arbitrária, permitindo que os recursos do sistema sejam redistribuídos de maneira eficiente.

Para determinar o calculo da prioridade de cacheamento de qualquer segmento, considera-se que, no tempo inicial *zero*, foram obtidos valores de referência a partir dos cálculos citados anteriomente. Esses valores iniciais são usados para calcular o *(TER($\sigma$))* com base na frequência de requisições registradas no último intervalo de tempo.

De acordo com Wu et al. (2012), se o número de requisições for maior que zero, o valor de *TER($\sigma$)* será aplicado, como ilustrado na Figura 5. Por outro lado, caso não haja requisições para o vídeo, será realizado o cálculo de um valor arbitrário, desde que duas condições sejam atendidas: o número de requisições por clientes seja maior que zero e o segmento em questão não seja o inicial do vídeo.

Conforme é apresentado na Figura 5, o bloco de vídeo $\sigma_2$ está a uma distância de 2 blocos do seu cliente mais próximo $C_2$, sendo este, portanto, o valor da sua prioridade de cacheamento de acordo com a ótica do algoritmo RTBC.

Figura 5 – Cálculo de prioridade de cacheamento no algoritmo RTBC




Fonte: do Autor(2026)

Quando é necessário descartar segmentos para liberar espaço no cache do simulador e acomodar novo conteúdo proveniente do servidor principal, a fila de substituição é processada. Nesta ação, os segmentos com maior valor de tempo de reuso são priorizados para remoção.

Se algum dos segmentos candidatos a remoção pertencer ao vídeo atualmente em reprodução, o segmento mais próximo ao final do vídeo será selecionado para descarte, garantindo que o impacto na reprodução seja minimizado. Em situações de empate entre

segmentos com o mesmo tempo de reuso, a decisão será tomada com base na frequência de acesso por clientes, assegurando uma escolha coerente e alinhada aos padrões de uso.

## 3.5 Coeficiente Zipf e Poisson

A distribuição de Zipf foi utilizada para modelar a popularidade dos conteúdos disponíveis no sistema de vídeo sob demanda. Segundo Dan, Sitaram e Shahabuddin (1996), a demanda por conteúdos multimídia apresenta comportamento concentrado, no qual um conjunto reduzido de vídeos recebe a maior parte das requisições, enquanto os demais conteúdos são acessados com frequência inferior. Esse padrão é recorrente em sistemas de distribuição de mídia e motivou a utilização da distribuição de Zipf como mecanismo de representação da popularidade dos objetos armazenados.

Matematicamente, a distribuição estabelece que a probabilidade de acesso a um conteúdo é inversamente proporcional à sua posição em um ranking de popularidade. O parâmetro $(\alpha)$ controla o grau de concentração das requisições, permitindo representar diferentes cenários de consumo. Valores reduzidos de $(\alpha)$ produzem distribuições mais uniformes entre os conteúdos, enquanto valores mais elevados concentram a demanda em um conjunto menor de vídeos. No simulador, essa distribuição é utilizada para selecionar os conteúdos requisitados pelos clientes, permitindo reproduzir padrões de acesso observados em sistemas de vídeo sob demanda.

A geração dos instantes de chegada dos clientes foi baseada em um processo de Poisson, amplamente utilizado para representar eventos independentes distribuídos ao longo do tempo. Conforme descrito por Almeida et al. (2001), esse modelo permite representar a entrada de usuários em sistemas de distribuição de conteúdo por meio de uma taxa média de chegadas $(\lambda)$, produzindo uma sequência de eventos cuja ocorrência varia ao longo da execução, mas mantém um comportamento estatístico controlado.

A utilização do processo de Poisson permite representar cenários nos quais os usuários ingressam no sistema em instantes distintos, reproduzindo variações naturais de carga observadas em ambientes de streaming. Embora a taxa média de chegadas permaneça constante, os intervalos entre usuários consecutivos variam de forma probabilística, produzindo períodos com maior ou menor concentração de acessos. No simulador, esse mecanismo é utilizado para determinar os instantes de entrada dos clientes, servindo como base para a construção das cargas de trabalho empregadas na avaliação dos algoritmos de substituição de cache.

### 3.6 Simulador de proxy VoD

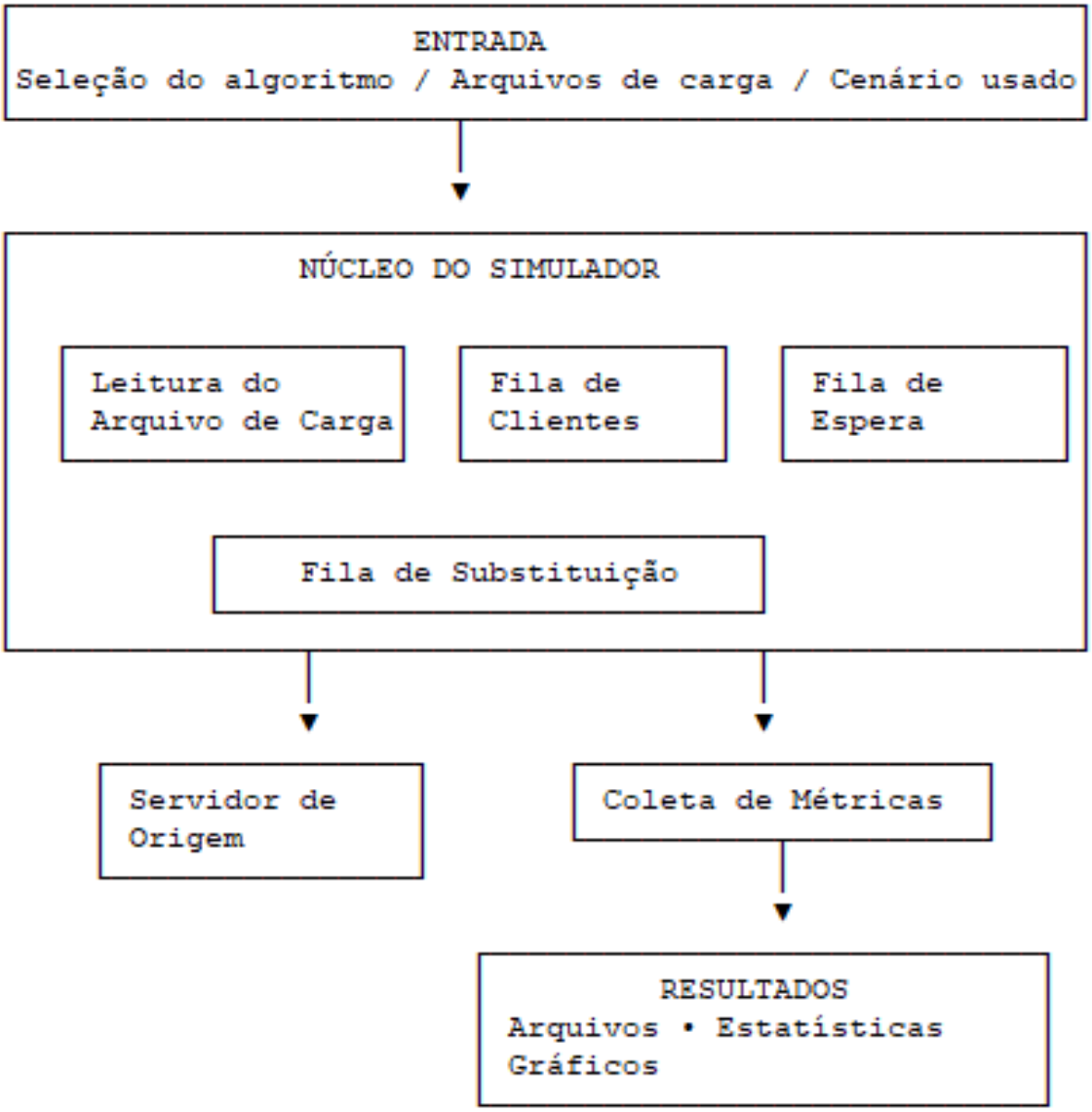
O simulador de proxy utilizado neste estudo, implementado por Koch (2022) em linguagem Python, foi desenvolvido com o objetivo de incorporar os elementos essenciais para simular um servidor de Vídeo sob Demanda (VoD). Ele é constituído pelos seguintes componentes: uma lista de clientes, uma matriz de classificação de blocos de vídeo, uma árvore binária para implementar a ordem de prioridade, espaço de armazenamento para a cache, uma fila de espera por blocos não presentes na cache, uma fila de substituição de blocos de vídeo e controle de tempo para cálculo de cada requisição concluída. A Figura 6 apresenta de forma macro, a correlação entre os módulos responsáveis pelo funcionamento do simulador.

A lista de clientes retém os usuários que aguardam atendimento até que completem suas sessões de vídeo. As informações das solicitações de vídeo de cada cliente são lidas a partir de um arquivo de texto, onde cada linha contém quatro colunas: o primeiro campo indica o ID do cliente, o segundo identifica o filme, o terceiro especifica um valor numérico usado para simular o instante de chegada da requisição (em segundos) e o quarto armazena um valor numérico usado para simular o instante de chegada de encerramento da sessão de vídeo.

A classificação dos blocos de vídeo, para fins de determinação de quais blocos tem maior prioridade para se manter na memória do proxy, é determinada pela matriz de classificação, que é atualizada a cada solicitação realizada. O tamanho da matriz varia conforme o número de blocos de vídeo mantidos na cache do proxy. Para cada bloco armazenado é atribuída uma posição específica na matriz que representa sua prioridade no cache. Quando um bloco é removido, seu valor de classificação é retirado da árvore binária. Essa árvore binária é responsável por organizar os dados da matriz de classificação, onde cada nó na árvore contém um valor de prioridade de cache e uma fila de blocos com igual prioridade, o endereço do nó à esquerda, à direita e anterior. O intervalo deste valor de prioridade é considerado um acumulador, ajustado conforme os blocos são adicionados ou retirados. Ao reordenar a matriz de classificação, o mesmo processo é aplicado na árvore binária quando os blocos reorganizados estão em memória.

Em sua memória cache, os dados são dispostos em uma matriz, onde cada linha representa a quantidade de blocos de vídeo armazenados por filme, mas os frames dos vídeos não são salvos, permitindo simulações que demandam mais memória do que a disponível no computador.

Figura 6 – Arquitetura do simulador de Koch (2022).




Fonte: do Autor (2026)

A fila de espera é destinada a pedidos de blocos não presentes no cache, que precisam ser requisitados ao servidor principal. Ao concluir a leitura da lista de clientes, a fila de espera se organiza com base em três critérios: maior número de solicitações para o mesmo filme e bloco, maior quantidade de pedidos para o mesmo filme e por ordem dos pedidos. Essa organização visa maximizar o atendimento de solicitações dentro do limite de banda disponível entre o proxy e o servidor principal.

A fila de substituição armazena dados como a localização dos blocos de vídeo na memória, identificadores de filme e bloco, sendo gerada por uma leitura in-ordem da árvore binária. Essa fila de substituição tem o mesmo tamanho da fila de espera, pois é constituída através da sequência das junções das filas de blocos, dentro de cada nó da árvore, e obedece a dois critérios de substituição: o valor na matriz de classificação e a sequência da fila de blocos com a mesma classificação.

O tempo dentro da simulação é diferente do tempo necessário para executar o simulador no computador que o hospeda. O tempo fora da simulação pode variar bastante, pois depende do algoritmo de substituição utilizado e das configurações do computador. No entanto, o tempo na simulação permanece sempre o mesmo, independentemente das limitações físicas ou lógicas da infraestrutura real de um proxy VoD.

Para a implementação do novo código correspondente ao algoritmo RTBC, foi necessária uma análise detalhada do código existente, a fim de compreender o fluxo de funcionamento e a rotina do sistema de ponta a ponta. Essa parte revelou que o simulador de proxy VoD foi desenvolvido utilizando uma arquitetura modular, tal que os algoritmos já implementados são organizados em módulos e executados por meio da importação de classes específicas. Essa abordagem modular facilita a expansão do simulador e mantém a organização e escalabilidade do código.

Com base nessa estrutura, verificou-se a viabilidade de integrar o novo algoritmo ao sistema, permitindo sua importação e execução de forma consistente com os demais componentes.

### 3.6.1 Módulo de leitura do arquivo de carga

Neste módulo, a leitura das requisições é organizada em ciclos de tempo sincronizados, que representam as unidades de tempo de processamento do simulador. Durante cada ciclo, as solicitações dos clientes são capturadas e processadas, de forma a garantir que os segmentos de vídeo sejam entregues no ritmo necessário para viabilizar a

execução contínua do fluxo de distribuição do conteúdo pelo proxy.

A estrutura detalhada desse processo, bem como as interações entre as etapas, estão representadas na Figura 7. Essa figura ilustra de forma clara como as requisições são tratadas e como o sistema lida com os diferentes cenários operacionais, assim como quando são realizadas chamadas de outras funções, destacados com uma cor diferente no diagrama.

O funcionamento do módulo envolve diversas etapas críticas. Inicialmente, as requisições dos clientes são lidas e classificadas, considerando parâmetros como a prioridade e a disponibilidade dos segmentos na cache. O objetivo é garantir que as solicitações sejam atendidas com o mínimo de atraso, evitando interrupções ou degradações na experiência de reprodução do vídeo.

Antes do término de cada ciclo, o sistema realiza verificações importantes para assegurar a consistência e eficiência do processo. Primeiramente, verifica-se se a leitura do arquivo de vídeo foi concluída pelos clientes que estão ativos. Essa etapa é crucial para evitar inconsistências ou perda de dados durante a transmissão. Em seguida, o sistema analisa a fila de espera para identificar a presença de clientes pendentes.

Se houver clientes na fila de clientes, o módulo executa uma série de ações:

- Calcula e armazena as taxas de acerto *(cache hits)* e erro *(cache misses)*, permitindo uma avaliação contínua do desempenho do sistema.
- Monitora a largura de banda utilizada, ajustando dinamicamente os recursos para otimizar o atendimento das solicitações.
- Reinicializa os contadores e métricas internas para preparar o sistema para o próximo ciclo.

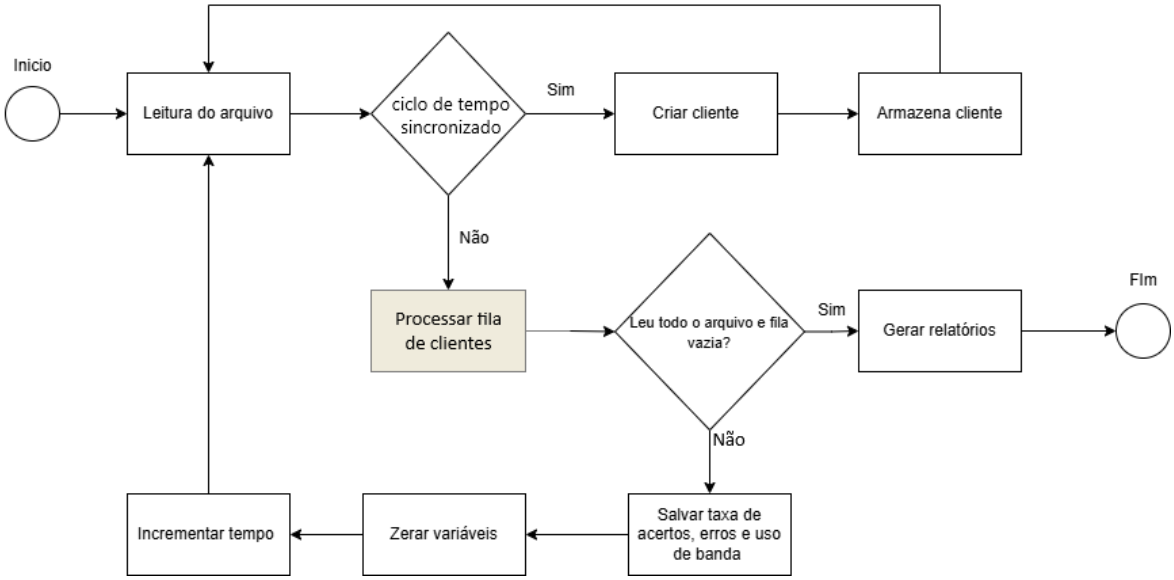
Por outro lado, se não houver clientes na fila de clientes, o simulador gera logs de desempenho ou estatísticas acumuladas e avança automaticamente para a próxima etapa do processamento.

### 3.6.2 Módulo de processamento da fila de clientes

O módulo de processamento da fila de clientes opera com base em etapas bem definidas para o atendimento das solicitações.

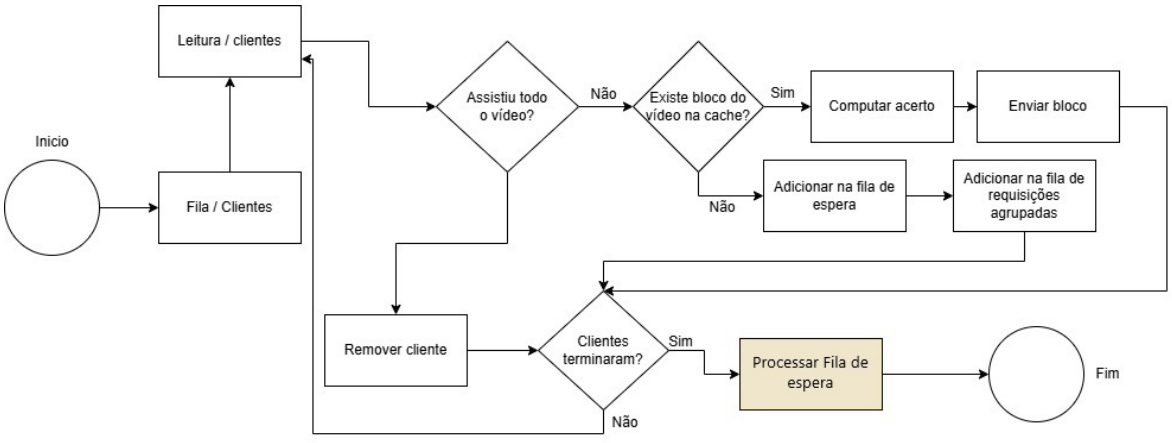
Primeiramente, o sistema verifica se o cliente na frente da fila já concluiu a exibição do filme solicitado. Se o cliente tiver assistido ao filme por completo, ele é

Figura 7 – Leitura de um arquivo de carga.




Fonte: do Autor (2026)

Figura 8 – Fila de Clientes.




Fonte: do Autor (2026)

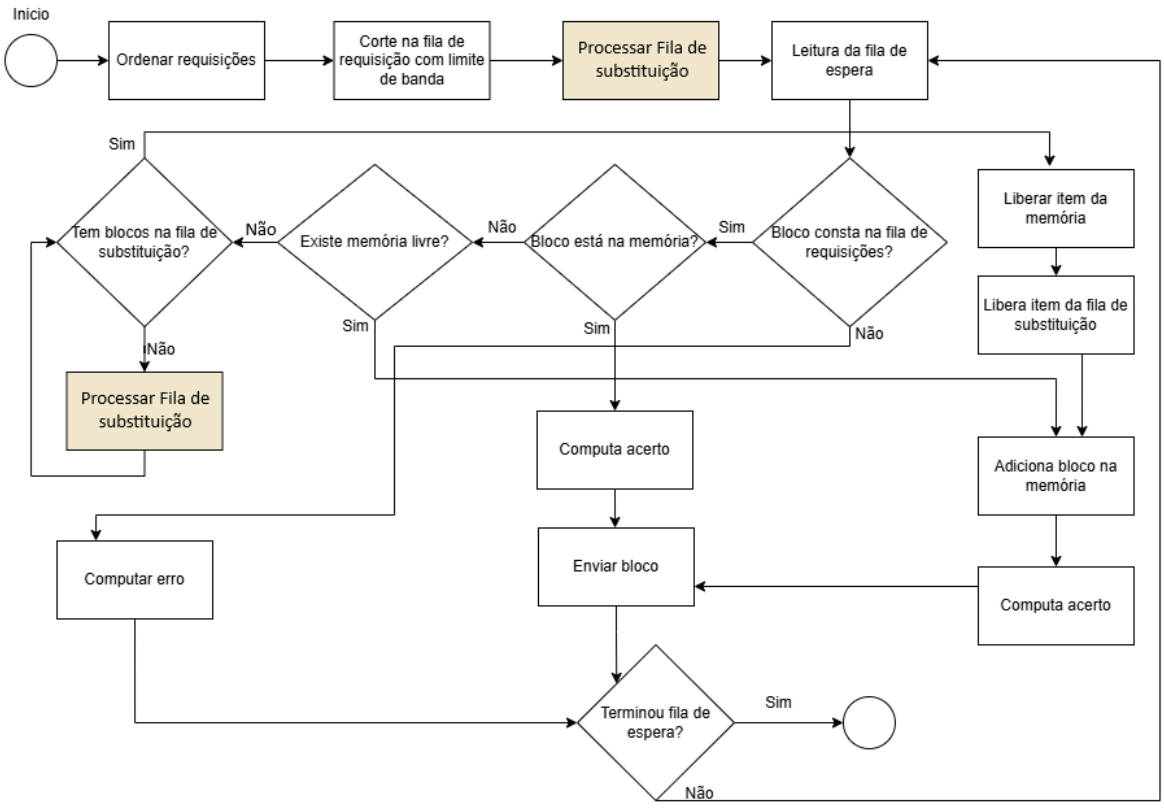
removido da fila e direcionado para a próxima etapa do fluxo de processamento, que é realizar a sua remoção da fila de espera.

Caso o cliente ainda não tenha finalizado o filme, o sistema realiza uma busca na cache para verificar a disponibilidade do bloco de vídeo solicitado. Se o bloco estiver presente na cache, ele é prontamente enviado ao cliente, garantindo a continuidade da reprodução sem interrupções e contribuindo assim para o incremento da taxa de acertos do proxy.

No entanto, se o bloco não estiver disponível na cache, o cliente é transferido para uma nova fila de espera. Nessa fila, ele aguardará até que o segmento solicitado esteja disponível para ser enviado. Esse processo visa otimizar a utilização dos recursos de cache e largura de banda, reduzindo o impacto da ausência de blocos na experiência do usuário.

A estrutura descrita, incluindo o fluxo entre as filas e os processos associados, está ilustrada na Figura 8, que oferece uma visão clara do funcionamento do módulo.

Figura 9 – Fila de Espera.




Fonte: do Autor (2026)

### 3.6.3 Módulo de processamento da fila de espera

O processamento da fila de espera abrange todos os clientes que não conseguiram acessar seus respectivos segmentos previamente armazenados na cache. Esses clientes, ao não encontrarem os dados necessários, precisam solicitar recursos diretamente ao servidor principal, o que gera um aumento na demanda por largura de banda e na latência de acesso.

Para otimizar esse processo, os clientes na fila de espera são primeiramente agrupados de acordo com o volume de requisições para cada segmento específico. Esse agrupamento prioriza os segmentos mais requisitados, garantindo que eles sejam atendidos primeiro. Essa abordagem é essencial para maximizar a eficiência do sistema, já que o simulador proxy opera dentro de limites predefinidos de largura de banda.

Após a priorização, o próximo passo é armazenar os segmentos na cache do proxy. Esse armazenamento é realizado com o objetivo de atender rapidamente os clientes da fila de espera e reduzir futuras solicitações ao servidor principal.

No entanto, quando a capacidade de armazenamento da cache é ultrapassada, o algoritmo de caching selecionado na simulação entrará em ação de forma a verificar quais segmentos serão excluídos, a fim de liberar espaço para armazenamento dos novos segmentos de vídeo oriundos do servidor principal. Esta ação está implementada na estrutura da fila de substituição. Cada passo de todo este processo, está apresentado na Figura 9.

### 3.6.4 Módulo de processamento da fila de substituição

A execução da fila de substituição utiliza uma estrutura de dados conhecida como árvore binária, que precisa estar previamente balanceada para garantir um desempenho eficiente.

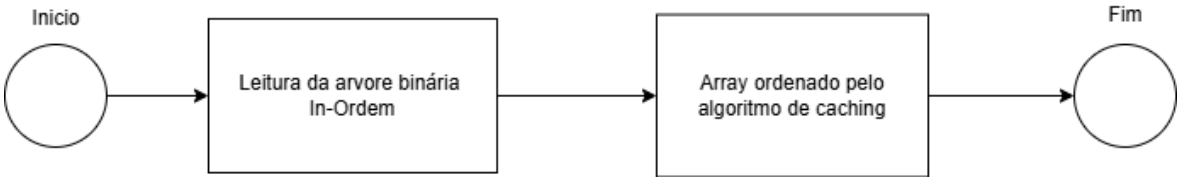
A cada requisição realizada pelos clientes, a pontuação dos nós da árvore é recalculada. Essa pontuação é ajustada com base no algoritmo de substituição selecionado. O balanceamento contínuo da árvore, aliado à atualização dinâmica da pontuação, assegura que o processo de substituição seja realizado de maneira ágil, isto é, rápida e precisa, que busca maximizar a eficiência no gerenciamento da cache.

Neste trabalho, este é o ponto de integração do algoritmo RTBC proposto por Wu et al. (2012), ao processo geral de simulação, sendo uma das funcionalidades centrais na

operação do simulador. Suas especificações, que incluem os princípios de funcionamento, etapas de implementação e objetivos esperados, estão detalhadas na Seção 3.4, oferecendo uma visão abrangente do seu comportamento.

Ainda ao nível de estrutura do simulador, a Figura 10 ilustra como as requisições são recebidas por este processo, fundamental para garantir a consistência das operações, o que permite ao simulador gerenciar de maneira ordenada os fluxos de dados e tomar decisões baseadas nas informações processadas. Em tempo, a figura apresenta de forma genérica a recepção de uma requisição de processamento desta fila de substituição, onde cada algoritmo de caching possuem rotinas de execução, as quais serão detalhadas no capítulo à seguir.

Figura 10 – Fila de Substituição.




Fonte: do Autor (2026)

# 4 TRABALHOS RELACIONADOS

Nesta seção são apresentados os algoritmos *CC* e *CCVC*, ambos pertencentes à categoria de algoritmos do tipo *Look Ahead*. O algoritmo *CC* já integra o simulador utilizado neste trabalho e serve como uma das referências para a análise das políticas de substituição de cache. O algoritmo *CCVC*, por sua vez, é introduzido como uma variação dessa abordagem, empregando mecanismos adicionais para a definição das prioridades de armazenamento dos segmentos.

Embora os resultados apresentados na literatura indiquem desempenho inferior ao observado em algoritmos como *RTBC* e *CARTE*, essas abordagens permanecem relevantes para este estudo por representarem estratégias distintas de gerenciamento de cache e por contribuírem para a compreensão da evolução das políticas de substituição aplicadas a sistemas de vídeo sob demanda.

### 4.0.1 Algoritmo CC (chunk-based cache)

Conforme pode ser verificado no artigo publicado por Hong, Vleeschauwer e Baccelli (2010), este algoritmo baseia-se na manutenção de uma pontuação $S_k$, onde o alfabeto de $k$ é determinado sequencialmente em $k$ $(k = 1,2,...,K)$. Quando um vídeo $k$ é solicitado, sua pontuação é aumentada em 1, enquanto a pontuação de todos os outros vídeos é diminuída em 1, considerando que sua pontuação inicial possui valor $A$. Quando um novo vídeo é solicitado pela primeira vez, sua pontuação é inicializada com um valor $B$. Além disso, a pontuação é mantida dentro do intervalo $[-C,C]$, sendo truncada a esses limites. A cada momento de solicitação, os primeiros $L$ vídeos mais bem ranqueados com base nessa pontuação $S_k$, são elegidos para serem armazenados em cache. Assim, nesses instantes de solicitação, um dos três eventos a seguir pode ocorrer:

- O vídeo requisitado já encontra-se em cache, o que é conhecido como um "*hit*". Nessa situação, o algoritmo atualiza a pontuação, porém nenhum vídeo é retirado do cache. O vídeo é disponibilizado diretamente do cache, eliminando a necessidade de tráfego na rede de distribuição.
- O vídeo requisitado não estava armazenado em cache e o algoritmo decide armazená-lo, pois a solicitação eleva o vídeo para uma posição entre as primeiras $L$ posições. O servidor transfere o vídeo para o cache, resultando em um

fluxo de informações na taxa do vídeo, gerando tráfego na rede de distribuição. Simultaneamente, o vídeo que originalmente ocupava a última posição antes da atual solicitação é gradualmente removido do cache, sendo substituído pelo novo vídeo. À medida que as informações do novo vídeo fluem e são copiadas para o cache, ocorre a sobreposição do vídeo nesta posição, onde também são encaminhados os dados para o usuário que fez a solicitação.

- O vídeo requisitado não está em cache. O vídeo é então entregue diretamente do servidor de origem, resultando no consumo de largura de banda na rede de distribuição.

Complementarmente, o algoritmo opera cada vídeo de forma a dividi-los em $m$ segmentos, e cada segmento herda a pontuação $S_k$ do vídeo ao qual pertence. Para cada segmento m do vídeo $k$, é mantido um valor $N_k,m$, que acumula o número de acertos garantidos que esse segmento terá, com base nas informações sobre quais vídeos estão sendo assistidos pelos usuários,assumindo que nenhum usuário interrompe a visualização de um vídeo. Um contador do valor $N_k,m$ é mantido da seguinte forma:

- Os valores $N_k,m$ são incrementados em 1 para todos os segmentos $m$ sempre que o vídeo $k$ é solicitado por um cliente.
- O valor de $N_k,m$ é decrementado em 1 após um cliente, que está assistindo ao vídeo $k$, consumir o segmento $m$.
- No caso de um usuário interromper a visualização de um vídeo, para cada valor de $m$ maior ou igual ao segmento atual que o usuário estava assistindo antes da interrupção, o valor $N_k,m$ precisa ser reduzido em 1. Contudo, esta ação não é contemplada neste algoritmo.

### 4.0.2 Algoritmo CCVC - *Collapsed Cooperative Video Caching*

Ishikawa e Amorim (2009) apresentam a implementação de um método de gerenciamento de cache de memória, denominado *Collapsed Cooperative Video Caching* (CCVC), para um servidor proxy em mídia contínua, sendo para vídeos ou áudios, tal método é separado em duas partes: A primeira gerencia o cache de um único servidor proxy local, enquanto a segunda aponta para o compartilhamento de conteúdos de mídia que estão armazenados em diferentes servidores distribuídos, promovendo a cooperação entre servidores proxy. Em outras palavras este método é capaz de funcionar de modo

híbrido seja em um servidor local ou de modo distribuído.

Para esta administração são alocados espaços reservados na memória cache afim de armazenar segmentos dos arquivos de mídia contínua. Essa porção de memória reservada na cache atua como um buffer colapsado, onde os segmentos de um arquivo de mídia são ininterruptamente eliminados para criar espaço para os segmentos subsequentes do mesmo arquivo. Deste modo, para cada cliente do sistema, é destinado um buffer colapsado na cache, e a área de memória desse buffer pode ser compartilhada com outros buffers, desde que o conteúdo seja idêntico.

Outro elemento crucial na estratégia adotada por esses autores é conferir uma prioridade para armazenamento em cache mais elevada a blocos situados em sequências de vídeo. Como vantagem principal proporcionada por essa tática, os blocos de vídeo transferidos para a memória para atender aos clientes mais antigos do sistema tendem a permanecer na memória para serem acessados pelos clientes mais recentes que assistem ao mesmo vídeo. Essa particularidade destaca o CCVC como um dos pioneiros algoritmos a considerar os clientes ativos que ainda não acessaram determinadas partes de um vídeo como um fator para determinar quais partes dos vídeos devem ter prioridade no armazenamento em cache no proxy (NEVES, 2015).

## 4.1 Workload sobre Video-on-Demand (VoD)

Ali-Eldin et al. (2015) definem que VoD é um serviço de compartilhamento de vídeos que abrange uma grande porcentagem de trafégo de downstream internet, sendo assim seu trabalho buscou analisar a carga destes serviços de VoD fornecidas por uma emissora de TV Suéca, com mais de 20.000 usuários exclusivos. Sua pesquisa apontou as taxas de chegada e solicitações, analisando o tempo de chegada, pico de carga de trabalho, e a distribuição de popularidade dos vídeos. Foi possivel constatar com base neste estudo o comportamento de usuários impacientes, abandonam as sessões de streaming logo após alguns minutos ou até mesmo segundos depois de iniciá-las.

Summers et al. (2016) analisou a plataforma Netflix, que é um serviço de assinatura para streaming de vídeo amplamente usado (cerca de 81 milhões de assinantes, em todo o mundo). Tal volume de clientes permite verificar a robustez deste serviço, operando sobre altas cargas de trabalho. Ao observar os registros de sessões individuais, nota-se que os clientes apresentam padrões distintos de solicitações. A prática de emitir solicitações para vários arquivos de vídeo em um breve intervalo de tempo é recorrente

em muitas sessões que foram examinadas. Também observou-se padrões nos quais os clientes acessam o conteúdo de forma sequencial a partir de uma única taxa de bits ou simplesmente não acessam nenhum arquivo de vídeo, indicando que a reprodução está pausada. Verificou-se que as sessões de execução são muito estáveis, onde cerca de 5% do tempo total é gasto durante as transições de um vídeo para o outro, 79% em fases estáveis e 16% em fases inativas, com duração média de 8,5 minutos. O autor também analisou dois algoritmos de pré-busca aplicados ao atendimento de uma carga de trabalho de streaming de vídeo HTTP. O primeiro algoritmo opera sem quaisquer informações prévias sobre a carga de trabalho, enquanto o segundo explora as características da carga de trabalho de duas formas. Primeiramente, ele seleciona um entre três tamanhos pré-definidos na cadeia de pré-busca para a primeira requisição, considerando o tipo da requisição e o deslocamento inicial no arquivo de vídeo. Com base nessas informações, o autor apresentou melhorias significativas, como uma redução de 13% na utilização do disco rígido e 30% na utilização de memória.

Segundo Summers et al. (2012), o aumento de tráfego de streaming também coincide com o uso constante do protocolo HTTP no tráfego de vídeo. Provedores como Apple, Microsoft, Akamai, utilizam este protocolo para transmitir conteúdo. Desta forma, o autor considera que entender o impacto desta carga de trabalho em servidores de vídeo torna-se de extrema importância. As dificuldades enfrentadas ao tentar empregar estudos de caracterização de carga de trabalho existentes, enfatiza a demanda por cargas de trabalho específicas, necessitando introduzir benchmarks exemplares, através de uma série de dados coletados pelo autor, desde configurações físicas dos servidores de vídeo e seu sistema operacional, a até características de comportamento dos usuários, como duração das sessões, se há repetições destas sessões de mesmo vídeo e o *throughput* de *web servers*. Também é relatado pelo autor que, não é possível determinar com precisão a quantidade de dados baixados apenas com base na duração da sessão, pois os clientes armazenam dados em buffer para compensar variações nas velocidades de download.

Foram utilizados estes benchmarks para avaliar o desempenho de três servidores web pré-existentes (*Apache, nginx e userver*). Verificou-se que modificações simples no *userver* proporcionaram benefícios promissores e notáveis em algumas cargas de trabalho representativas de streaming. Embora esses resultados necessitem de uma investigação mais aprofundada, eles evidenciam a importância e o valor dos benchmarks de transmissão de vídeo via HTTP no desenvolvimento de servidores web.

Já para García et al. (2007), o serviço de VoD conta com anos de atuação,

registrando em um banco de dados mais de 160.000 reproduções e 900 vídeos, com ampla diversidade de temas, conteúdo extenso e atualizações frequentes. Pesquisas anteriores eram limitadas a usuário, tópicos ou ambientes específicos, o que neste estudo, foi tratado de forma distinta. Por meio de uma análise estatística de comportamento, verificou-se o comportamento de usuários de streaming em termos de características de sessão, reprodução, mídia entregue, número e duração de pausas, alteração de fluxo de execução, além do índice de popularidade e seu perfil de acesso diário. Isso resultou na possibilidade de desenvolver um modelo de simulação capaz de gerar carga de trabalho compatível com diferentes cenários e situações de serviço.

A partir dos estudos realizados sobre os materiais bibliográficos relacionados, alguns cenários poderão ser explorados. Como pode ser visto em Kim et al. (2017), uma estimativa de tempo de vida, *Time To Live (*TTL), pode ser definida para um segmento de vídeo. O servidor principal determina se um segmento de vídeo é uma cena de destaque, ou seja, possui alta popularidade, com base nas solicitações dos clientes e nas interações entre o servidor de origem e os servidores de proxy. Quando o segmento é identificado como destaque, o servidor de origem aloca um *TTL* adaptativo ao servidor de proxy. Baseando-se em ações dos espectadores na sala de chat, como taxas de visualização, quantidade de mensagens, cliques em "Curtir"ou "Balão de Estrela", e pedidos de replay, se o segmento for classificado como destaque, recebe um TTL maior, aumentando a taxa de acertos no cache e melhorando a qualidade do serviço para os clientes, ao mesmo tempo que reduz a carga no servidor de origem. Se não for uma cena de destaque, o TTL será curto, otimizando o uso do espaço de armazenamento no servidor de cache.

Existem ainda outras possibilidade que podem ser exploradas, conforme Chen et al. (2006) *apud* Claeys et al. (2016), um sistema de proxy de streaming chamado *Hyper Proxy*, que utiliza pré-busca ativa para entregar segmentos não cacheados a tempo, abordando o problema de *jitter*[1] do proxy. Esse trabalho foi expandido com a adição de estratégias de segmentação baseadas na popularidade dos objetos e pelo uso de previsões de solicitações futuras de segmentos para pré-carregar segmentos não cacheados, otimizando o posicionamento do conteúdo com base na popularidade do conteúdo e na distribuição geográfica das solicitações.

O algoritmo de Hong, Vleeschauwer e Baccelli (2010) também classifica os blocos e armazena em cache os que possuem maior prioridade, mas adota um critério de classificação diferente. A classificação é principalmente determinada pela previsão

[1]Jitter: medida da variação no atraso entre a transmissão e a recepção de um segmento em uma rede de comunicação

Tabela 3 – Tabela de trabalhos correlacionados

| **Autores** | **Pontos Importantes** | **Resultados** |
|---|---|---|
| Ali-Eldin et al. (2015) | A análise de rastreamentos de VoD de um provedor de serviços sueco tem como objetivo fornecer insights para melhor entender e projetar sistemas VoD. | Pode-se concluir que a taxa de usuários que abandonam sessões de streaming poucos minutos após iniciá-las parece ser uma constante em cargas de trabalho de VoD. |
| Summers et al. (2016) | Neste artigo, analisa-se um arquivo de log de servidor web anonimizado para caracterizar a carga de trabalho de um servidor de vídeo por streaming da Netflix. O arquivo de log, contudo, não foi disponibilizado publicamente. | As solicitações foram organizadas e caracterizadas em cadeias e fases. A utilidade dessa caracterização da carga de trabalho foi demonstrada por meio do desenvolvimento e análise de um algoritmo de pré-busca, que utiliza características específicas da carga de trabalho para reduzir a utilização do disco rígido e o consumo de memória do sistema. |
| Summers et al. (2012) | Neste artigo, são desenvolvidas ferramentas e metodologias para gerar cargas de trabalho e benchmarks para sistemas de transmissão de vídeo. | Os resultados exigem investigação adicional, eles demonstram a necessidade e o valor de benchmarks de transmissão de vídeo via HTTP no desenvolvimento de servidores web. |
| García et al. (2007) | Apresenta um estudo estatístico do comportamento do usuário e do tráfego de streaming, analisando características de sessão, reproduções incorretas, quantidade de mídia entregue, número e duração de pausas e avanços na reprodução, popularidade e perfil de acesso diário. | Os resultados da análise permite desenvolver modelos de simulação e geradores de carga de trabalho para avaliar diferentes cenários e situações do serviço. |
| Kim et al. (2017) | Apresenta o uso de TTL como estimativa de vida de um segmento de vídeo | Permite uma avaliação dos segmentos de vídeo em um cenário de grande requisição, principalmente em transmissões ao vivo. |
| Claeys et al. (2016) | Cita Chen et al. (2006) e a proposta do Hyper Proxy, utilizando parametros de rede como jitter da rede e localização geográfica. | Utiliza estratégias baseadas na popularidade, tentando prever solicitações futuras. |
| Hong, Vleeschauwer e Baccelli (2010) | Possui abordagem semelhante ao RTBC | Recorre ao histórico do padrão de consumo, em caso de empates na substituição de segmentos. |

Fonte: Do Autor (2026)

de quantas vezes os blocos serão acessados no futuro próximo. Esse valor é calculado observando quais blocos de quais vídeos estão sendo assistidos no momento, juntamente com o conhecimento de que, quando um determinado bloco de vídeo está sendo consumido, todos os blocos subsequentes desse vídeo também serão acessados em breve, supondo que o cliente não interrompa a reprodução do vídeo. Para resolver empates entre blocos com a mesma previsão de acesso, o algoritmo recorre ao padrão histórico de consumo.

Com base nos dados apresentados neste capítulo, foram resumidos os objetivos e propósitos que justificam a realização deste trabalho, a partir de artigos semelhantes que servem como base comparativa para os objetivos propostos nesta pesquisa. Dessa forma, a Tabela 3 visa reforçar a relevância e a necessidade do tema, onde foi possível identificar pontos em comum, como a dificuldade do servidor em compreender o comportamento dos usuários, independentemente do algoritmo aplicado. Isso resulta na maioria das vezes em um ambiente de baixo desempenho, algo que pode ser aprimorado com modelos híbridos que integrem as melhores características dos algoritmos abordados neste trabalho.

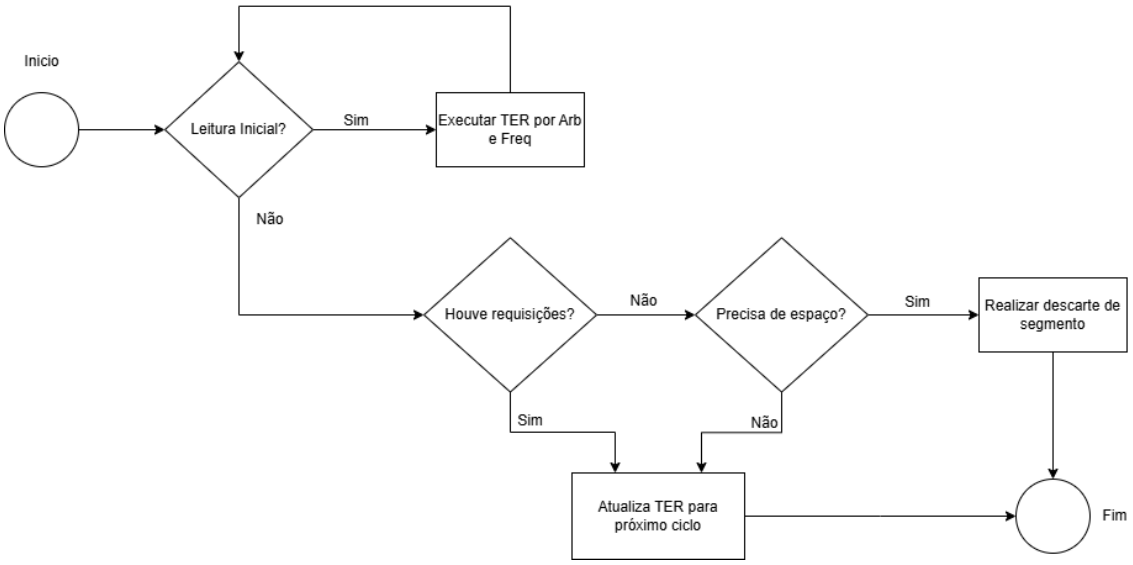
# 5 DESENVOLVIMENTO DESTE TRABALHO

## 5.1 Análise do algoritmo RTBC

O algoritmo RTBC *(Real-Time Based Cache)* foi concebido por Wu et al. (2012) para estimar e gerenciar o acesso a segmentos de vídeo, desempenhando uma função central na estrutura de servidores de vídeo. Ele opera com base no conceito de Tempo Exato de Reuso *(TER)*, que é definido como a distância entre cada segmento e o cliente mais próximo, considerando sua posição de leitura. Quando o cliente mais próximo acessa determinado segmento, o sistema atualiza dinamicamente essa métrica, recalculando a distância em relação ao próximo cliente mais próximo.

Em cenários nos quais não há clientes ativos, o algoritmo utiliza valores arbitrários para garantir a continuidade das operações do simulador. Essa abordagem, abordada pelo autor como Tempo Previsto de Reuso (Predicted Reuse Time - PRT) é essencial, pois permite calcular a popularidade inicial de segmentos, mesmo em situações de baixa interação. O cálculo da prioridade de cacheamento, por sua vez, é baseado na frequência de requisições observadas em intervalos anteriores. Nos casos em que essas requisições são inexistentes, são atribuídos valores arbitrários elevado ao tempo de reuso, que prioriza a remoção desses segmentos que não correspondem ao início do vídeo. No gerenciamento do cache, quando é necessário liberar espaço para armazenar novos

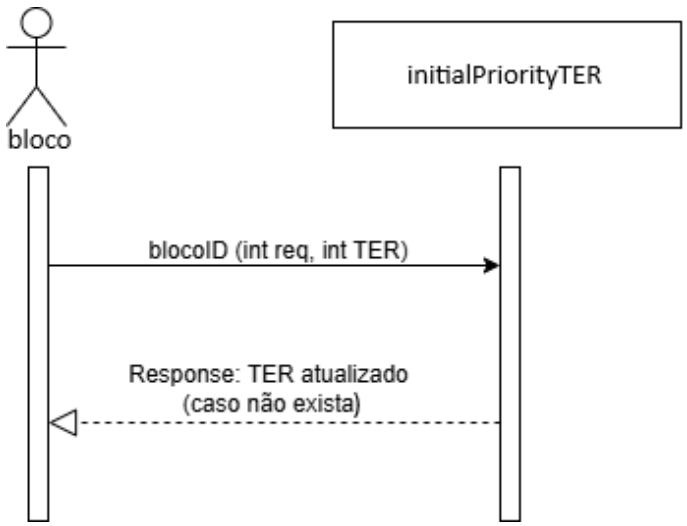
Figura 11 – Fluxo de execução do algoritmo RTBC por bloco




Fonte: do Autor(2026)

conteúdos provenientes do servidor principal, a fila de substituição do simulador processa os segmentos a serem descartados. Nesse contexto, os segmentos com maior valor de tempo de reuso são priorizados para remoção. Se o segmento em questão pertencer ao vídeo em reprodução, a política de descarte seleciona o segmento mais próximo ao final do vídeo, minimizando o impacto na experiência do usuário. Adicionalmente, em situações de empate entre segmentos com tempos de reuso idênticos, a decisão de substituição considera a frequência de acessos por clientes, de forma a manter a eficiência do sistema. Este cenário de operação é semelhante à estrutura de simulação construída por Koch (2022), o que foi uma das motivações para o desenvolvimento deste trabalho. Desta forma, um fluxograma de execução de um arquivo de carga baseado no RTBC foi construído, de forma a facilitar o entendimento de sua execução, assim como servir como modelo para a construção de seu módulo de código. A Figura 11 demonstra o tratamento de cálculo para cada bloco, seguindo as premissas vistas na Seção 3.4, onde é apresentado o fluxo de execução acionado pela função responsável por gerir a fila de substituição do simulador. Ao ser invocada, é realizada uma leitura inicial para verificar a existência de uma matriz de classificação. Caso a matriz exista, o primeiro valor de *TER* será calculado seguindo as proposições de arbitrariedade e/ou frequência, conforme descrito na Seção 3.4. Se não for a primeira execução, verifica-se se há alguma requisição de vídeo.

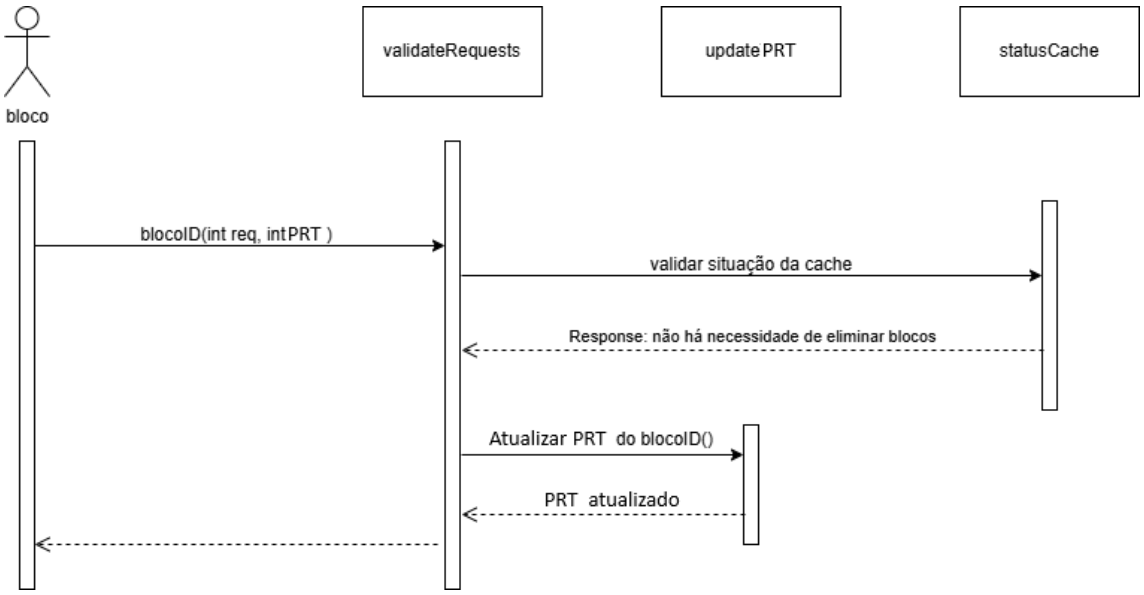
Figura 12 – Diagrama de sequência para um novo bloco de vídeo




Fonte: do Autor(2026)

Caso exista, seu *TER* é atualizado com base na frequência e contabilizado para o próximo ciclo. Caso contrário, é realizada uma nova verificação para avaliar a necessidade de liberação de espaço. Se necessário, é selecionado para remoção o segmento que

Figura 13 – Diagrama de sequência para um bloco de vídeo sem requisições de usuários




Fonte: do Autor(2026)

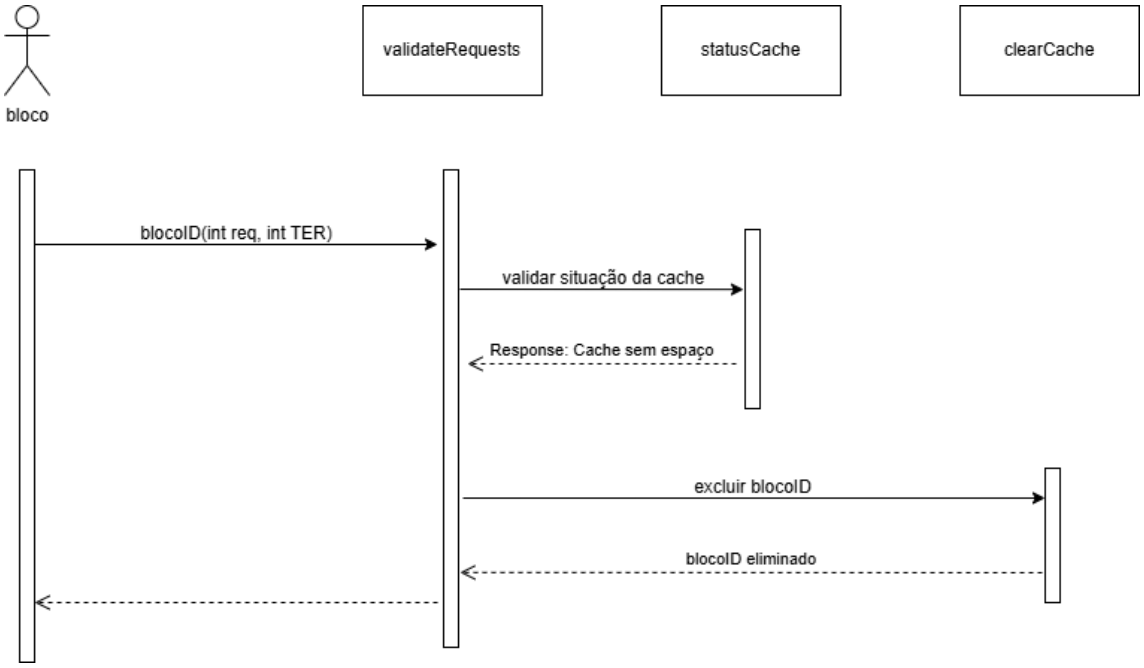
apresenta o maior tempo de reuso. Do contrário, são atualizados os valores de *TER*, e em sequência, inserido no cache.

Para melhor compreensão dos possíveis fluxos de execução do algoritmo, foram elaborados diagramas de sequência. A Figura 12 ilustra a interação entre os componentes do algoritmo no fluxo de validação de um novo bloco de vídeo que ainda não possui priorização, representada pela variável *TER*. O valor inicial de prioridade é determinado com base nas condições do algoritmo, podendo ser definido de forma arbitrária ou com base na frequência de requisições. Para o caso em que o bloco de vídeo já existe, o diagrama da Figura 12 ilustra o fluxo correspondente.

Inicialmente, é realizada a verificação das informações de reuso, conforme a Figura 12. Se o bloco já possui um valor de *TER* válido, o algoritmo verifica a existência de requisições de clientes, sendo este um cenário positivo. Nesse contexto, o tempo de reuso é recalculado com base na distância temporal entre o segmento e o espectador ativo mais próximo, atualizando os valores utilizados pelo mecanismo de substituição.

O diagrama apresentado na Figura 13 ilustra o fluxo de execução do algoritmo para o cenário em que um bloco de vídeo existente não possui requisições de usuários. Do mesmo modo que ao anterior, a validação da prioridade será realizada, assim como a validação de requisições de usuários. Nesse caso, como não existem requisições, a resposta confirma a ausência de solicitações. Em seguida, são atribuidos valores arbitrários conforme as regras definidas para o cálculo do tempo previsto de reuso,

Figura 14 – Diagrama de sequência para excluir um bloco de vídeo com menor prioridade




Fonte: do Autor(2026)

atualizando o estado do cache para determinar a necessidade de futuras substituições.

Por fim, o diagrama apresentado na Figura 14 descreve o fluxo do algoritmo para um cenário em que um bloco de vídeo existente não possui requisições de usuários e o cache está sem espaço disponível. A função de validação de prioridade é executada e, no caso de ausência de requisições, a resposta reflete esta condição. Assim, verifica-se a necessidade de liberar espaço no cache. Diante da confirmação de que o cache está cheio, o algoritmo procede com a limpeza, realizando a exclusão do bloco correspondente.

Com os cenários de execução modelados, espera-se que sua implementação no simulador seja realizada de acordo com as premissas estabelecidas, garantindo que o comportamento do algoritmo *RTBC* seja fielmente reproduzido e avaliado em condições realistas. O arquivo de carga utilizado nas simulações reúne os parâmetros responsáveis pela definição da carga de trabalho aplicada ao simulador. Esses valores são estabelecidos antes da execução de cada experimento e controlam o comportamento das requisições durante a simulação. Entre os parâmetros disponíveis destacam-se o número total de requisições por segmento, o intervalo médio entre chegadas de clientes, modelado pela distribuição de *Poisson*, a quantidade de vídeos disponível no acervo e a distribuição de popularidade dos conteúdos, representada pela distribuição de $Zipf$.

## 5.2 Integrações ao simulador

### 5.2.1 Modelagem do Algoritmo RTBC

O algoritmo RTBC (Reuse Time Based Cache), proposto por Wu et al. (2012), implementa uma política de substituição de blocos para sistemas de streaming de vídeo baseada no tempo de reutilização dos segmentos armazenados em cache. O modelo calcula tanto a reutilização de blocos por clientes ativos, quanto a frequência histórica de acesso, utilizando essas informações para determinar a permanência dos blocos em memória.

Esse algoritmo foi modelado por um conjunto de funções, responsáveis pela manutenção das estruturas de dados, cálculo dos tempos de reutilização, monitoramento dos acessos aos vídeos e gerenciamento da política de substituição. Essas funções atuam de forma integrada para registrar eventos de requisição, atualizar informações de frequência, calcular as prioridades dos blocos armazenados em cache, manter a estrutura de ordenação para à remoção de candidatos, além de executar os procedimentos de inserção, atualização e substituição de segmentos em memória. Nos parágrafos seguintes, cada função é descrita individualmente, detalhando sua participação no fluxo de execução do RTBC.

Para estimar a reutilização futura dos conteúdos, o algoritmo mantém um histórico das requisições realizadas ao primeiro segmento de cada vídeo dentro de uma janela temporal de observação. Esse comportamento é implementado pelas funções apresentadas na Figura 15, responsável pela remoção dos registros fora da janela considerada, e Figura 16, utilizada para contabilizar a quantidade de requisições observadas para cada vídeo. A partir desse histórico é obtido o parâmetro $(H_i)$, correspondente ao número de acessos registrados durante o período analisado.

Figura 15 – Função Contadora 1

```
    def prune_request_log(self) -> None:
        t0 = self._instante_tempo - self.window_w  ## janela observada
        self._req_primeiro_segmento = [
            (t, fid) for t, fid in self._req_primeiro_segmento if t >= t0
        ]
```

Fonte: do Autor(2026)

Com base nessas informações, é calculado o tempo de reutilização $(R(i,k))$ de

Figura 16 – Função Contadora 2

```
    def count_hi(self, id_filme: int) -> int:
        self._prune_request_log()
        return sum(1 for _t, fid in self._req_primeiro_segmento if fid == id_filme)
```

Fonte: do Autor(2026)

cada segmento armazenado em cache por meio da função 17. Quando existem clientes ativos que ainda consumirão o bloco $(k)$, o algoritmo utiliza o Tempo Exato de Reuso *(TER)*, determinado pela distância entre o segmento e o cliente mais próximo que o requisitará. Caso não seja possível identificar um consumidor futuro para o segmento, é aplicado o Tempo Previsto de Reuso *(Predictible Reuse Time (PRT))*, calculado a partir da frequência histórica de acessos ao vídeo. Para conteúdos sem registros de acesso na janela observada, é atribuído um valor elevado de reutilização.

Figura 17 – Exact Time for Reuse and Predictible Reuse Time Algorithm

```
    def _reuse_time_R(self, lista_filme, k: int) -> float:
        """
        Retorna R(i,k) em unidades compativeis com comparacao por magnitude.
        TER quando existe cliente com idBloco < k; sen o PRT.
        """
        clientes = list(lista_filme.clientes.values())
        precisam_k = [c for c in clientes if c.idBloco < k]
        if precisam_k:
            mx = max(c.idBloco for c in precisam_k)
            return float(k - mx)
        hi = self._count_hi(lista_filme.idFilme)
        if hi <= 0:
            r1 = float(self._RT_LN_LARGE) # Constante de Potencia elevada (10^12)
        else:
            r1 = float(self.window_w) / float(hi)
        return r1 + float(k)
```

Fonte: do Autor(2026)

Após o cálculo dos tempos de reutilização, cada bloco recebe uma chave de classificação construída pela função 18. Essa classificação utiliza informações responsáveis pela manutenção da ordenação dos filmes segundo seu histórico de acesso. A composição da chave segue os critérios definidos pelo RTBC, considerando inicialmente o tempo de reutilização do bloco, seguido pela quantidade de espectadores ativos do vídeo associado, pela posição do filme na lista de acesso mais recente e pela posição do segmento dentro do vídeo.

Quando há necessidade de liberar espaço em memória, o algoritmo executa o procedimento de substituição de blocos por meio da função `substituicaoBlocos()`. Antes da remoção de um segmento, as prioridades armazenadas são atualizadas pelas funções `reclassificar_cache_completa()` e `calcularClasificacao()`, garantindo a consistência da estrutura de ordenação utilizada pela política de substituição. Em seguida, o bloco selecionado é removido do cache e o novo segmento é inserido na posição liberada, recebendo uma classificação compatível com os critérios definidos pelo RTBC.

Figura 18 – Classificação da ordenação dos filmes/histórico de acesso

```
    def bst_key(self, lista_filme, k: int) -> int:

        r = self._reuse_time_R(lista_filme, k)
        vi = len(lista_filme.clientes)
        lru = self._lru_rank(lista_filme.idFilme)

        return (
          -int(round(r * 1e6)) * self._SCALE_R # Constante de Peso sobre o tempo de Reuso
          + vi * self._SCALE_VI # Constante de Peso sobre numero de clientes
          - lru * self._SCALE_LRU #  Constante de Criterio de desempate, WU 2012 (10^3)
          - k
        )
```

Fonte: do Autor(2026)

## 5.3 Proposta Híbrida - Algoritmo RTBC-CARTE

O algoritmo híbrido RTBC-CARTE surgiu a partir da integração de mecanismos presentes nos algoritmos RTBC e CARTE, combinando informações de reutilização temporal, frequência histórica de acessos e popularidade local dos blocos armazenados em cache. Essa proposta busca explorar características complementares de ambas as abordagens, utilizando simultaneamente a estimativa de reutilização futura dos segmentos e a concentração de clientes próximos aos blocos de vídeo.

A motivação para o desenvolvimento deste algoritmo está relacionada às limitações observadas em ambientes de simulação baseados exclusivamente em condições ideais de operação. Conforme pode ser visto em Basiri et al. (2016), em sistemas reais de distribuição de conteúdo, a dinâmica de acesso dos usuários pode ser influenciada por variações de carga, mudanças nos padrões de consumo e oscilações na disponibilidade dos

recursos computacionais e de rede. Nessas situações, critérios baseados exclusivamente na reutilização temporal ou apenas na popularidade dos blocos podem não representar adequadamente o comportamento observado durante a execução do sistema, como oscilações entre disponibilidade de recursos, sejam por memória, largura de banda e/ou ambos simultaneamente.

Dessa forma, o RTBC-CARTE procura combinar a capacidade do $RTBC$ de estimar a reutilização futura dos segmentos com o mecanismo de popularidade local empregado pelo $CARTE_{BB}$. A política resultante prioriza blocos que apresentam potencial de reutilização por clientes ativos e, simultaneamente, considera a distribuição dos acessos em torno de cada segmento armazenado. Com isso, busca-se reduzir substituições de blocos que possam ser requisitados em curto prazo, preservando informações relevantes tanto da perspectiva temporal quanto da concentração de demanda observada no cache.

### 5.3.1 Composição matemática

A formulação proposta para o cálculo de R_hibrido foi inspirada nos princípios de priorização discutidos por Harchol-Balter (2013), nos quais a seleção de tarefas e recursos é frequentemente baseada na relação entre uma métrica de benefício e fatores que representam custo, carga ou demanda do sistema. Nesse contexto, o valor R_RTBC representa a estimativa temporal de reutilização do segmento obtida pelo algoritmo RTBC, constituindo a métrica principal de decisão. O parâmetro PopJanela_CARTE corresponde à popularidade local do bloco observada em uma janela de análise, atuando como um fator de ajuste associado à concentração de demanda sobre o segmento. Já o parâmetro $\alpha$ controla a intensidade da influência exercida pela popularidade na classificação final. Dessa forma, a expressão reduz o valor efetivo do tempo de reutilização à medida que a popularidade local aumenta, tornando menos provável a substituição de blocos que apresentam maior potencial de reutilização imediata. O valor 1 que está presente no denominador, evita a indeterminação de divisão por zero, pois pode acontecer que a população da janela seja nulo. A estrutura da equação preserva a lógica temporal do RTBC, ao mesmo tempo em que incorpora informações de demanda local derivadas do $CARTE_{BB}$, produzindo uma métrica única para ordenação dos segmentos em cache.

Assim, o tempo de reutilização calculado pelo RTBC é ajustado por um fator inversamente proporcional à popularidade local observada pelo mecanismo $CARTE_{BB}$.

Os segmentos com maior concentração de clientes apresentam redução no valor efetivo de reutilização, tornando-se menos suscetíveis à substituição.

### 5.3.2 Modelagem do Algoritmo RTBC-CARTE

O algoritmo híbrido RTBC-CARTE é composto por estruturas responsáveis pelo monitoramento dos acessos aos vídeos, manutenção do histórico temporal de requisições, cálculo dos tempos de reutilização, estimativa da popularidade local dos blocos e gerenciamento da política de substituição. Essas estruturas atuam de forma integrada para combinar informações temporais de reutilização derivadas do RTBC, com informações de concentração local de demanda, inspiradas no $CARTE_{BB}$. Os parâmetros window_w, janela_carte e $\alpha$ controlam, respectivamente, a janela de observação utilizada no cálculo da frequência histórica, o alcance da análise dos blocos e a intensidade da influência de popularidade local na classificação final dos segmentos.

O cálculo do tempo de reutilização segue os princípios do RTBC e é implementado pela função 17. Inicialmente, é verificada a existência de clientes ativos que ainda vão solicitar determinado segmento. Quando essa condição é satisfeita, o algoritmo utiliza o Tempo Exato de Reuso $(TER)$, definido pela distância entre o bloco analisado e o cliente mais próximo que o requisitará, como já explicado detalhadamente na subseção 5.2.1. Na ausência de consumidores futuros identificáveis, é empregado o Tempo Previsto de Reuso $(PRT)$, calculado a partir da frequência histórica de acessos ao vídeo.

Além da componente temporal, o algoritmo incorpora um mecanismo de popularidade local inspirado no $CARTE_{BB}$. Para cada segmento armazenado em cache, a função mostrada na Figura 19 contabiliza a quantidade de clientes ativos cujas posições de reprodução se encontram dentro de uma janela anterior ao bloco analisado. O resultado desse processo corresponde à métrica PopJanela_CARTE, utilizada para representar a concentração de demanda associada ao segmento.

A integração entre os mecanismos RTBC e $CARTE_{BB}$ é apresentada na Figura 20, responsável pelo cálculo da métrica híbrida de reutilização. Inicialmente, é obtido o valor R_RTBC por meio da função mostrada na Figura 17. Em seguida, é calculada a popularidade local PopJanela_CARTE, apresentada na Figura 19, que é utilizada para ajustar o valor temporal segundo a expressão:

$$R\_hibrido = \frac{R\_RTBC}{(1+\alpha * PopJanela_{CARTE})}$$

Figura 19 – Popular Janela CARTE - Algoritmo

```
def pop_janela_carte(self, lista_filme, k):
        lo = k - self.janela_carte
        pop = 0
        for cliente in lista_filme.clientes.values():
            if cliente.idBloco >= 0 and cliente.idBloco < k and cliente.idBloco >= lo:
                pop += 1
        return pop
```

Fonte: do Autor(2026)

onde $(\alpha)$ representa o fator de ponderação da influência da popularidade na janela. Dessa forma, segmentos associados a maiores concentrações de clientes apresentam redução no valor efetivo de reutilização, tornando-se menos prováveis ao processo de substituição.

Figura 20 – Reuse Time Hybrid - Algoritmo

```
def reuse_time_hibrido(self, lista_filme, k):
        r_rtbc = self._reuse_time_rtbc(lista_filme, k)
        pop = self._pop_janela_carte(lista_filme, k)
        return r_rtbc / (1.0 + self.alpha_carte * float(pop))
```

Fonte: do Autor(2026)

Após o cálculo da métrica híbrida, cada bloco recebe uma chave de classificação construída pela função da Figura 18, já apresentada anteriormente. A composição dessa chave utiliza informações produzidas pela função da Figura 20 e auxiliares, responsáveis pelo cálculo da reutilização híbrida e pela manutenção da estrutura de acesso recente dos filmes. A ordenação resultante considera o tempo de reutilização híbrido, a quantidade de espectadores ativos, a posição do filme na lista de ordenação e o identificador do bloco, estabelecendo a prioridade utilizada pela política de substituição.

Quando ocorre a necessidade de liberar espaço em memória, o algoritmo executa o procedimento de substituição por meio da função `substituicaoBlocos()`. Antes da remoção de um segmento, as prioridades armazenadas são atualizadas, garantindo que a estrutura de ordenação reflita o estado corrente do sistema. O registro dos acessos ao primeiro segmento e a atualização das informações temporais também são realizados, mantendo os dados necessários para o cálculo das métricas de reutilização. Após a seleção do candidato à remoção, o novo segmento é inserido na posição liberada.

## 5.4 Cenários realistas - Uso de *Chaos Engineering*

A avaliação de algoritmos costuma ser frequentemente realizada sob condições estáveis de operação, onde estão disponíveis recursos computacionais, capacidade de rede e padrões de acesso constantes ao longo da execução. Embora esta abordagem permita analisar o comportamento dos mecanismos propostos, não são contemplados às variações que são observadas em sistemas reais. Segundo Basiri et al. (2016), a dinâmica operacional de ambientes distribuídos é influenciada por alterações de carga, limitações temporárias de recursos, degradações parciais de infraestrutura e mudanças no comportamento dos usuários, fatores que podem impactar diretamente o desempenho dos componentes avaliados.

Nesse contexto, estratégias associadas ao *Chaos Engineering* propõem a introdução de perturbações controladas durante a execução dos sistemas, permitindo observar seus mecanismos de adaptação, degradação e recuperação. A abordagem não tem como objetivo provocar falhas arbitrárias, mas reproduzir condições operacionais que possibilitem avaliar o comportamento dos componentes sob diferentes estados do ambiente. Dessa forma, torna-se possível analisar não apenas o desempenho obtido em condições nominais, mas também a forma como algoritmos e mecanismos de gerenciamento respondem a alterações na demanda e na disponibilidade de recursos.

Considerando essas premissas, este trabalho propõe a extensão do simulador original por meio da introdução de mecanismos capazes de representar variações temporais de carga, restrições de largura de banda e heterogeneidade de clientes. A estratégia busca ampliar o conjunto de cenários avaliados, permitindo que os algoritmos de substituição sejam analisados tanto em condições estáveis quanto em situações que reproduzam oscilações operacionais observadas em ambientes distribuídos.

### 5.4.1 Estratégia aplicada - Picos de clientes $X$ Largura de Banda

A estratégia foi aplicada como extensão do simulador por meio de mecanismos configuráveis, capazes de representar variações temporais de carga, restrições de largura de banda e diversidade de clientes. A estrutura mantém compatibilidade com o fluxo original do simulador, permitindo que os novos componentes sejam ativados ou desativados de forma isolada, através de variáveis de ambiente. Seu objetivo é observar o comportamento dos algoritmos em cenários de degradação e recuperação operacional

dentro das camadas de testes.

Foram definidas três variáveis de ativação: *ENABLE_PHASE_ARRIVALS*, que determinam aleatoriedade de chegadas de clientes no tempo, *ENABLE_DYNAMIC_BANDWIDTH*, responsável pela largura de banda variável durante a simulação e *ENABLE_CLIENT_PROFILES*, que simula os tipos de conexões dos clientes, no que diz respeito a perfis de conexão arbitrários.

O mecanismo de chegadas por fase foi desenvolvido para representar variações temporais na taxa de admissão de clientes ao longo da execução da simulação. A implementação utiliza uma função temporal definida por intervalos de execução, na qual cada período possui um multiplicador associado à taxa de admissão de clientes. Em cada ciclo da simulação, esse multiplicador determina a quantidade máxima de novos usuários que podem ingressar no sistema. Dessa forma, torna-se possível representar intervalos de operação estável, crescimento de demanda e redução da carga de trabalho. Para simular os picos de clientes, é implementada uma taxa temporal variável de admissão no sistema. A estratégia divide a execução em intervalos distintos, representando estados como operação normal e picos de requisições. O modelo altera dinamicamente a intensidade de entrada de clientes no laço principal da simulação, permitindo representar oscilações de demanda semelhantes às utilizadas em testes de resiliência operacional (ROSENTHAL; JONES, 2020).

O mecanismo de perfis de clientes introduzido pela variável de ambiente *ENABLE_CLIENT_PROFILES*, propõe diversidade no comportamento dos usuários representados na simulação. Cada cliente passa a possuir uma capacidade individual de consumo de conteúdo, permitindo representar diferentes condições de acesso aos recursos disponibilizados pelo sistema. No momento de sua criação, cada cliente recebe um perfil associado a uma categoria de capacidade de transmissão. A atribuição é realizada com base em uma distribuição configurável definida para o cenário de experimento. Cada perfil determina a frequência com que o cliente pode avançar na reprodução dos blocos de vídeo durante a execução da simulação. As informações de perfil são armazenadas juntamente com os atributos do cliente e passam a influenciar as decisões relacionadas ao atendimento das requisições.

Como exemplo, considere um cenário em que três clientes ingressam no sistema no mesmo instante. Com a funcionalidade *ENABLE_CLIENT_PROFILES* ativada, cada usuário recebe um perfil de forma probabilística. Supondo que o usuário A receba o perfil *high*, o usuário B o perfil *mid* e o usuário C o perfil *low*, seus ritmos de consumo

serão diferentes. O usuário A avança um bloco a cada ciclo da simulação, assim como o usuário B avança um bloco a cada dois ciclos e o usuário C avança um bloco a cada três ciclos. Embora todos permaneçam conectados ao sistema durante esse período, apenas os clientes cujo perfil permite progressão no ciclo corrente realizam novas requisições. Como consequência, clientes com menor taxa de consumo permanecem por mais tempo associados aos blocos iniciais do vídeo, modificando a distribuição de usuários ao longo dos segmentos armazenados em cache.

Esse comportamento produz efeitos sobre as decisões das políticas de substituição. Algoritmos que consideram a concentração de clientes, como o $CARTE_{BB}$, podem identificar uma maior permanência de usuários em determinados segmentos, enquanto algoritmos que utilizam informações de reutilização temporal, como o *RTBC*, passam a observar padrões de acesso distintos daqueles produzidos por clientes homogêneos. Dessa forma, a introdução de perfis de consumo permite representar situações mais próximas de ambientes nos quais coexistem usuários com diferentes capacidades de acesso à rede.

### 5.4.2 Ambiente experimental

Os experimentos foram executados em um computador equipado com processador *Intel Core i5-1135G7* de 11ª geração, com frequência de 2,40 GHz, 12 GB de memória *RAM* e unidade de armazenamento *SSD* de 256 GB, operando em sistema operacional *Ubuntu Linux*. A descrição do ambiente experimental tem como finalidade registrar a plataforma utilizada e permitir a reprodução dos experimentos em condições equivalentes.

A carga de trabalho foi definida por meio de parâmetros estabelecidos antes da execução das simulações. O acervo utilizado foi composto por 10.000 vídeos e cada experimento foi executado durante 6.600 segundos virtuais de simulação. Essas configurações permaneceram constantes em todos os experimentos, permitindo que a comparação entre os algoritmos fosse realizada sob as mesmas condições de execução.

A distribuição de popularidade dos conteúdos foi representada por parâmetros de $Zipf$ iguais a 0.2, 0.6, 0.8 e 1.0. Esses valores permitiram avaliar o comportamento dos algoritmos em diferentes níveis de concentração de acessos, desde distribuições mais uniformes até cenários em que um conjunto reduzido de conteúdos concentra a maior parte das requisições. A intensidade de chegada de clientes foi controlada por meio da distribuição de Poisson. Inicialmente, foram utilizados os intervalos médios de chegada 1, 4, 7 e 10. Posteriormente, a análise foi ampliada utilizando os valores 1, 3, 5, 7, 9,

11, 13, 15, 17 e 20, permitindo observar o comportamento dos algoritmos em diferentes níveis de carga. A tabela 4 reforça os dados apresentados, aos quais os resultados obtidos são apresentados com maior detalhe.

Tabela 4 – Configuração do ambiente experimental

| **Parâmetro** | **Valor** |
|---|---|
| Processador | Intel Core i5-1135G7 (11ª geração), 2,40 GHz |
| Memória RAM | 12 GB |
| Memória RAM usada por Thread | 1 GB |
| Armazenamento | SSD 256 GB |
| Sistema operacional | Ubuntu Linux |
| Linguagem de implementação | Python 3.6 |
| Número de vídeos | 10.000 |
| Número de ciclos por simulação | 6.600 |
| Distribuição de popularidade (Zipf) | 0.2; 0.6; 0.8; 1.0 |
| Intervalo médio entre chegadas (Poisson) | 1, 3, 5, 7, 9, 11, 13, 15, 17 e 20 |
| Quantidade total de ciclos por filme ($W$) | 6600 segundos virtuais |
| Fator de ponderação do algoritmo híbrido ($\alpha$) | 0,5 |
| Distribuição de perfis de clientes | Configurável por cenário |
| Variação temporal de chegadas | Configurável por cenário |
| Variação temporal de largura de banda | Configurável por cenário |

# 6 RESULTADOS

Os algoritmos *RTBC, CARTE$_{BB}$ e Híbrido (RTBC-CARTE)* foram analisados ao longo deste trabalho sob diferentes perspectivas relacionadas à gestão de cache em sistemas de vídeo sob demanda. O *RTBC* fundamenta suas decisões na estimativa do tempo de reutilização dos segmentos, utilizando informações temporais para determinar quais blocos devem permanecer armazenados. O *CARTE$_{BB}$*, por sua vez, utiliza informações associadas à concentração de clientes e à popularidade local dos segmentos, orientando a substituição dos blocos a partir da demanda observada no sistema. A proposta híbrida *RTBC-CARTE* foi desenvolvida com o objetivo de combinar características presentes em ambas as abordagens, incorporando simultaneamente informações de reutilização temporal e popularidade local no processo de classificação dos segmentos.

Além da análise dos mecanismos de substituição, foram apresentados modelos de carga de trabalho baseados em distribuições de *Zipf* e *Poisson*, amplamente utilizados na literatura para representar, respectivamente, a popularidade dos conteúdos e a chegada de usuários em sistemas de streaming. Também foram introduzidos mecanismos inspirados em estratégias de *Chaos Engineering*, permitindo a representação de variações temporais de carga e alterações na disponibilidade de largura de banda entre clientes. Cada conjunto de resultados provenientes das diferentes cargas, possuem 30 arquivos de saída por intervalo *Poisson*, aos quais são calculados seus pontos de média e desvio padrão. Esses componentes ampliam o conjunto de cenários avaliados, possibilitando observar o comportamento dos algoritmos tanto em condições controladas quanto em situações caracterizadas por mudanças na demanda e na disponibilidade de recursos.

A partir dessa fundamentação, esta seção apresenta os resultados obtidos durante os experimentos realizados no simulador. Inicialmente são analisados os cenários controlados utilizados para comparação entre os algoritmos *RTBC* e *CARTE$_{BB}$*. Em seguida, inserido o algoritmo híbrido às simulações, são discutidos os resultados produzidos pelos cenários que incorporam variações de carga, largura de banda e perfis de clientes, permitindo avaliar a influência dessas condições sobre o desempenho das políticas de substituição implementadas.

A Figura 21 apresenta a comparação entre os algoritmos *RTBC* e *CARTE$_{BB}$* utilizando uma carga de trabalho gerada com parâmetro *Zipf* igual a $0,271$ e intervalos médios de chegada definidos por valores de *Poisson* entre $\{1, 4, 7, 10\}$. Para este cenário

de carga, possui um resultante de 210 arquivos de saída, aos quais foram analisados e extraído o gráfico da Figura 21. Estes valores foram inspirados pela carga de trabalho utilizada por Neves (2015) em seu trabalho, buscando obter-se condição de equivalência para execução do novo algoritmo implementado, o $RTBC$. Observa-se que ambos os algoritmos apresentam crescimento da taxa de acertos à medida que o intervalo entre chegadas aumenta. Esse comportamento está associado à redução da concorrência pelos recursos de cache, permitindo que os segmentos armazenados permaneçam disponíveis por períodos mais longos antes de serem substituídos.

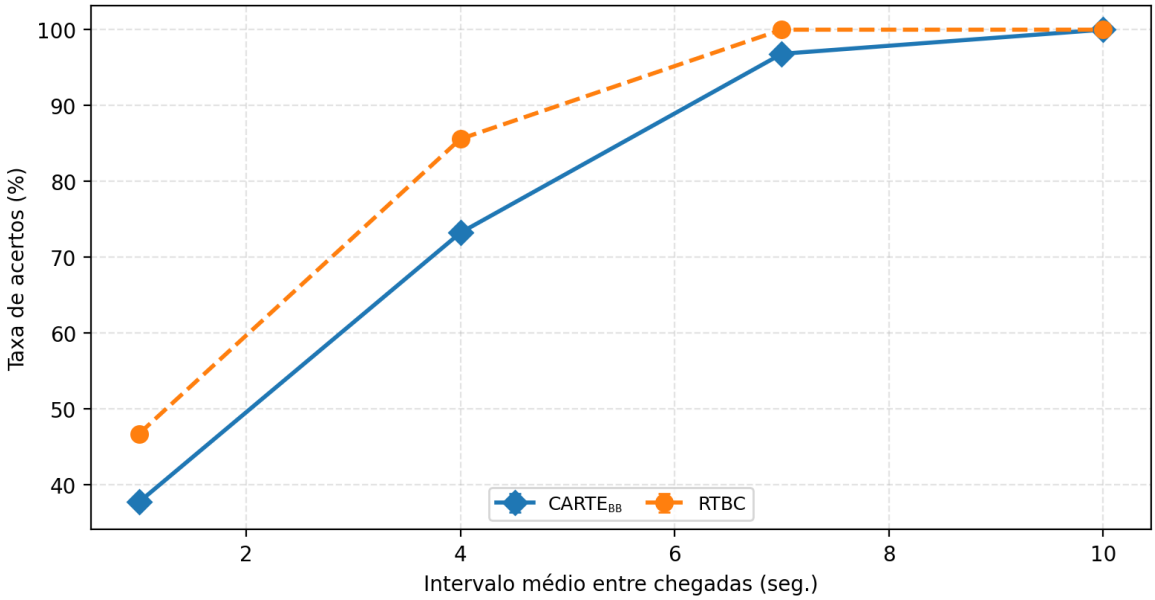
Apesar da tendência comum de crescimento, o *RTBC* mantém taxas de acerto superiores às obtidas pelo $CARTE_{BB}$ em todos os cenários avaliados. A diferença torna-se mais evidente para os menores intervalos de chegada, situação em que a quantidade de clientes ativos e a frequência de substituições no cache são mais elevadas. Enquanto o $CARTE_{BB}$ fundamenta suas decisões na concentração de clientes em torno dos segmentos armazenados, o *RTBC* utiliza estimativas de reutilização futura para determinar a prioridade dos blocos. As cargas de trabalho empregadas favorecem mecanismos de reutilização temporal devido ao consumo sequencial dos vídeos, à ausência de saltos de reprodução e à utilização de distribuições de *Poisson* e $Zipf$, que produzem padrões de acesso relativamente previsíveis. Nessas condições, o algoritmo $RTBC$ consegue estimar com maior precisão a reutilização futura dos segmentos, resultando em taxas de acerto superiores às obtidas pelo $CARTE_{BB}$. Como consequência, segmentos com maior probabilidade de acesso em curto prazo tendem a permanecer armazenados por mais tempo, reduzindo substituições que poderiam resultar em novas requisições ao servidor de origem.

Esse comportamento está alinhado aos resultados apresentados por Wu et al. (2012) que demonstram a capacidade de políticas baseadas em tempo de reutilização aproximarem o comportamento do algoritmo ótimo em sistemas de vídeo sob demanda. Nessas condições, a utilização de informações sobre acessos futuros contribui para aumentar a permanência dos segmentos com maior potencial de reutilização.

À medida que o intervalo entre chegadas aumenta, as curvas dos dois algoritmos convergem para taxas de acerto próximas de 100%. Esse resultado indica que a disponibilidade de recursos passa a exercer influência maior sobre o desempenho do sistema do que a política de substituição empregada. Com a redução da intensidade de carga, a maior parte dos segmentos permanece em cache durante tempo suficiente para atender futuras requisições, reduzindo a frequência de substituições e, consequentemente,

a diferença entre os mecanismos de decisão adotados pelos algoritmos avaliados.

Figura 21 – Comparativo RTBC *X* $CARTE_{BB}$




Fonte: do Autor(2026)

A Figura 22 amplia a análise realizada anteriormente ao considerar uma carga de trabalho gerada com parâmetro *Zipf* igual a $0,6$ e intervalos médios de chegada variando entre 1 e 20 unidades de tempo. Os resultados mostram que o aumento do intervalo entre chegadas produz crescimento progressivo da taxa de acertos para os algoritmos *RTBC* e $CARTE_{BB}$ até a região compreendida entre os valores 8 e 10. Esse comportamento está associado à redução da intensidade de chegada de clientes, que diminui a competição pelos recursos de cache e reduz a frequência de substituição dos segmentos armazenados.

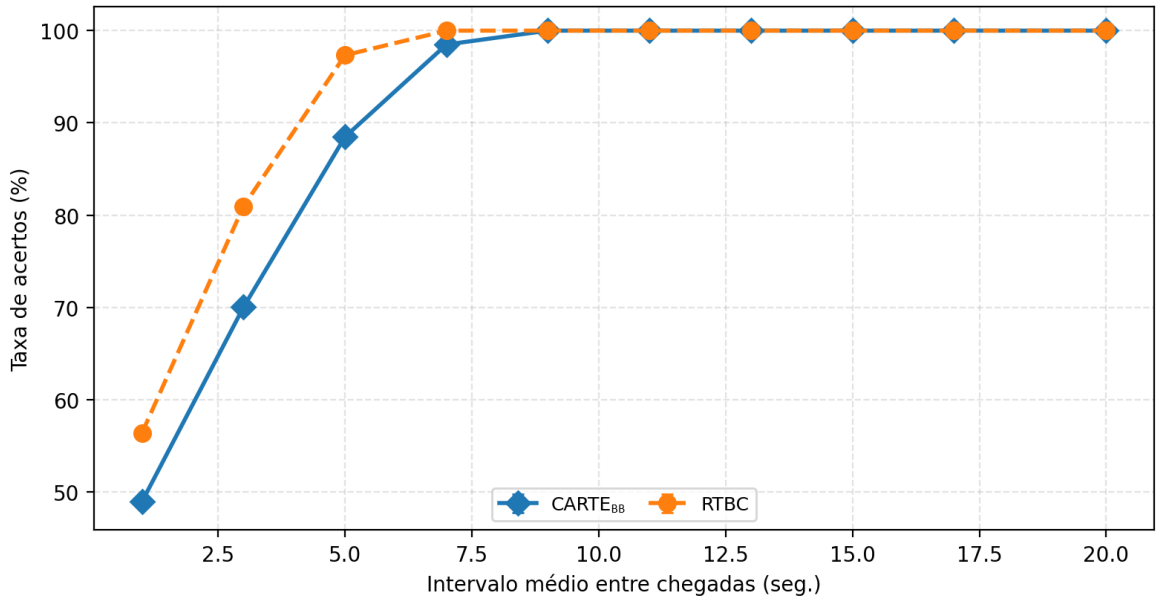
Ao longo da fase inicial do experimento, o *RTBC* apresenta taxas de acerto superiores às observadas para o $CARTE_{BB}$, especialmente nos menores intervalos de chegada. Esse resultado reforça a contribuição dos mecanismos de reutilização temporal em cenários caracterizados por maior volume de acessos simultâneos. Nessa condição, a utilização de estimativas sobre a reutilização futura dos segmentos permite ao *RTBC* preservar blocos com maior probabilidade de acesso, reduzindo substituições que poderiam resultar em novas requisições ao servidor de origem. O comportamento observado é compatível com os resultados apresentados por Wu et al. (2012), nos quais a utilização de informações temporais contribui para aumentar a eficiência do gerenciamento de cache em sistemas de vídeo sob demanda.

À medida que o intervalo entre chegadas aumenta, a diferença entre os algoritmos

diminui progressivamente, até que ambos passam a apresentar taxas de acerto próximas de 100%. Esse comportamento indica a existência de um ponto de estabilização a partir do qual a redução da carga deixa de produzir ganhos significativos no desempenho do sistema.

A convergência observada para os maiores intervalos de *Poisson* sugere que o sistema passa a operar em uma condição na qual a maior parte dos conteúdos requisitados já se encontra disponível em *cache*. Nessa situação, a necessidade de remoção de segmentos ocorre com menor frequência e a disponibilidade de recursos passa a exercer influência maior sobre os resultados do que a política de substituição adotada. Como consequência, as diferenças estruturais entre *RTBC* e $CARTE_{BB}$ tornam-se menos relevantes para o desempenho global do sistema, produzindo curvas de comportamento semelhantes nos cenários de menor intensidade de carga.

Figura 22 – Comparativo RTBC *X* $CARTE_{BB}$




Fonte: do Autor(2026)

A Figura 23 apresenta a comparação entre o algoritmo híbrido RTBC-CARTE, o algoritmo $CARTE_{BB}$ e o algoritmo RTBC, utilizando a mesma configuração experimental empregada na Figura 21. Os resultados demonstram que a incorporação dos mecanismos de reutilização temporal de RTBC produzem aumento da taxa de acertos em todos os intervalos de chegada avaliados. Esse comportamento indica que a utilização simultânea de informações temporais e de popularidade local contribui para reduzir a substituição de

segmentos que apresentam potencial de reutilização em curto prazo.

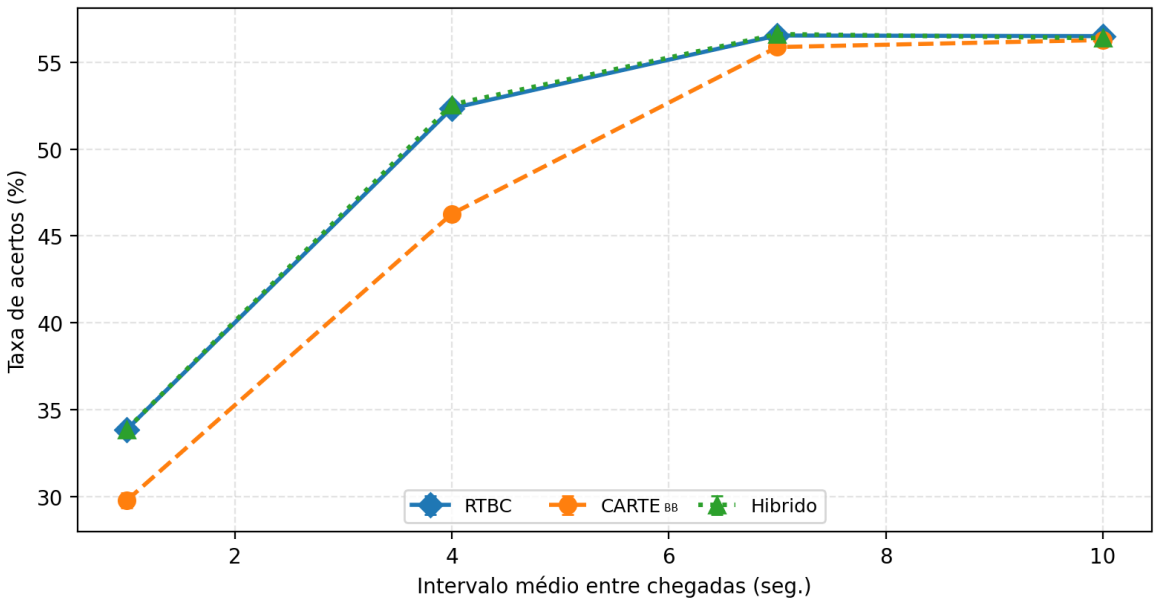
A diferença entre os algoritmos torna-se mais evidente para os menores valores de *Poisson*, condição na qual a chegada de clientes ocorre com maior intensidade e a concorrência pelos recursos de *cache* é mais elevada. Nesses cenários, a redução do intervalo entre chegadas aumenta simultaneamente a quantidade de clientes ativos e a frequência de substituições realizadas pelo sistema. Como consequência, tornam-se mais relevantes tanto as informações de reutilização futura dos segmentos quanto a concentração local de clientes observada em torno dos blocos armazenados. Ao combinar esses dois critérios, o algoritmo híbrido consegue identificar segmentos com maior probabilidade de reutilização em curto prazo, contribuindo para a manutenção de taxas de acerto superiores às observadas nas abordagens que utilizam apenas uma dessas informações. Como consequência, segmentos que apresentam elevada probabilidade de reutilização tendem a permanecer em *cache* por mais tempo, reduzindo a ocorrência de novas requisições ao servidor de origem.

O ganho observado em relação ao $CARTE_{BB}$ demonstra que a inclusão das informações temporais complementa a estratégia baseada exclusivamente na popularidade local. Enquanto o $CARTE_{BB}$ prioriza segmentos associados à concentração de clientes em uma janela específica, o algoritmo híbrido incorpora também a previsão de reutilização futura, ampliando os critérios considerados durante o processo de classificação dos blocos. Essa característica torna-se particularmente relevante em situações nas quais o sistema realiza um número elevado de substituições, aumentando a importância da seleção dos segmentos que devem permanecer armazenados.

À medida que o intervalo entre chegadas aumenta, os resultados dos dois algoritmos passam a convergir. Esse comportamento sugere que a redução da intensidade de carga diminui a pressão exercida sobre o *cache* e reduz a frequência de substituições. Ainda assim, o algoritmo híbrido mantém taxas de acerto superiores às obtidas pelo $CARTE_{BB}$ em todas as configurações avaliadas, indicando que a combinação entre reutilização temporal e popularidade local permanece efetiva mesmo quando a concorrência pelos recursos do sistema é reduzida.

As Figuras 24 e 25 apresentam os resultados obtidos após a ativação dos mecanismos de chegadas por fase, largura de banda variável e perfis de clientes. Diferentemente dos cenários controlados analisados anteriormente, esses experimentos incorporam variações temporais de carga e disponibilidade de recursos, permitindo avaliar o comportamento dos algoritmos sob condições inspiradas em ambientes operacionais nos

Figura 23 – Comparativo Hibrido *X* $CARTE_{BB}$ *X* RTBC

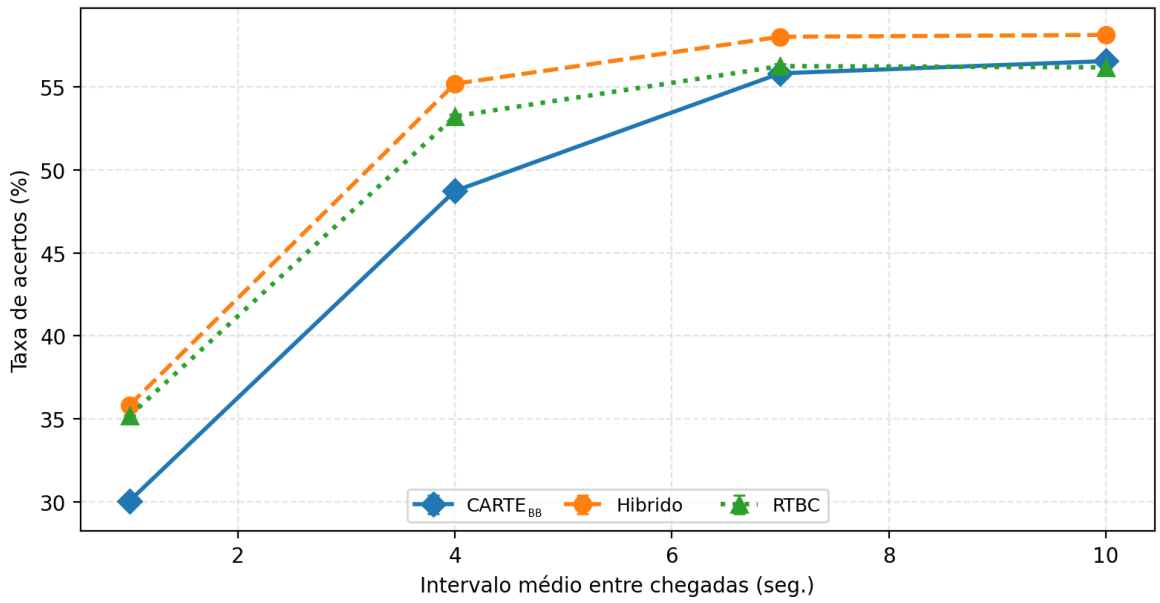



Fonte: do Autor(2026)

quais a demanda e a capacidade do sistema se alteram ao longo da execução.

A Figura 24 apresenta os resultados para a carga de trabalho gerada com parâmetro $Zipf$ igual a $0,271$. Observa-se que o algoritmo híbrido mantém as maiores taxas de acerto em todos os intervalos de chegada avaliados, seguido pelo *RTBC* e, posteriormente, pelo $CARTE_{BB}$. Esse comportamento pode ser explicado pelos mecanismos utilizados na composição da métrica híbrida. Enquanto o *RTBC* utiliza exclusivamente informações relacionadas à reutilização temporal dos segmentos, o algoritmo híbrido incorpora adicionalmente a métrica de popularidade local. Nos cenários com chegadas por fase, largura de banda variável e perfis heterogêneos de clientes, a distribuição dos usuários ao longo dos vídeos torna-se mais dinâmica, produzindo regiões com diferentes níveis de concentração de acessos. Nessas condições, a componente temporal continua permitindo identificar segmentos com elevada probabilidade de reutilização futura, enquanto a componente de popularidade local destaca blocos associados a maiores concentrações de clientes. Como resultado, a métrica híbrida passa a utilizar simultaneamente informações sobre reutilização futura e demanda observada, contribuindo para a preservação dos segmentos mais relevantes mesmo diante das variações introduzidas no ambiente de simulação. Nessas condições, a utilização simultânea de múltiplos critérios de decisão permite ao algoritmo híbrido responder às mudanças do ambiente sem comprometer a permanência dos blocos mais relevantes em cache.

Figura 24 – Taxa de Acertos RTBC X $CARTE_{BB}$ X Hibrido - Cenário com Instabilidades de Recursos




Fonte: do Autor(2026)

Comportamento semelhante é observado na Figura 25, correspondente à carga gerada com parâmetro *Zipf* igual a $0,6$. Embora o algoritmo híbrido continue apresentando os maiores índices de acerto, a diferença em relação ao *RTBC* torna-se menos pronunciada à medida que o intervalo entre chegadas aumenta. Esse resultado indica que a redução da intensidade de carga diminui a pressão exercida sobre o cache, reduzindo a frequência de substituições e, consequentemente, a influência da política de gerenciamento sobre o resultado final.

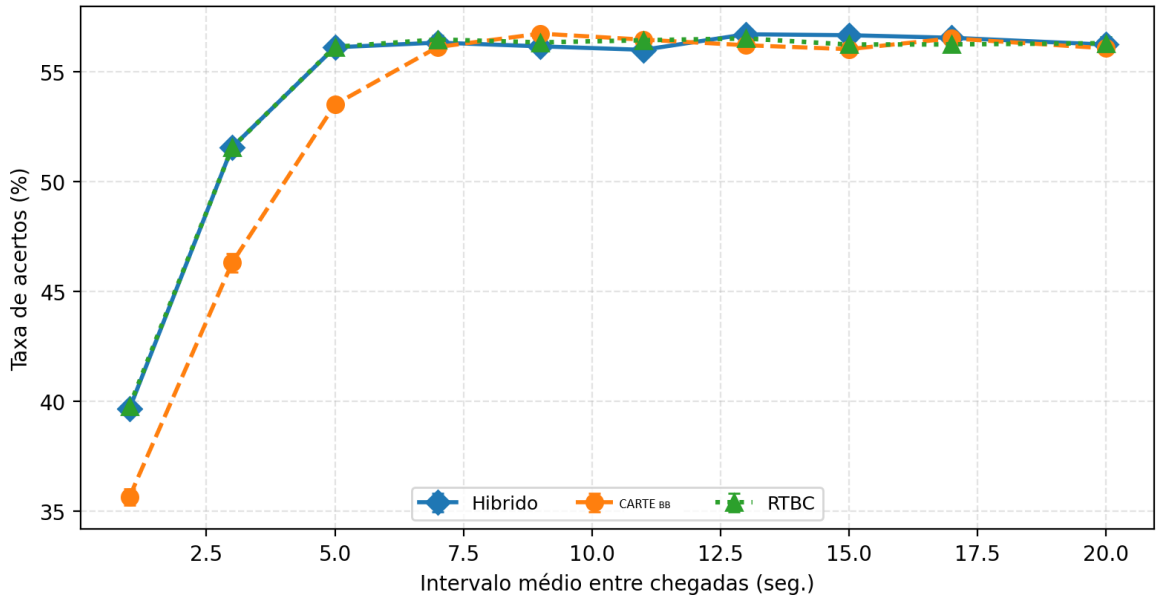
A comparação entre os cenários controlados e os cenários com variações operacionais evidencia que os algoritmos respondem de maneira distinta às alterações introduzidas no ambiente de simulação. A presença simultânea de oscilações na admissão de clientes, modificações na capacidade de transmissão e heterogeneidade entre usuários altera a distribuição dos acessos ao longo do tempo, influenciando diretamente a reutilização dos segmentos armazenados e a ocupação da memória cache. Nesse contexto, o algoritmo híbrido demonstra maior capacidade de adaptação, mantendo desempenho superior ou equivalente aos demais algoritmos ao longo da maior parte dos experimentos realizados.

Esse comportamento é consistente com princípios observados em estudos de *Chaos Engineering*, nos quais sistemas submetidos a condições variáveis apresentam

respostas diferentes daquelas observadas em cenários estáticos (ROSENTHAL; JONES, 2020). Ao introduzir perturbações controladas no ambiente de simulação, torna-se possível observar como os mecanismos de decisão reagem às alterações de carga e disponibilidade de recursos. Os resultados obtidos indicam que a combinação entre reutilização temporal e popularidade local permite ao algoritmo híbrido manter desempenho estável mesmo diante das variações introduzidas, preservando sua capacidade de retenção dos segmentos mais relevantes.

Por fim, observa-se que as diferenças entre os algoritmos diminuem para os maiores intervalos de chegada, comportamento semelhante ao verificado nos cenários controlados. Com a redução da intensidade de carga, a quantidade de substituições realizadas pelo sistema também diminui, reduzindo a influência dos mecanismos de classificação sobre o desempenho global. Como consequência, as curvas de acerto tendem a convergir, embora o algoritmo híbrido mantenha vantagem sobre as demais abordagens ao longo dos experimentos analisados.

Figura 25 – Taxa de Acertos RTBC X $CARTE_{BB}$ X Hibrido - Cenário com instabilidades de Recursos




Fonte: do Autor(2026)

A Figura 26 apresenta a comparação entre os algoritmos *RTBC*, $CARTE_{BB}$ e *RTBC-CARTE* para diferentes valores do parâmetro *Zipf*, com intervalo de *Poisson* fixo em 1.0. O objetivo deste experimento consiste em analisar a influência da concentração de popularidade dos conteúdos sobre o desempenho das políticas de substituição

implementadas no simulador. Como discutido anteriormente, o parâmetro *Zipf* controla a distribuição das requisições entre os vídeos disponíveis, permitindo representar cenários nos quais a demanda está distribuída entre diversos conteúdos ou concentrada em um conjunto reduzido de objetos.

Os resultados demonstram que todos os algoritmos apresentam crescimento da taxa de acertos à medida que o parâmetro *Zipf* aumenta. À medida que um número maior de requisições passa a incidir sobre um conjunto menor de conteúdos, aumenta também a probabilidade de reutilização dos segmentos armazenados, reduzindo a necessidade de acesso ao servidor de origem e elevando a eficiência das políticas de substituição.

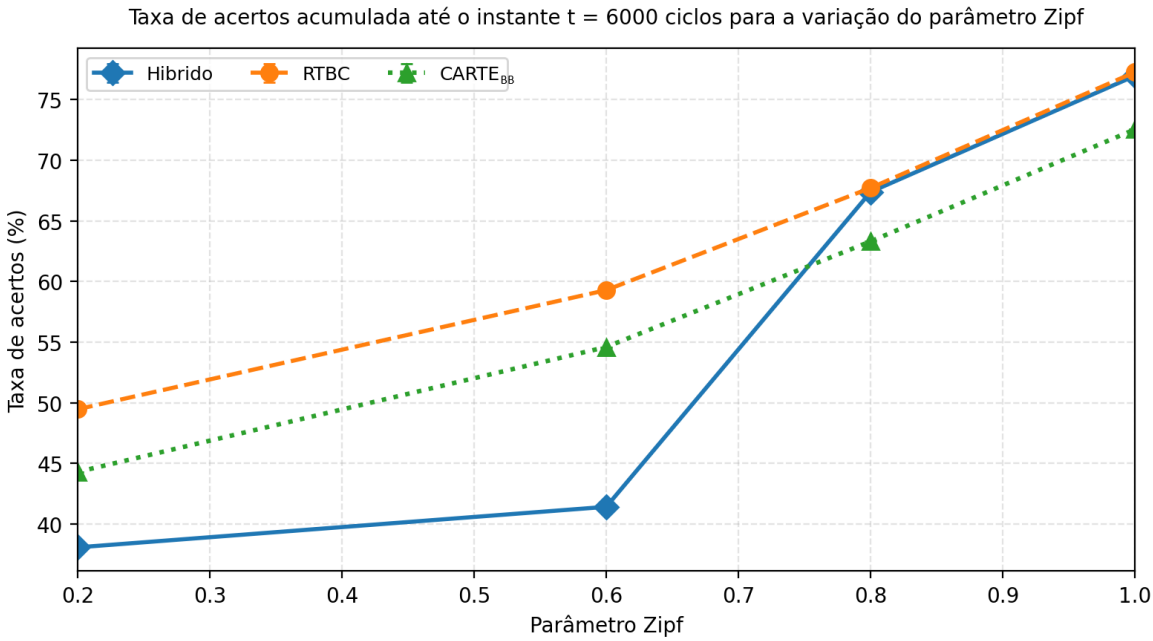
Para valores reduzidos de Zipf, a demanda encontra-se distribuída entre uma quantidade maior de vídeos, aumentando a diversidade de segmentos requisitados e a frequência de substituições realizadas pelo sistema. Nessa condição, o *RTBC* apresenta as maiores taxas de acerto entre os algoritmos avaliados. Esse resultado sugere que os mecanismos de reutilização temporal exercem papel relevante quando a popularidade dos conteúdos é menos concentrada, permitindo identificar segmentos com maior probabilidade de reutilização futura mesmo diante de um conjunto mais amplo de objetos concorrendo pelos recursos de cache.

À medida que o parâmetro *Zipf* aumenta, observa-se uma aproximação progressiva entre os resultados do algoritmo híbrido *RTBC-CARTE* e aqueles obtidos pelo *RTBC*, enquanto a diferença em relação ao $CARTE_{BB}$ torna-se mais evidente. Esse comportamento indica que a componente de popularidade local passa a contribuir de forma mais efetiva quando a demanda se concentra em um subconjunto dos conteúdos disponíveis. Nessa situação, a combinação entre reutilização temporal e concentração local de acessos permite ao algoritmo híbrido identificar segmentos que apresentam simultaneamente potencial de reutilização futura e elevada frequência de acesso em curto prazo.

Os resultados evidenciam que o algoritmo híbrido mantém desempenho próximo ao melhor resultado observado em cada configuração experimental, dentro dos cenários testados neste trabalho. Em cenários com menor concentração de popularidade, o comportamento aproxima-se daquele apresentado pelo *RTBC*, beneficiando-se da utilização de informações temporais de reutilização. Em cenários com maior concentração de acessos, a influência da popularidade local torna-se mais significativa, contribuindo para a retenção dos segmentos mais requisitados. Dessa forma, a integração entre os mecanismos do *RTBC* e do *CARTE_BB* permite ao algoritmo híbrido adaptar-se a

diferentes distribuições de demanda sem apresentar perdas significativas de desempenho nos cenários avaliados.

Figura 26 – Comparativo Zipf Hibrido *X* RTBC *X* CARTE$_{BB}$




Fonte: do Autor(2026)

# 7 CONCLUSÕES

Os resultados obtidos ao longo dos experimentos demonstram que as políticas de substituição avaliadas apresentam comportamentos distintos conforme as características da carga de trabalho e as condições de execução do sistema. O RTBC apresentou vantagens em cenários nos quais a reutilização temporal dos segmentos exerce maior influência sobre o desempenho do cache, enquanto o $CARTE_{BB}$ demonstrou sensibilidade à concentração local de acessos. O algoritmo híbrido RTBC-CARTE, por sua vez, embora não tenha apresentado os maiores resultados no cenário caracterizado por variação do parâmetro *Zipf* e intervalos de *Poisson* constantes, manteve desempenho consistente ao longo dos diferentes cenários avaliados. A combinação entre informações de reutilização temporal e popularidade local permitiu que o algoritmo apresentasse resultados próximos aos melhores observados em cada configuração experimental, sem perdas significativas de desempenho. Os resultados também evidenciaram a influência das distribuições de *Zipf* e *Poisson*, bem como dos mecanismos inspirados em *Chaos Engineering*, sobre o comportamento das políticas de cache, demonstrando a importância da avaliação em cenários que incorporem variações de carga e disponibilidade de recursos.

Como continuidade deste trabalho, novas avaliações podem ser realizadas considerando diferentes tamanhos de cache, distribuições de popularidade com valores mais amplos do parâmetro *Zipf* e cenários com múltiplos níveis de degradação de largura de banda. Também podem ser investigados mecanismos adaptativos para o parâmetro de ponderação $(\alpha)$ do algoritmo híbrido, permitindo ajustar dinamicamente a influência da popularidade local de acordo com o estado do sistema. Por fim, a avaliação em arquiteturas distribuídas, ambientes de *edge caching* e cenários com mobilidade de usuários constitui uma oportunidade para ampliar a análise das políticas propostas e investigar sua aplicabilidade em contextos distintos dos considerados neste estudo.

# REFERÊNCIAS

ALI-ELDIN, A. et al. Analysis and characterization of a video-on-demand service workload. In: **Proceedings of the 6th ACM Multimedia Systems Conference**. New York, NY, USA: Association for Computing Machinery, 2015. (MMSys '15), p. 189–200. ISBN 9781450333511. Disponível em: https://doi.org/10.1145/2713168.2713183.

ALMEIDA, J. M. et al. Analysis of educational media server workloads. In: **Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)**. Port Jefferson, New York, United States of America: ACM, 2001. p. 21–30.

AMANSYAH, H. V.; BANDUNG, Y. Development of distributed multimedia system for ensuring quality of experience (qoe) on video-on- demand (vod) streaming service. **International Conference on Information Technology Systems and Innovation**, Bandung, Indonesia, p. 30–36, 2022.

BASIRI, A. et al. Chaos engineering. **IEEE Software**, v. 33, n. 3, p. 35–41, 2016.

BELADY, L. A. A study of replacement algorithms for a virtual-storage computer. **IBM Systems Journal**, v. 5(2), n. 3, p. 78–101, 1966.

CHEN, S. et al. Segment-based streaming media proxy: modeling and optimization. **IEEE Transactions on Multimedia**, v. 8, n. 2, p. 243–256, 2006.

Cisco. **Cisco Annual Internet Report (2018–2023) White Paper**. 2020. Disponível em: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-pap

CLAEYS, M. et al. An announcement-based caching approach for video-on-demand streaming. In: **2015 11th International Conference on Network and Service Management (CNSM)**. [S.l.: s.n.], 2015. p. 310–317.

CLAEYS, M. et al. Cooperative announcement-based caching for video-on-demand streaming. **IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT**, 2016.

CloudFlare, Inc. **O que é rede de distribuição de conteúdo (CDN)? | Como a CDN funciona?** 2023. Disponível em: Available at <https://www.cloudflare.com/pt-br/learning/cdn/what-is-a-cdn/>.

DAN, A.; SITARAM, D.; SHAHABUDDIN, P. Dynamic batching policies for an on-demand video server. **Multimedia Systems**, Springer, Berlin, v. 4, n. 3, p. 112–121, 1996.

Decoded Magazine. **Netflix eats up 15almost as much as YouTube and TikTok combined**. 2023. Disponível em: https://www.sandvine.com/inthenews/netflix-eats-up-15-of-global-downstream-traffic.

GARCíA, R. et al. Statistical characterization of a real video on demand service: User behaviour and streaming-media workload analysis. **Simulation Modelling Practice and Theory**, v. 15, n. 6, p. 672–689, 2007. ISSN 1569-190X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1569190X07000275.

HARCHOL-BALTER, M. **Performance Modeling and Design of Computer Systems: Queueing Theory in Action**. [S.l.]: Cambridge University Press, 2013.

HONG, D.; VLEESCHAUWER, D. D.; BACCELLI, F. A chunk-based caching algorithm for streaming video. **HAL Open Science**, 2010.

ISHIKAWA, E.; AMORIM, C. Collapsed distrubuted cooperative memory for interactive and scalale media-on-demand systems. **HAL Open Science**, 2009.

KALAN, R. S. Improving video-on-demand performance by multi-channel/multicast approach in cellular networks. In: **2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)**. [S.l.: s.n.], 2017. p. 1–5.

KIM, N. et al. An adaptive ttl allocation scheme for real-time live and on-demand personal broadcasting service. **IEEE 7th International Conference on Consumer Electronics**, 2017.

KOCH, M. H. Avaliando os impactos do uso de variantes para os algoritmos lru e lfu no contexto de sistemas video sob demanda. Bagé, 2022.

KRISHNA, R.; NAYAK, V.; TUMULURU, V. K. Energy-efficient proactive content caching for on-demand video streaming under uncertainty. **IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING**, v. 6, n. 3, p. 1739–1750, 2022.

NEVES, B. S. **Proposta de algoritmo de cacheamento para proxies VoD e sua avaliação usando um novo conjunto de métricas**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2015.

NIKBAKHT, R.; KAHVAZADEH, S.; MANGUES-BAFALLUY, J. Video on demand streaming using rl-based edge caching in 5g networks. **IEEE Conference on Standards for Communications and Networking**, 2022.

PATHAN, A.-S. K.; BUYYA, R.; COMPUTING, D. A taxonomy and survey of content delivery networks. In: . [s.n.], 2006. Disponível em: https://api.semanticscholar.org/CorpusID:943556.

PATTERSON, D. A.; HENNESSY, J. L. **ORGANIZAÇÃO E PROJETO DE COMPUTADORES - A INTERFACE HARDWARE/SOFTWARE**. 3. ed. São Paulo, SP: Elsevier, 2005. 484 p. ISBN 9788535215212.

PATTERSON, D. A.; HENNESSY, J. L. **Computer Architecture: A Quantitative Approach**. 5. ed. [S.l.]: Morgan Kaufmann, 2011. ISBN 9780123838728.

PRODANOV, C. C.; FREITAS, E. C. de. **METODOLOGIA DO TRABALHO CIENTÍFICO: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. Novo Hamburgo, RS: Universidade FEEVALE, 2013. 275 p. ISBN 9788577171583.

RANGANATHAN, P. et al. Warehouse-scale video acceleration: Co-design and deployment in the wild. **26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems**, New York, NY, USA, 2021.

REJAIE, R.; KANGASHARJU, J. Mocha: a quality adaptive multimedia proxy cache for internet streaming. In: **Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video**. New York, NY, USA: Association for Computing Machinery, 2001. (NOSSDAV '01), p. 3–10. ISBN 1581133707. Disponível em: https://doi.org/10.1145/378344.378345.

ROSENTHAL, C.; JONES, N. (Ed.). **Chaos Engineering - System Resiliency in Practice**. Sebastopol, CA 95472: O'Reilly Media, Inc., 2020. v. 1.

SUMMERS, J. et al. Methodologies for generating http streaming video workloads to evaluate web server performance. In: **Proceedings of the 5th Annual International Systems and Storage Conference**. New York, NY, USA: Association for Computing Machinery, 2012. (SYSTOR '12). ISBN 9781450314480. Disponível em: https://doi.org/10.1145/2367589.2367602.

SUMMERS, J. et al. Characterizing the workload of a netflix streaming video server. In: **2016 IEEE International Symposium on Workload Characterization (IISWC)**. [S.l.: s.n.], 2016. p. 1–12.

TU, W. et al. Proxy caching for video-on-demand using flexible starting point selection. **IEEE TRANSACTIONS ON MULTIMEDIA**, v. 11, n. 4, p. 716–729, 2009.

WU, T. et al. Reuse time based caching policy for video streaming. In: **2012 IEEE Consumer Communications and Networking Conference (CCNC)**. [S.l.: s.n.], 2012. p. 89–93.

ZABROVSKIY, A. et al. Fspot: Fast and efficient video encoding workloads over amazon spot instances. **Computers,Materials Continua**, v. 71, n. 3, p. 5677–5696, 2022.