Universidade Estadual de Londrina

GEOVANI PEREIRA DOS SANTOS

PARTICIONAMENTO DINÂMICO ESPAÇO-TEMPORAL PARA SÉRIES DE DADOS MATRICIAIS

LONDRINA

2026

GEOVANI PEREIRA DOS SANTOS

PARTICIONAMENTO DINÂMICO ESPAÇO-TEMPORAL PARA SÉRIES DE DADOS MATRICIAIS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Daniel dos Santos Kaster

LONDRINA
2026

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Santos, Geovani Pereira dos.
Particionamento dinâmico espaço-temporal para séries de dados matriciais / Geovani Pereira dos Santos. - Londrina, 2026.
203 f. : il.

Orientador: Daniel dos Santos Kaster.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2026.
Inclui bibliografia.

1. Particionamento espaço-temporal - Tese. 2. Séries de dados matriciais - Tese. 3. Bancos de dados geoespaciais - Tese. 4. Indexação espaço-temporal - Tese. I. Kaster, Daniel dos Santos. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 519

GEOVANI PEREIRA DOS SANTOS

PARTICIONAMENTO DINÂMICO ESPAÇO-TEMPORAL PARA SÉRIES DE DADOS MATRICIAIS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Daniel dos Santos Kaster
Universidade Estadual de Londrina

Prof. Dr. Lucio Fernandes Dutra Santos
Instituto Federal do Norte de Minas Gerais – IFNMG

Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina – UEL

Londrina, 29 de Abril de 2026.

Dedico este trabalho à minha família, pelo apoio incondicional em cada dia de esforço e luta, e a todos os amigos — os de longa data e os que a pós-graduação me presenteou — que tornaram o caminho mais leve.

AGRADECIMENTOS

Agradeço primeiramente à minha família, por compreenderem minhas ausências e por serem o incentivo necessário para superar cada desafio. Sem o suporte de vocês, esta conquista não seria possível.

Aos meus amigos, tanto os que acompanham minha trajetória há anos quanto os novos parceiros de jornada feitos durante o mestrado, obrigado pelas discussões técnicas, pelas horas de convivio no laboratorio e por compartilharem as dificultades e vitórias da pesquisa. A amizade de vocês foi um dos grandes prêmios deste percurso.

Expresso minha profunda gratidão ao meu orientador, Dr. Daniel Kaster, pela confiança, paciência e pelos ensinamentos que foram fundamentais para o meu crescimento acadêmico e profissional. Agradeço também a todos os professores do programa que contribuíram para a minha formação.

Chegar a este resultado exigiu um esforço, noites de estudo e resiliência diante dos problemas complexos que busquei superar. No entanto, olho para o caminho trilhado com a certeza de que cada obstáculo valeu a pena diante de todas as conquistas e do aprendizado acumulado.

O presente trabalho contou com o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Spectrum Line Prestação de Serviços Eletro-Eletrônicos Ltda. (Espectro Ltda.), por meio da bolsa RHAe, vinculada ao projeto PreCISIA (Predição de Colheita por Imagem de Satélite e Inteligência Artificial), com processo nº 424490/2021 − 8 e bolsa do processo nº 350921/2022 − 9. Agradeço também à Fundação Araucária, pela concessão da bolsa vinculada ao NAPI CIA-Agro.

Por fim, manifesto minha gratidão a UEL e a todos os colegas que, direta ou indiretamente, contribuíram para a realização deste trabalho.

"Os conceitos mais difíceis podem ser
explicados ao homem mais lento,
se ele ainda não os adquiriu;
mas a coisa mais simples não pode ser
esclarecida ao homem mais inteligente,
se ele estiver firmemente convencido de que
já sabe, sem sombra de dúvida."

∞

Liev Tolstói,
O Reino de Deus Está em Vós (1894)

SANTOS, G. P. DOS.. **Particionamento Dinâmico Espaço-Temporal para Séries de Dados Matriciais**. 2026. 203f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2026.

RESUMO

A análise de séries temporais de dados matriciais, como as encontradas em geociências e sensoriamento remoto, frequentemente envolve volumes massivos de dados, impondo desafios significativos de entrada e saída. Nesse cenário, o particionamento de dados é uma técnica para otimizar o desempenho de consultas. Contudo, as abordagens descritas na literatura geralmente aplicam um particionamento estático, seja no domínio espacial ou no temporal, que não se adapta aos padrões de acesso conforme o contexto das consultas. Isso pode resultar em uma grande recuperação de dados sem relevância semântica quando as consultas se concentram em regiões de interesse específicas. Essas regiões consistem em delimitações espaciais definidas por um MBR (do inglês ***M**inimum **B**ounding **R**ectangle*) de área reduzida em relação à matriz completa, cuja posição espacial é fixa e estende-se ao longo de um ou mais períodos consecutivos. Para superar essa limitação, este trabalho propõe um sistema de particionamento dinâmico de séries de dados matriciais. A estratégia consiste em realizar a divisão dos dados no espaço e o agrupamento no tempo para reduzir a dimensão espacial e prolongar temporalmente mantendo a mesma dimensionalidade do bloco, conforme o contexto das consultas sobre as regiões de interesse. Os particionamentos espaço-temporais resultantes são organizados em blocos de armazenamento de tamanho fixo, cujo tamanho pode ser definido para otimizar os acessos à memória secundária. A avaliação experimental, conduzida com dados geoespaciais, demonstrou que, em comparação com abordagens estáticas, a proposta alcançou uma redução de até 87% no tempo de resposta para consultas sobre regiões otimizadas, evidenciando o potencial do particionamento dinâmico e orientado à consulta.

Palavras-chave: particionamento, particionamento dinâmico, particionamento matricial, séries temporais, array databases

SANTOS, G. P. DOS.. **Dynamic Spatiotemporal Chunking for Matrix Data Series**. 2026. 203p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2026.

ABSTRACT

Time series analysis of matrix data, such as that found in geosciences and remote sensing, often involves massive data volumes, imposing significant I/O challenges. In this scenario, data chunking is a technique used to optimize query performance. However, approaches described in the literature generally apply static chunking, either in the spatial or temporal domain, which fails to adapt to access patterns based on the query context. This can result in the retrieval of large amounts of semantically irrelevant data when queries focus on specific regions of interest. These regions consist of spatial boundaries defined by an MBR (**M**inimum **B**ounding **R**ectangle) with a reduced area relative to the full matrix, whose spatial position is fixed and extends across one or more consecutive periods. To overcome this limitation, this work proposes a dynamic partitioning system for matrix data series. The strategy consists of chunking the data in space and grouping it in time to reduce spatial dimensions and extend them temporally—maintaining the same block dimensionality—according to the query context for the regions of interest. The resulting spatiotemporal partitions are organized into fixed-size storage blocks, the size of which can be defined to optimize secondary memory access. Experimental evaluation conducted with geospatial data demonstrated that, compared to static approaches, the proposal achieved a reduction of up to 87% in response time for queries on optimized regions, highlighting the potential of dynamic, query-oriented chunking.

Keywords: Data chunking, dynamic chunking, data tiling, data partitioning, matrix chunking, time series, array databases

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANA	Agência Nacional de Águas
DDR4	*Double Data Rate 4*
GDAL	*Geospatial Data Abstraction Library*
GNU	*GNU's Not Unix*
HDD	*Hard Disk Drive*
HDF5	*Hierarchical Data Format 5*
I/O	*Input/Output*
IT	Instante Temporal
JA	Janela Aleatória
JD	Janela Deslizante
JP2	*JPEG 2000*
LAI	*Leaf Area Index*
LC	*Leaf Cover*
LCC	*Leaf Chlorophyll Content*
LRU	*Least Recently Used*
MBR	*Minimum Bounding Rectangle*
NetCDF	*Network Common Data Form*
RAM	*Random Access Memory*
ROI	*Regions of Interest*
RPM	Rotações por Minuto
SIG	Sistemas de Informação Geográfica
TIF	*Tagged Image File Format*

SUMÁRIO

1 INTRODUÇÃO

A constante evolução das tecnologias de registro de sensoriamento remoto resulta em um aumento considerável no volume de dados matriciais gerados e consumidos, o que permite a realização de análises cada vez mais diversificadas e precisas [1, 2, 3]. Estes tipos de dados são persistidos como séries temporais matriciais, por serem registros bidimensionais de vários espectros eletromagnéticos ao decorrer do tempo [2]. Dados *raster* geoespaciais são uma categoria particular deste tipo de dados, pois são caracterizados pelo armazenamento e análise de informações que abrangem extensas regiões e incluem componentes temporais [4, 5, 6, 7].

Os conjuntos de dados utilizados são volumosos e cobrem extensas áreas, tipicamente muito maiores do que as regiões de interesse (ROIs, do inglês ***R**egions **o**f **I**nterest*) envolvidas em operações espaço-temporais. Assim, o particionamento dos dados no espaço (*tiling* ou *chunking*) é uma estratégia adotada para ajustar o armazenamento dos dados à demanda das consultas [8, 9, 10]. Entretanto, as estratégias de particionamento descritas na literatura trabalham de forma estática apenas no espaço, considerando isoladamente cada instante temporal de uma série, ou de forma localizada no espaço-tempo, fixando uma região espacial e agrupando suas instâncias no tempo. As estratégias de particionamento sobre um instante temporal [11, 12, 13] tratam como o dado vai ser dividido independente dos demais dados da série temporal. Podem existir casos onde métodos *in situ* [9] são aplicados, possibilitando que os dados matriciais possam ter configurações de particionamento estáticos diferentes em uma mesma série temporal. Contudo, esta estratégia não é dinâmica, pois depende de como o arquivo foi codificado e o método de particionamento só pode ser alterado se todo o dado for codificado novamente. Já estratégias que consideram o espaço-tempo [14, 15], tratam uma região em um período pré-fixado como um bloco, ou seja, um tensor tridimensional. Embora tais abordagens sejam eficientes para executar operações sobre a região e o intervalo escolhidos, são limitadas a este caso e penalizam operações realizadas em intervalos ou regiões diferentes.

Diante deste cenário, o presente trabalho propõe uma nova abordagem para particionamento de séries temporais de dados matriciais de forma dinâmica. O dinamismo da proposta é do tipo estrutural, isto é, consiste na otimização restrita a regiões espaço-temporais específicas, sem impactar a estrutura dos dados nas áreas adjacentes. A abordagem particiona os dados e indexa para possibilitar reorganizações espaço-temporais dos blocos sob demanda, visando otimizar o tempo de consulta. Estas alterações ocorrem conforme o contexto das consultas, podendo modificar blocos para armazenar informações de mais de um período temporal, reduzindo o acesso à memória secundária independente da configuração dos blocos de regiões adjacentes. A abordagem proposta é materializada

em um método de acesso que particiona os dados de forma dinâmica, implementando uma estratégia de indexação em três níveis. O primeiro nível é para busca temporal dos dados por meio de uma B^+-tree. O segundo nível é responsável pela indexação espacial dos dados matriciais, com uso de uma estrutura KD-tree com restrições específicas para esta proposta para cada instante de tempo. Por fim, o terceiro nível é uma indexação *in loco*, responsável pela localização do período temporal correto de cada bloco.

1.1 Definição do Problema

O particionamento de séries de dados matriciais é uma atividade usada para a otimização de consultas, especificamente quando estas acessam subconjuntos de dados que representam uma pequena parte destes dados [16, 17, 14, 12, 18, 19, 20, 10].

No contexto de séries de dados matriciais, as estratégias de particionamento dividem em duas ramificações principais, *(i)* particionamento espacial e *(ii)* particionamento espaço-temporal (Seção 2.5). O particionamento Espacial ocorre quando cada instante temporal da série é particionado de forma independente [11, 12, 13, 20, 8, 15]. Os trabalhos que seguem esta abordagem são otimizados para consultas que acessam um único instante temporal, múltiplos instantes não sequenciais ou quando uma ROI varia drasticamente a posição espacial entre os instantes de tempo. No entanto, operações espaço-temporais são frequentes em várias aplicações e ocorrem quando os dados são agrupados em blocos que contêm múltiplos instantes temporais para uma mesma região espacial [14, 21, 15, 22]. Existem também algumas propostas de particionamento espaço-temporal, que visam beneficiar consultas que analisem séries temporais sobre uma ROI espacialmente consistente. Porém, tais propostas são limitadas e uma carga de trabalho bastante específica, pois utilizam blocos grandes, que armazenam um volume de dados significativamente maior do que o demandado pela maioria das operações de cargas de trabalho mais gerais.

O problema central reside no fato de que a escolha entre essas estratégias é definição estrita. Um sistema é configurado para atuar sobre um conjunto de dados apenas um tipo de particionamento, levando a uma compensação de desempenho inevitável ao se aplicar diferentes tipos de consultas. Uma consulta temporal sobre uma ROI fixa em um sistema com particionamento espacial exige a leitura de múltiplos blocos, um para cada instante temporal, resultando em alta sobrecarga de I/O, pois cada bloco pode conter muitos dados fora da ROI. Entretanto, uma consulta sobre um único instante de tempo em um sistema com particionamento espaço-temporal força a recuperação de um bloco que contém múltiplos instantes, resultando na leitura de dados temporalmente irrelevantes para a consulta.

A lacuna é, portanto, a ausência de uma abordagem que possa adaptar a estratégia de particionamento de um subconjunto, para reconciliar a eficiência de ambas as estratégias

em um único sistema conforme o necessário.

1.2 Proposta de Solução

Este trabalho investigou a hipótese de que um particionamento dinâmico, capaz de adaptar a estruturação de regiões delimitadas ao contexto das consultas, promoveu a otimização do desempenho do sistema. Neste contexto, a hipótese central validada por esta pesquisa estabeleceu que:

> ***Hipótese*** *– Uma abordagem de particionamento dinâmico, capaz de reorganizar os dados sob demanda entre a configuração espacial e a espaço-temporal sem alterar a quantidade e tamanho dos blocos de dados, pode reduzir de forma considerável a latência de consultas em diferentes contextos de cargas de trabalho.*

A partir desta hipótese, em vez de selecionar uma única estratégia global, a abordagem proposta baseia-se em permitir que a organização em memória secundária dos dados seja adaptada localmente. A estratégia não consiste em reparticionar os dados em blocos menores, mas sim em reorganizar a disposição dos dados dentro dos blocos existentes.

A abordagem proposta pode ser descrita como uma transformação, dado um conjunto de blocos organizados segundo um particionamento puramente espacial, uma operação de otimização pode ser acionada para uma região de interesse delimitada. Esta operação recupera os dados de múltiplos instantes temporais referentes àquela região e os reescreve nos mesmos blocos (ou em um novo conjunto de blocos de mesmo tamanho), mas agora em uma sequência que localiza em um bloco os dados de uma sub-região espacial ao longo do tempo.

O resultado esperado é a maximização da localidade dos dados para consultas temporais, agrupando a informação de interesse em um número reduzido de blocos. Isso visa reduzir a quantidade de operações de recuperação de dados da memória secundária (I/O), um conhecido gargalo de desempenho neste contexto. Formalmente, a expectativa é que o tempo de consulta ($T_{consulta}$) seja minimizado ao reduzir o número de blocos acessados (N_{blocos}), visto que o tempo de recuperação dos blocos ($T_{I/O}$) gera o maior impacto:

$$T_{consulta} \propto N_{blocos} \times T_{I/O}$$

A reorganização dinâmica dos dados visa, portanto, minimizar N_{blocos} para consultas mais frequentes ou onerosas, conforme o padrão de consulta considerado na análise.

A dinamicidade da proposta é obtida por meio do que este trabalho chama de operações de *split* e *unsplit* espaço-temporais. Essas operações são inspiradas nas operações de organização de índices dinâmicos, como a B^+-tree e R-tree, mas estendidas para agrupamentos espaço-temporais de dados. Isto é, um *split* espaço-temporal divide uma região espacial armazenada em um bloco para agrupar dados de cada uma das sub-regiões obtidas de múltiplos instantes de tempo no mesmo bloco. Assim, consultas que tenham ROIs menores sobre um intervalo de tempo podem recuperar os dados necessários com menos acessos à memória secundária, por meio de uma arquitetura guiada por um particionamento dinâmico, com índices e gerenciamento de memória adequados.

A premissa é que muitas aplicações concentram-se em regiões reduzidas, por exemplo, um município, uma unidade de conservação, uma propriedade rural ou mesmo um talhão, mas demandam realizar análises sobre séries de dados. Por exemplo, na área da agricultura, o comportamento de uma variável no tempo, como temperatura, umidade e pluviometria, tem grande impacto no desempenho de uma safra. Portanto, análises espaço-temporais são frequentes para realizar estimativas e modelos preditivos para uma ROI.

1.3 Objetivo geral

O objetivo geral é apresentar uma abordagem de particionamento dinâmico para séries de dados matriciais, visando reduzir o tempo de recuperação de dados de interesse. A solução utiliza um gerenciador de indexação hierárquica de três níveis e gerenciamento de particionamento dos blocos, que possibilita o acesso eficiente à região espaço-temporal de interesse.

Dentre estes, os objetivos específicos são:

- Propor uma metodologia de indexação hierárquica de três níveis;
- Elaborar uma abordagem de particionamento espacial que utilize divisões equivalentes, dada uma região espacial e um nível de partição definido;
- Estabelecer diretrizes e normas para o particionamento de dados espaço-temporais;
- Projetar uma arquitetura de software para a implementação da solução proposta;
- Realizar experimentos quantitativos para validar a hipótese de redução no tempo de recuperação de dados com o método desenvolvido;
- Analisar os resultados obtidos para avaliar a viabilidade e o desempenho da solução em domínios de aplicação de séries de dados matriciais.

1.4 Organização do texto

Além deste capítulo introdutório, o restante desta dissertação foi organizado da seguinte forma: o Capítulo 2 aborda a fundamentação teórica sobre a temática desta proposta e as abordagens relacionadas ao particionamento de dados matriciais. O Capítulo 3 descreve a abordagem proposta para particionamento dinâmico de séries de dados matriciais. O Capítulo 4 apresenta os experimentos realizados, incluindo as cargas de trabalho, metodologia de testes para avaliação da abordagem proposta e os resultados obtidos. Por fim, o Capítulo 5 apresenta uma conclusão desta dissertação.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Este capítulo apresenta os conceitos para a compreensão das estratégias de particionamento matriciais discutidas neste trabalho. A análise se inicia com uma visão sobre o modelo de sistemas que atuam sobre dados vetoriais utilizados para processar este tipo de dados [16]. Em seguida, são discutidos os princípios de estruturação de dados e as estruturas de indexação, que em conjunto asseguram uma organização lógica e um acesso rápido a partições específicas. Com essa base técnica estabelecida, a discussão avança para o domínio dos dados geoespaciais. Este tipo de dado, que se beneficia de estratégias de particionamento eficientes [23, 24, 6], serve como principal caso de uso e motivação. Dentro deste contexto, o presente estudo enfatiza o modelo de dados *raster*, por ser o formato predominante para a representação de superfícies contínuas e o objeto das otimizações propostas.

2.1 Dados Matriciais

Em diversas áreas têm-se constatado um aumento considerável no volume de dados matriciais gerados e consumidos, incluindo dados matriciais espaciais, dados científicos e outros. A disponibilidade de dados e o seu processamento eficiente permitem a realização de análises cada vez mais diversificadas e precisas [1, 2]. Frequentemente, estes tipos de dados são persistidos como séries temporais matriciais, por serem registros multidimensionais de variáveis e medidas ao decorrer do tempo.

Matematicamente, uma matriz é um objeto bidimensional. Contudo, é comum que tensores de ordem superior sejam denominados *matrizes N-dimensionais*, o que pode gerar ambiguidades. Para evitar conflitos, neste texto, é adotada a convenção matemática: o termo matriz se restringe a objetos bidimensionais, enquanto para dimensões superiores é utilizado o termo *tensor N-dimensional*.

Apesar de existirem abordagens que lidam com o gerenciamento deste tipo de dado [15, 9, 20], ainda existem desafios relacionados ao gerenciamento e processamento que podem surgir como alternativas mais eficientes. Este trabalho concentra-se no aspecto do particionamento de dados matriciais, para acesso otimizado de dados no processamento de operações espaço-temporais.

2.1.1 Dados Espaciais

Os dados espaciais podem ser classificados em duas categorias: vetoriais e matriciais [25, 26]. O modelo vetorial representa elementos por meio de pontos, linhas e polígonos,

sendo adequado para mapear variáveis discretas com limites definidos, possibilitando a delimitação geométrica de estruturas como edificações e hidrografias. Entretanto, essa representação apresenta limitações na modelagem de superfícies contínuas. Em contrapartida, os dados matriciais estruturam-se como tensores densos, tipicamente geradas por instrumentos de sensoriamento ou simulações. Esse formato, como imagens de satélite e fotografias aéreas, é destinado à representação de variações espaciais contínuas, suprindo a restrição da abordagem vetorial.

Um dado geográfico ou geoespacial é um tipo de dado espacial cuja dimensão espacial está associada à sua localização na superfície terrestre, em determinado instante ou período [27, 28]. Com a popularização de satélites e outras tecnologias, os dados geoespaciais podem ser obtidos de diversas fontes. É importante salientar que, para cada finalidade, existe uma precisão necessária que define a escala e, consequentemente, o método e a tecnologia a serem utilizados [28].

O uso de satélites e outros mecanismos de sensoriamento remoto tem se mostrado relevante no monitoramento de diversas culturas agrícolas, tanto espacial quanto temporalmente [6]. Um exemplo proeminente é o projeto Sentinel, desenvolvido pela Agência Espacial Europeia (ESA, do inglês ***E****uropean* ***S****pace* ***A****gency*), que fornece dados geoespaciais de satélite abertamente. O projeto consiste em uma série de missões, atualmente 5 em operação, cada uma com objetivos específicos, como fornecer dados para mapeamento de cobertura e mudança do uso da terra, e apoiar a avaliação de parâmetros biofísicos como Índice de Área Foliar (LAI, do inglês ***L****eaf* ***A****rea* ***I****ndex*), Teor de Clorofila Foliar (LCC, do inglês ***L****eaf* ***C****olor* ***C****hart*) e Cobertura Foliar (LC, do inglês ***L****eaf* ***C****overage*) [29].

2.1.2 Dados Matriciais Espaciais ou *Raster*

Os dados provenientes de sensoriamento remoto, como os do projeto Sentinel, são predominantemente estruturados segundo o modelo de dados matricial *raster*. Este modelo representa o espaço geográfico como uma grade de células, onde cada célula contém um valor que quantifica uma característica daquela localidade, como temperatura, elevação ou reflectância espectral.

O *raster* é um formato específico de imagem digital, normalmente associado a dados geoespaciais, estruturado como uma matriz bidimensional [30] em que cada elemento possui localização e valor definidos. Cada coordenada da matriz de um determinado *raster* contém a representação de um *pixel*, abreviação do termo em inglês *picture element*, que é o termo mais usado para aos elementos utilizados para representar um *raster* [31]. Assim, o valor de cada *pixel* é proporcional à intensidade de luminosidade do ponto correspondente na cena [30]. Contudo, diferentemente das imagens tradicionais, como o RGB, os valores em um dado *raster* não se limitam ao intervalo de 8 bits ($[0, 255]$) e, normalmente, são registrados vários *rasters* de uma mesma área, onde um *raster* é composto por uma única

camada, que representa uma medição distinta para a mesma área.

A aquisição de um *raster* requer o emprego de dois dispositivos: *(i)* um dispositivo físico sensível à energia irradiada pelo objeto na cena em que ser deseja efetuar a captura; e *(ii)* um dispositivo digitalizador, responsável por converter a saída do dispositivo físico em formato digital [31].

Uma das formas de se gerar um *raster* é através da iteração de uma fonte de emissora de radiação eletromagnética e os objetos presentes em uma cena. Esses objetos podem refletir, transmitir e/ou absorver essa radiação [31]. Esta radiação é absorvida por sensores constituídos de um material fotossensível, convertida em energia elétrica. Assim, cada *pixel* do *raster* é o resultado da discretização da onda de tensão de saída do sensor [31, 30]. A resolução espacial final do dado depende diretamente das especificações do sensor de captura. Em plataformas satelitais, um único instrumento pode gerar imagens com múltiplas resoluções, uma vez que diferentes bandas do espectro eletromagnético exigem áreas de captação distintas para assegurar uma relação sinal-ruído adequada [29].

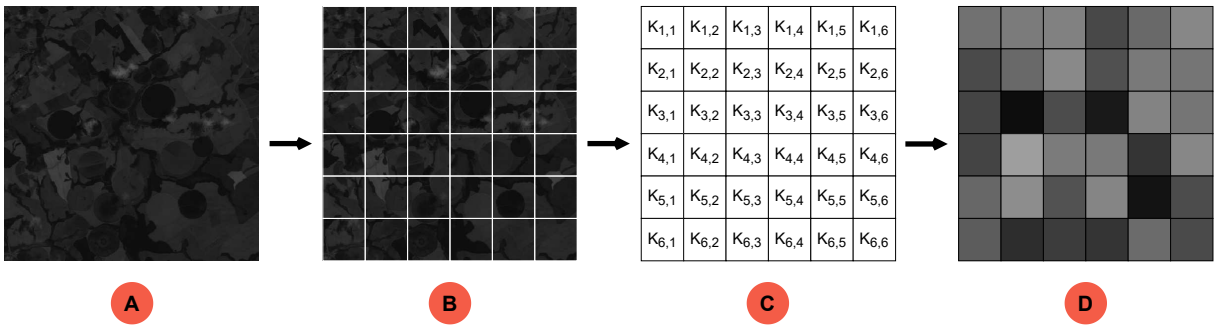

Figura 1 – Processo simplificado de aquisição e discretização de uma cena.

A Figura 1 representa um processo de captura de um *raster*. A Figura 1 **A** é a representação da cena que se deseja adquirir, a Figura 1 **B** representa a cena dividida em regiões que simbolizam a área que um sensor vai capturar a energia daquela região na totalidade. A Figura 1 **C** representa a discretização do sinal obtido pelo sensor referente à região correspondente na cena a ser adquirida. Por fim, a Figura 1 **D** é uma representação do *raster* resultante de todo o processo de aquisição.

Rasters provenientes de dados geoespaciais costumam gerar uma quantidade massiva de dados. Portanto, para lidar com esses dados de modo eficiente é necessário o uso ferramentas ou metodologias projetadas especificamente para isso, incluindo o particionamento e a indexação de dados.

2.2 Operações sobre Séries de Dados Matriciais

Os dados geoespaciais são amplamente utilizados em diversos domínios de aplicação, fato que motivou o desenvolvimento da Álgebra de Mapas. Esta é uma extensão da álgebra convencional para operar sobre dados matriciais multidimensionais, ou *rasters* [32]. As operações têm como entrada um ou mais *rasters* e incluem operadores matemáticos elementares sobre células individuais ou vizinhanças [33]. Existem três classes principais de operações: *(i)* local, *(ii)* zonal e *(iii)* focal. Cada operação processa um ou mais *rasters* de entrada para gerar um novo raster de saída, cujo conteúdo é determinado pela função aplicada [34, 35, 36].

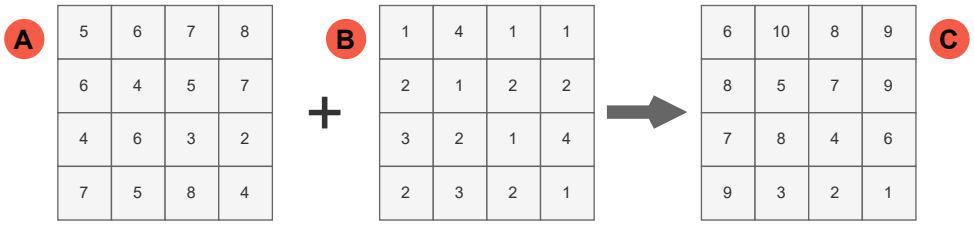

Figura 2 – Exemplo de operação local de soma.

A Figura 2 ilustra a operação local de soma, que utiliza dois rasters de entrada (A e B) e gera um raster de saída (C). A operação é executada célula a célula, aplicando-se o operador definido sobre as células em posições correspondentes. Assim, para um operador de soma, a matriz resultante é calculada como $C_{i,j} = A_{i,j} + B_{i,j}, \forall (i,j) \in A \cap B$. Esta classe de operação não se limita à soma, sendo aplicável para calcular diferenças, produtos, razões, raízes, funções trigonométricas e estatísticas como média, mediana, máximo e mínimo.

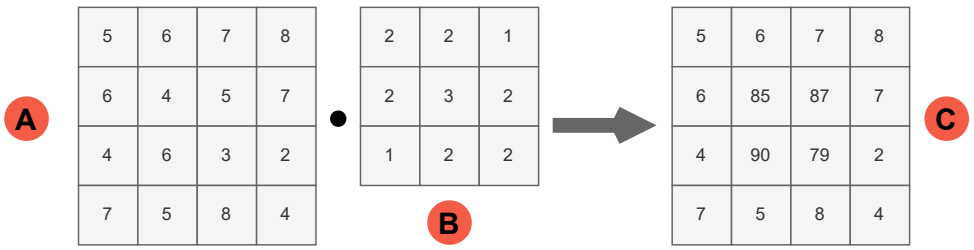

Figura 3 – Exemplo simplificado de operação focal com máscara 3×3.

A Figura 3 exibe uma operação focal utilizando um kernel (ou máscara) de vizinhança de 3×3. Neste exemplo, adota-se a estratégia de manter o valor original da célula caso sua vizinhança não seja completa (células de borda conforme o tamanho da máscara). A operação requer um raster de entrada (A) e uma máscara (B), que tipicamente possui dimensões ímpares. O valor de cada célula $C_{i,j}$ no *raster* de saída é resultado da

multiplicação, valor a valor, dos elementos da máscara pelos valores correspondentes na vizinhança da célula $A_{i,j}$, seguida pela soma de todos os produtos. Outros métodos de operação podem ser aplicadas na janela de vizinhança, como estatísticas, identificação de grupos contíguos e magnitude relativa.

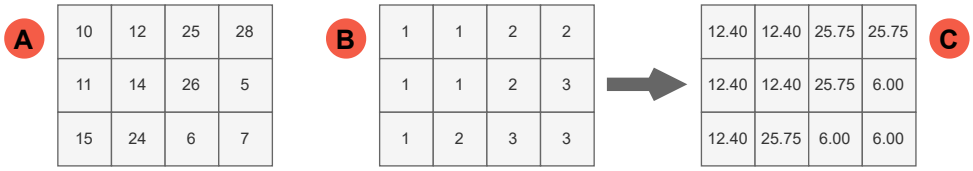

Figura 4 – Exemplo simplificado de operação zonal.

Por fim, a Figura 4 ilustra um exemplo de aplicação do operador zonal. Esta requer dois *rasters* de entrada: um *raster* de valores (A), como dados de elevação, e um *raster* de zonas (B), que agrupa as células em categorias, como uso do solo. A operação calcula uma estatística para os valores em A que estão contidos em cada zona definida em B. O resultado é um *raster* de saída (C), onde todas as células pertencentes a uma mesma zona recebem o valor único calculado. Neste exemplo, para cada zona em B, calcula-se a média dos valores de elevação correspondentes em A. Outras funções zonais incluem o cálculo de somas, produtos, contagem de valores distintos e outras estatísticas por zona.

Deste modo, assim como na álgebra convencional, os operadores de mapas podem ser combinados para construir expressões e modelos complexos. A capacidade de aplicar operações de forma iterativa, utilizando a saída de uma como entrada para a seguinte, confere uma elevada capacidade analítica e flexibilidade para a modelagem geoespacial.

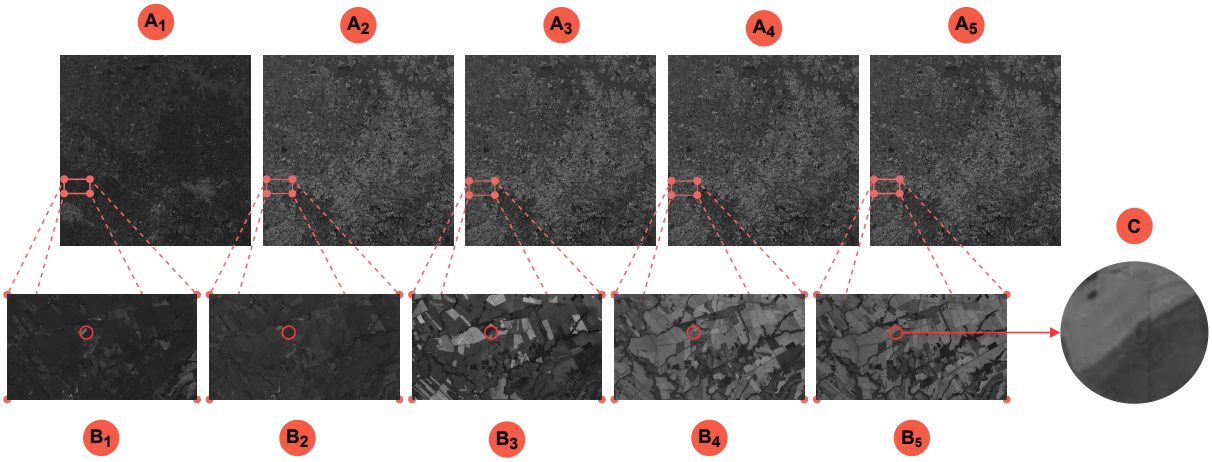

Figura 5 – Exemplo de janela espacial em uma consulta espaço-temporal.

Na prática, a aplicação destes operadores é frequentemente restringida a janelas espaciais, que correspondem a uma sub-região geográfica que delimita o escopo de uma

consulta ou análise de dados [37]. Ao se trabalhar com séries temporais de *rasters*, este conceito se estende para uma janela espaço-temporal, que representa a mesma ROI observada ao longo de um intervalo de tempo definido.

As consultas espaço-temporais exemplificam o uso prático deste método, no qual se recuperam dados de uma área e período específicos para análise. A Figura 5 ilustra um exemplo de uma janela sobre uma série temporal de imagens de satélite, composta por cinco períodos subsequentes ($[A_1, A_5]$). Devido à grande dimensionalidade dos dados originais, recortes de cada *raster* ($[B_1, B_5]$) são utilizados para destacar a ROI, demarcada com um círculo em vermelho. Esta região, ampliada em Figura 5C no instante de tempo 5, constitui a janela espacial que, ao ser aplicada sequencialmente sobre os *rasters*, forma a janela espaço-temporal (neste caso, do período 1 ao 5). A análise restrita a estas janelas permite aplicar os operadores de Álgebra de mapas de forma eficiente e focada em áreas de relevância semântica, que frequentemente representam uma fração mínima da área total dos dados. Portanto, realizar um procedimento prévio de particionamento destes dados pode ser benéfico para a redução do tempo para recuperação dos dados que estão dentro das janelas espaciais.

2.3 Particionamento de Dados Matriciais

Uma estratégia para agilizar o processamento de dados matriciais é o particionamento, que decompõe um tensor lógico em um conjunto de blocos fisicamente independentes. Assim, o tempo de acesso é otimizado, pois uma consulta necessita recuperar da memória secundária apenas os blocos que intersectam a área de interesse. Bancos de dados espaciais que lidam com dados *raster* e bancos de dados de *arrays* normalmente fazem uso desta estratégia. Bancos de dados de *arrays* (*array databases*) são sistemas especializados para o gerenciamento de dados tensores n-dimensionais, tratando-os como objetos nativos [16].

A eficácia do particionamento depende da sua geometria, que pode seguir diferentes padrões [21]. A Figura 6 apresenta os quatro tipos de particionamentos espaciais empregados em bancos de dados *array*, cada um com características distintas que influenciam sua aplicação, em harmonia com os objetivos específicos de cada sistema. A Figura 6 A serve como base para a descrição dos demais casos, ilustrando uma matriz sem qualquer forma de particionamento prévia.

Os dois primeiros tipos de particionamento são classificados como regulares, caracterizados pela aplicação de linhas que intersectam o plano em direções verticais e horizontais. A Figura 6 B demonstra o **particionamento regular**, método amplamente utilizado devido à sua natureza de divisão uniforme do espaço [21]. Este método apresenta padrões de particionamento repetitivos tanto na direção vertical quanto na horizontal. A Figura 6 C ilustra o **particionamento irregular alinhado**, distinguindo-se pela ausência

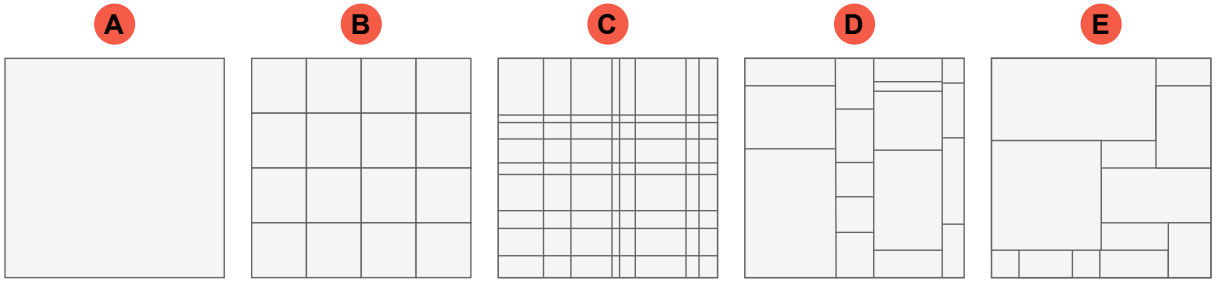

Figura 6 – Estratégias de particionamento espacial.

de padrões de divisão uniformes [21]. Cada faixa de orientação de particionamento é definida por propriedades uniformes em um eixo (altura ou largura) e por outra faixa com características distintas.

Os dois tipos finais de particionamentos espaciais são definidos como desalinhados, onde a matriz não é dividida por linhas retas que cortam o plano em pelo menos um dos eixos. A Figura 6 D apresenta o **particionamento parcialmente desalinhado**, no qual um dos eixos é segmentado por linhas que percorrem todo o plano, enquanto o outro eixo é dividido por segmentos de reta que não abrangem a totalidade do plano [21]. Essa abordagem permite maior flexibilidade na divisão de um dos eixos (no sentido das retas que cortam o plano), possibilitando a adaptação ao contexto dos dados em análise. Contudo, pode resultar em métodos de particionamento e manutenção mais complexos. A Figura 6 E ilustra o **particionamento totalmente desalinhado**, onde ambos os eixos são divididos por segmentos de reta que não intersectam completamente o plano [21]. Neste cenário, a partição dos dados é realizada de forma livre, adaptando-se às necessidades específicas do contexto de análise. Apesar da flexibilidade proporcionada, essa estratégia pode demandar atualizações frequentes na matriz, com destaque para sistemas com múltiplos contextos de análise, o que pode gerar custos significativos, uma vez que a modificação de um particionamento pode desencadear alterações em outros.

Além da estratégia de particionamento, o tamanho dos blocos de armazenamento é outro fator que impacta diretamente o desempenho e representa um *trade-off* relevante. Os blocos de grande dimensão minimizam a contagem de operações de I/O para consultas que cobrem grandes áreas, mas causam uma sobrecarga de leitura significativa para consultas pequenas e pontuais. Inversamente, blocos pequenos são ideais para acessos granulares, mas podem resultar em um número excessivo de acessos à memória secundária para recuperar uma região ampla, degradando o desempenho.

Portanto, a configuração ótima de particionamento não é universal, exigindo adequação aos padrões de consulta e à natureza da carga de trabalho. Especificamente para operações espaço-temporais sobre séries de dados matriciais, conforme abordado na Seção 2.5, existe uma lacuna na literatura quanto a soluções eficientes para essa adaptação.

Para que o particionamento seja efetivo, é efetuada a associação de estruturas de indexação que possibilitem o acesso localizado a partições. Tais estruturas são determinantes para a eficiência, pois se alinham à organização espacial dos dados e aumentam a velocidade de acesso [10, 13].

2.4 Estruturas de indexação

Esta seção descreve métodos que podem ser utilizados como forma de indexação de dados. Neste caso, a indexação é voltada para a otimização de consultas em grandes volumes de dados, particularmente em cenários onde o acesso sequencial se torna computacionalmente proibitivo. A técnica consiste em criar e manter uma estrutura de dados auxiliar que mapeia os valores de um ou mais atributos (chave de busca) para a localização física dos registros correspondentes.

A implementação de um índice representa um compromisso clássico em sistemas de dados. O custo de um índice manifesta-se como uma sobrecarga de armazenamento para a própria estrutura do índice e uma sobrecarga computacional para sua manutenção durante operações de modificação (inserção, remoção e atualização). O propósito é minimizar o número de operações de I/O e reduzir a complexidade da busca, permitindo que um subconjunto de dados seja localizado de forma mais eficiente do que mediante uma varredura completa.

A seguir, serão detalhadas diferentes estratégias de indexação, incluindo abordagens clássicas para dados unidimensionais (B^+ tree e *hashing*) e multidimensionais (R-tree, KD-tree, Quadtree e Octree). No contexto de operações espaço-temporais, estruturas unidimensionais são úteis para a indexação do dado temporal enquanto índices multidimensionais tratam da organização espacial dos dados.

2.4.1 Índices Unidimensionais

2.4.1.1 Árvore B^+

A Árvore B^+ (B^+ tree) é uma variação aprimorada da Árvore B proposta por Bayer e McCreight [38], com o objetivo principal de melhorar o acesso sequencial e a eficiência em sistemas de armazenamentos [39]. Dessa forma, trata-se de uma estrutura normalmente encontrada em sistemas de gerenciamento de dados [40]. A B^+ tree é uma árvore do tipo N-ária, o que significa que cada nó contém até N elementos. Diferentemente da B tree tradicional, ela armazena somente chaves nos nós internos, enquanto os registros completos (chave e valor) ficam restritos aos nós folhas [39, 40]. Essa distinção permite mais ramificação, o que reduz significativamente a altura da árvore e melhora o desempenho nas operações de busca e inserção [39].

Os nós folha são interligados sequencialmente, possibilitando buscas ordenadas e varreduras contínuas sem a necessidade de acessar repetidamente os níveis superiores da árvore. Quando um nó atinge sua capacidade máxima, ocorre um *split*, em que o nó é dividido em dois e a chave mediana é promovida ao nó interno imediatamente acima, garantindo que cada nó mantenha pelo menos metade de sua capacidade. Trivialmente, esse caso não se aplica aos casos onde a quantidade de elementos inseridos na árvore é menor que a metade dos itens que cabem em nó. Nestes casos, a árvore vai ter um único nó folha com todos os itens [39, 40]. O processo de *split* e o seu processo oposto *unsplit* assegura que a estrutura mantenha-se balanceada e com manutenção localizada a cada operação de inserção ou remoção, resultando em operações de custo logarítmico com relação à quantidade de entradas indexadas.

A B^+ tree é notadamente vantajosa para sistemas que manipulam grandes volumes de dados, ao permitir o armazenamento parcial em memória primária enquanto o restante é mantido em memória secundária. Essa abordagem eficiente otimiza o uso dos recursos computacionais e assegura um desempenho estável mesmo em cenários com alta demanda de inserções e remoções [40].

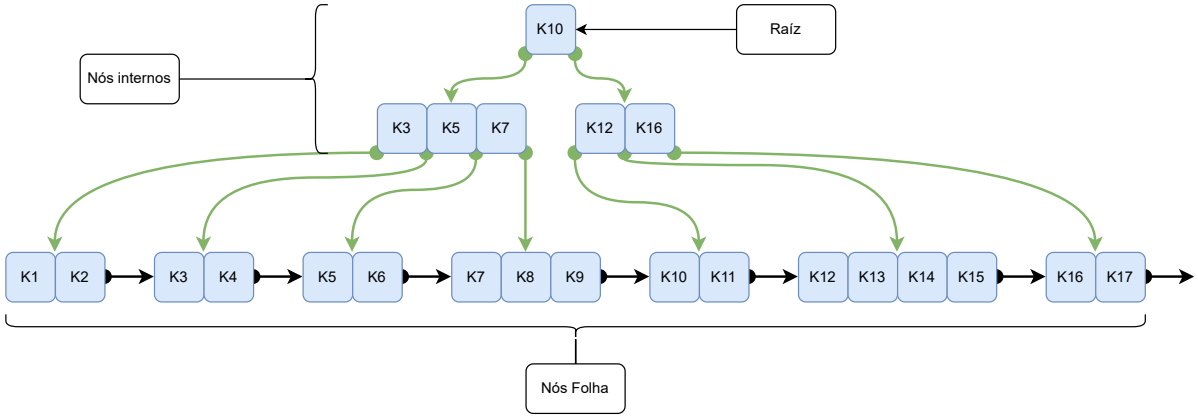

Figura 7 – Estrutura de uma B^+ tree.

A Figura 7 representa a estrutura de uma B^+ tree que indexa valores usando 17 chaves ($K_1 - K_{17}$). Cada nó tem capacidade de até 4 registros, onde o tamanho mínimo é de 2 registros, exceto para o nó raiz, que pode ter de 1 a 4 registros. Como pode se perceber, algumas chaves que aparecem em nós folha também aparecem nos nós internos, devido ao processo de promoção. Nós internos têm até N valores e $N + 1$ ponteiros para os nós para o próximo nível da árvore. Para cada nó folha, existe até N chaves, N valores e 1 ponteiro para o próximo nó, que está em ordenado conforme as chaves (dois ponteiros se as folhas estiverem em uma lista duplamente encadeada). Os valores podem armazenar objetos genéricos, como um valor, uma tupla, ou um ponteiro para um binário em memória secundária.

2.4.1.2 Índices *Hash*

A técnica de *hash* possui ampla aplicação em sistemas computacionais, desde operações que abrangem autenticação de mensagens e a identificação de arquivos duplicados até o processamento analítico de grandes volumes de dados. A capacidade de criar representações compactas dos dados permite otimizar o armazenamento e a recuperação de informações, além de fornecer uma forma robusta de verificar a integridade dos dados, detectando alterações mesmo que mínimas [41]. A complexidade computacional associada ao hash é frequentemente subestimada. Embora a criação de um possa parecer simples, a implementação eficiente de algoritmos de hash, sobretudo os utilizados em contextos de segurança, envolve considerações significativas. A eficiência teórica das estruturas de indexação baseadas em *hash* contrasta com os desafios práticos de sua implementação. Para garantir um custo de busca próximo a $\mathcal{O}(1)$, o método exige funções que assegurem a distribuição uniforme das chaves e estratégias eficientes para a resolução de colisões. Deste, modo estes fatores são necessários para evitar a degradação estrutural e minimizar os acessos à memória secundária conforme o volume de dados aumenta.

Um índice *hash* aplica uma função de espalhamento à chave de busca para computar diretamente o endereço físico ou lógico (*bucket*) em que o registro correspondente está armazenado [41]. Embora a noção de *hash* possa ser aplicada para indexar dados de diferentes tipos, inclusive dados espaciais e multidimensionais, seu uso mais trivial é para indexar dados unidimensionais.

Um índice *hash* aplica uma função de espalhamento à chave de busca para computar diretamente o endereço físico ou lógico (*bucket*) do registro correspondente [41]. Por sua natureza arquitetural, essa estrutura apresenta alta eficiência em buscas exatas (por igualdade), sendo ineficaz para consultas de intervalo. Embora o método possa ser aplicado para dados espaciais e multidimensionais, sua aplicação normalmente ocorre na indexação de atributos unidimensionais.

Como ilustrado na Figura 8, a chave de cada registro ou documento é transformada em um valor *hash* correspondente. Esse *hash* serve então como chave para acessar o registro ou documento por meio do índice, permitindo uma busca eficiente, muito superior à busca linear mediante todos os registros do banco de dados. A comparação é uma operação muito rápida, geralmente realizada via tabelas que permitem acesso direto aos dados correspondentes ao *hash* encontrado. Em vez de comparar cada elemento individualmente, o sistema utiliza a velocidade da tabela *hash* para localizar rapidamente o item desejado.

2.4.2 Índices Multidimensionais

Existem diversos índices para dados espaciais e multidimensionais na literatura. Um dos mais importantes é a R-tree [42], que tem implementações em inúmeros sistemas

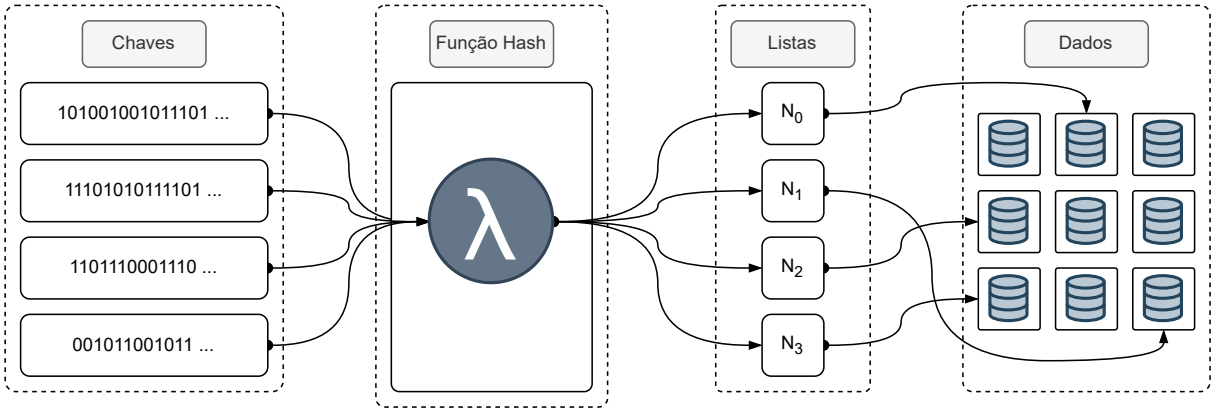

Figura 8 – Representação gráfica de uma estrutura de dados Hash.

de bancos de dados espaciais e aplicações. A estrutura da R-tree é hierárquica, onde cada nó interno armazena um retângulo delimitador mínimo (MBR, do inglês ***M****inimum* ***B****ounding* ***R****ectangle*) que encapsula todos os MBRs de seus nós filhos. MBR é um retângulo com menor área possível dentre todos os retângulos que encapsulam um objeto. Essa organização particiona o espaço de maneira eficaz, por meio de *splits* e *unsplits* inspirados na B-tree, mas agrupando objetos espacialmente próximos nos mesmos nós. O intuito é reduzir a área de busca em consultas, embora possa haver sobreposição entre as regiões cobertas por nós diferentes.

A Figura 9 ilustra o funcionamento da estrutura. A Figura 9A exibe um exemplo de particionamento de um espaço bidimensional, onde o retângulo raiz $R1$ contém os retângulos $[R2, R3]$. Por sua vez, estes contêm os retângulos folha, que neste exemplo são os objetos $[R4, R8]$. A Figura 9B ilustra a estrutura de árvore derivada deste particionamento, com $R1$ como nó raiz, e o retângulos $[R4, R8]$ como os nós folhas. As chaves K_i são as coordenadas que definem os MBRs correspondentes aos nós.

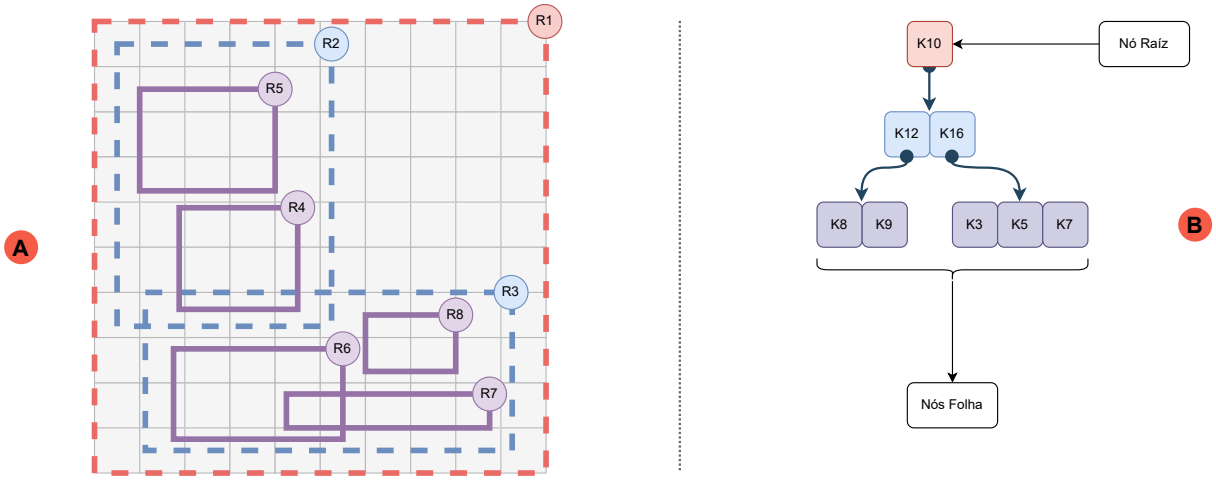

Figura 9 – Representação gráfica de uma estrutura de dados R-tree.

Embora seja amplamente usada para indexação de dados espaciais, a R-tree não é adequada para uso alinhado a estratégias de particionamento de dados, pelo fato de haver sobreposição entre os nós, o que implicaria em replicação de dados em sobreposição nos blocos de armazenamento correspondentes, o que não é vantajoso. Para este tipo de uso, índices multidimensionais sem sobreposição são mais adequados, como a KD-tree, Quadtree e Octree.

2.4.2.1 KD-Tree

A KD-tree é uma árvore de busca binária multidimensional, inicialmente proposta por Bentley [43], que pode indexar dados que podem ser recuperados por meio de buscas associativas. Uma busca associativa é um método de recuperação de dados onde os registros são localizados com base em múltiplas chaves ou atributos. Esta abordagem se caracteriza por dividir o espaço de busca multidimensional em regiões menores por uma estrutura de árvore [44]. Cada nó da árvore representa um ponto em k-dimensional, e as arestas de divisão são construídas ao longo de cada dimensão em alternância ortogonal (vertical seguida de horizontal seguida de vertical, e assim por diante) e sem sobreposição. Deste modo, é criada uma organização hierárquica que permite a realização de buscas eficientes. A inserção de novos registros é realizada através da localização da região referente as informações do registro, navegando pela árvore a partir da raiz até encontrar uma posição adequada [43, 45].

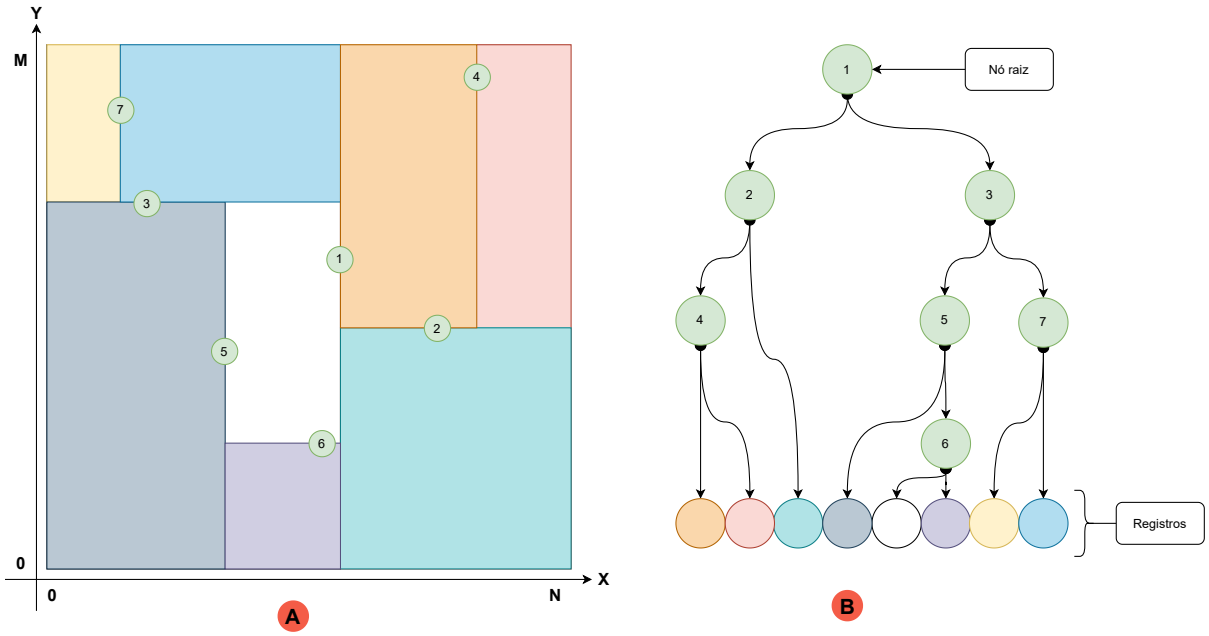

Figura 10 – Representação gráfica de uma árvore KD-tree.

A Figura 10 mostra a representação gráfica de uma instância de uma KD-tree. A Figura 10 (A) é a divisão multidimensional nos eixos X e Y, um ponto definido no

eixo X pode assumir valores entre 0 a N e no eixo Y pode assumir valores entre 0 e M. Os círculos verdes (1-7) representam os pontos onde foi requisitado o particionamento ortogonal. Neste caso, o primeiro particionamento é paralelo ao eixo Y. A Figura 10(B) representa a estruturação hierárquica da KD-tree conforme o particionamento realizado em Figura 10(A). O primeiro nível é o primeiro particionamento requisitado e o último nível representa abstratamente os registros de que cada área pode conter, que podem ser pontos ou objetos contidos na área, ou os *pixels* que a compõem.

2.4.2.2 Quadtree

A Quadtree é uma estrutura de dados normalmente utilizada para indexação espacial, mas não se restringindo a isto, sendo utilizada em outras aplicações, como compactação de dados e processamento de imagens. Tem como principio básico, dividir uma área bidimensional, normalmente uma matriz, em quatro partes iguais, dividindo ao meio, tanto horizontalmente quanto verticalmente. Cada nó interno possui quatro nós filhos, onde cada filho corresponde a uma $\frac{1}{4}$ da divisão homogênea do nó pai [45, 46] (*NW*, *NE*, *SW* e *SE* na Figura 11). Esta abordagem permite representar dados complexos de maneira eficiente, como *rasters*, ao decompor o espaço em níveis progressivos de detalhe [45].

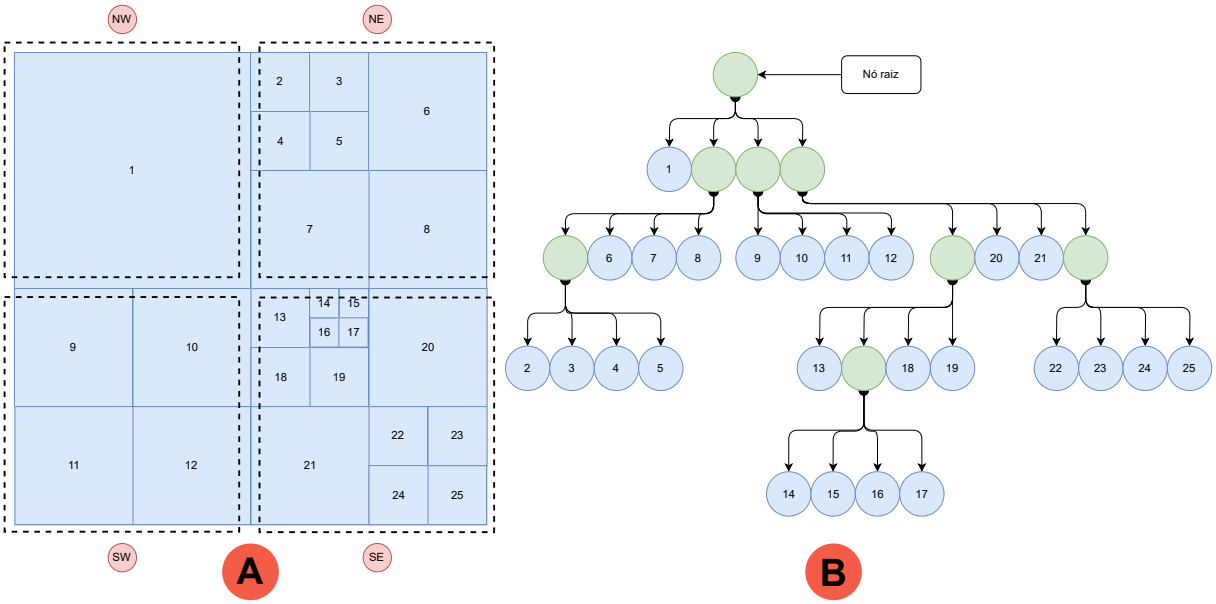

Figura 11 – Representação gráfica de uma Quadtree.

O conceito central da Quadtree está em sua capacidade de acomodar variações significativas na densidade dos dados. Ao dividir recursivamente uma região, a árvore se adapta à distribuição espacial dos elementos, concentrando recursos computacionais nas áreas com maior concentração e simplificando as representações em regiões menos densas [46]. Essa flexibilidade torna a Quadtree particularmente adequada para lidar com dados irregulares e não uniformemente distribuídos.

2.4.2.3 Octree

A Octree é uma extensão da Quadtree empregando uma estrutura hierárquica tridimensional para organizar dados espaciais [46]. De maneira análoga à Quadtree, a Octree divide recursivamente uma região, mas agora a região é um volume tridimensional subdividido em oito sub volumes iguais (Figura 12A). Cada nó indexa os oito sub volumes cobertos pelo seu volume (Figura 12B). Essa representação permite buscas espaciais rápidas e eficientes, pois o algoritmo pode acessar diretamente para a região relevante da Octree evitando a necessidade de examinar todos os dados do volume, o que a torna uma estrutura valiosa em indexação tridimensional sem sobreposição entre nós. Em se tratando de dados espaço-temporais, a noção da Octree poderia ser usada para representar dois *rasters* subsequentes, considerando a dimensão adicional como sendo o tempo.

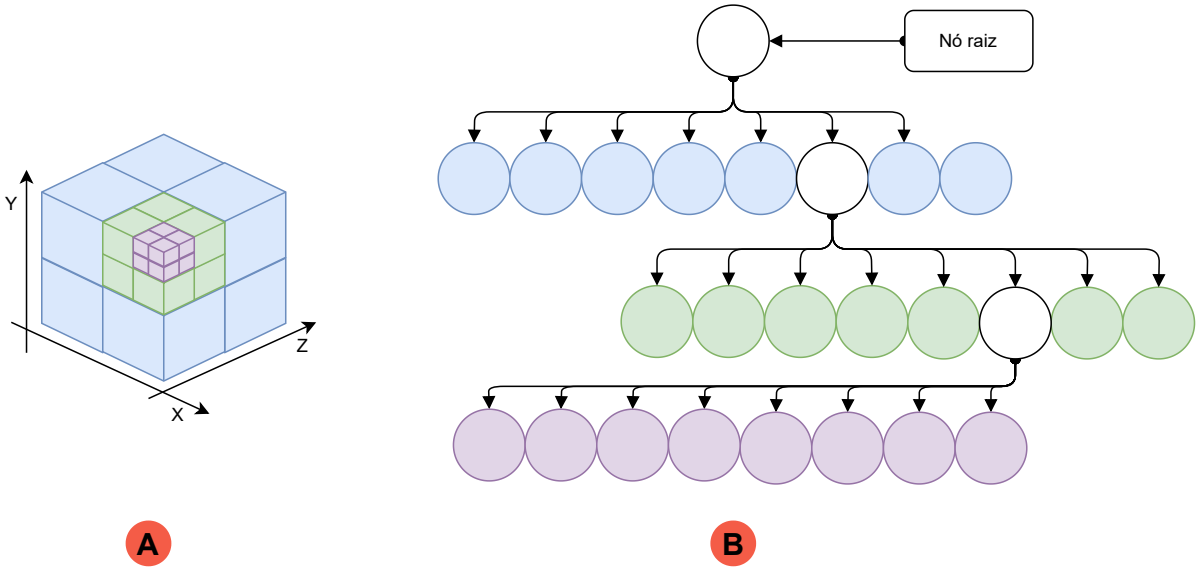

Figura 12 – Representação gráfica de uma estrutura de dados Octree.

2.5 Trabalhos Relacionados

Em sistemas computacionais destinados ao processamento de séries temporais de alto volume, as restrições impostas pelas limitações de capacidade da memória primária exigem uma otimização das metodologias de persistência e recuperação de dados. Portanto, conjuntos de dados massivos superam frequentemente o espaço disponível na memória primária, resultando em estratégias de acessos frequentes à memória secundária (que é um gargalo computacional). Neste contexto, a recuperação completa de blocos de armazenamento quando apenas subconjuntos específicos são requeridos para o uso em processamento adiciona uma sobrecarga desnecessária, degradando o tempo da operação. A redução do tamanho dos blocos via particionamento pode ser uma solução parcial, entretanto existe um equilíbrio no tamanho do bloco para maximizar o tempo de recuperação dos de uma

região de interesse. Assim, a redução excessiva do tamanho de bloco não é uma solução que minimize o tempo de recuperação de dados da memória secundária.

A reorganização estratégica dos blocos tem objetivo de minimizar o desperdício e otimizar o tempo de consulta. Na literatura, são descritos métodos para particionamento de dados matriciais, como os dados *raster*, destacando duas vertentes: (i) o particionamento espacial [11, 12, 13, 20, 8, 9] e (ii) espaço-temporal [14, 47]. No entanto, é importante salientar a diferença de que a otimização do particionamento é mais ampla do que a indexação espaço-temporal, que, embora análoga, possui foco distinto. Conforme discutido nos trabalhos de Mahmood et al. [48] e Tian et al. [22], a pesquisa em indexação espaço-temporal concentra-se na criação e otimização de estruturas de acesso para suportar o processamento eficiente de consultas. Esses trabalhos propõem diversas abordagens de indexação para tipos de dados específicos, como dados históricos [49], em tempo real [50], textuais [51] e vetoriais [52]. Em contrapartida, este trabalho não visa apenas propor uma nova estrutura de índice. O foco é utilizar uma organização hierárquica de índices como um método para permitir o acesso eficiente dos dados, mas com objetivo central no particionamento espaço-temporal físico dos dados. Portanto, enquanto a literatura citada se concentra na estrutura do índice como o produto final da otimização, aqui o índice é um meio para alcançar o objetivo principal de otimizar a organização dos dados em blocos persistidos em memória secundária.

No particionamento espacial, cada matriz de uma série temporal (instante temporal) é particionada de forma independente. Ou seja, os dados particionados não têm relação direta com os dados temporais adjacentes na série temporal. Por exemplo, a R*-Grove [11] é apresentada como uma estratégia de particionamento para instantes espaciais baseada na R-tree, restringindo as divisões com dimensões voltadas para a redução de fragmentação. Deste modo, reduz-se o número de recuperações de blocos da memória secundária, promovendo consultas mais eficientes, mas não tanto para operações espaço-temporais. De forma semelhante, a DSTree [13], que é uma estrutura para consultas espaço-temporais que combina uma árvore de intervalos temporais com uma Quadtree para cada instante de tempo, efetua o particionamento apenas para instantes espaciais. A primeira camada gerencia intervalos temporais e a segunda indexa dados espaciais, mas os blocos armazenam dados de um único instante de tempo, não explorando a relação temporal dos dados no particionamento.

Já o particionamento espaço-temporal considera sequências de dados matriciais ou de tensores, ou seja, é previsto que em um único bloco existam dados de uma região em um período temporal pré-fixado. Por exemplo, o trabalho de Hu et al. [14] divide os dados em subconjuntos de tensores N-dimensionais e os armazena em memória secundária seguindo um particionamento uniforme. O trabalho aproveita as propriedades da arquitetura Hadoop, que mantém grandes blocos (e.g., 64 MB/128 MB), assim conseguindo armazenar regiões

em longos períodos em um mesmo bloco. De forma análoga, o trabalho de Li et al. [47] utiliza a arquitetura Hadoop para particionar e armazenar dados tridimensionais nos blocos. A abordagem se baseia na distribuição estratégica e na indexação para otimizar a execução de consultas. No entanto, a utilização de blocos muito grandes limita o desempenho em operações que requerem acesso a uma porção minoritária do bloco, em termos espaço-temporais. Por exemplo, consultas que exigem apenas um ou poucos instantes de tempo, ou consultas com uma região de interesse relativamente pequena com relação à região de cobertura do bloco, ainda que envolvendo um intervalo maior de instantes de tempo, são forçadas a recuperar todo o bloco, que contém um volume muito maior de dados do que o necessário, degradando o processamento.

Além desses métodos, existe a abordagem *in situ*, que pode trabalhar com as duas vertentes de particionamento, adotando uma delas dependendo do tipo de arquivo e codificação, mas de forma estática. Isto é, para modificar o método de particionamento, é preciso codificar o dado completamente. Por exemplo, a biblioteca `GDAL` [15] pode trabalhar com arquivos no formato `TIF`, `JP2`, `NetCDF` e `HDF5`. Nos dois primeiros tipos, cada instante temporal é processado isoladamente. Já nos formatos `NetCDF` e `HDF5`, os arquivos podem ser codificados para trabalharem usando a metodologia espaço-temporal. De forma similar, o `ChronosDB` [9] também opera *in-situ*, abstraindo formatos via tensores N-dimensionais e delegando processamento à biblioteca `GDAL`. Embora o `ChronosDB` possa trabalhar com o particionamento espaço-temporal oferecido pela `GDAL`, seu foco é o particionamento de instantes temporais. Esta abordagem particiona os dados adicionando uma margem com informações provenientes dos particionamentos vizinhos, criando uma margem de sobreposição. Assim, ele visa reduzir o número de acessos à memória secundária quando somente poucos dados de interesse se encontram na margem de sobreposição.

A análise detalhada dessas abordagens revela um panorama onde a eficiência é frequentemente sacrificada em prol da especialização em uma única dimensão (espacial ou temporal). Enquanto métodos espaciais falham em capturar a continuidade temporal, métodos baseados em grandes volumes (Hadoop) impõem uma penalidade severa de I/O para consultas de granularidade fina. A Tabela 1 sintetiza as principais características, métodos e limitações dos trabalhos discutidos, evidenciando a necessidade de uma abordagem mais flexível.

Deste modo, percebe-se que há uma lacuna de propostas de particionamento de dados envolvendo séries de dados matriciais que sejam ajustadas para operações espaço-temporais. Tais estratégias permitiriam recuperar somente as partições que intersectam a área de interesse para consultas e conjuntos de dados variados, melhorando significativamente o desempenho em aplicações de aprendizagem de máquina [16], identificação de padrões e monitoramento de regiões [53]. As abordagens existentes permitem melhorar consultas específicas, mas ao custo de reduzir a eficiência de outras. Os particionamentos es-

tritamente espaciais não prevem otimizações para reduzir o custo de operações envolvendo sequências de dados de uma mesma região. Por outro lado, as propostas de particionamento espaço-temporal atuais são voltadas a um cenário específico de operações sobre volumes massivos de dados, tornando-se ineficientes para cargas de trabalho mais gerais. Neste contexto, a proposta deste trabalho estende o estado da arte por ser uma estratégia de particionamento dinâmico de dados espaço-temporal, que pode ser ajustada à carga de trabalho sem a necessidade de reprocessamento completo do particionamento. Desta forma, permite um melhor balanço do tamanho do bloco e o consequente agrupamento de dados de instantes subsequentes para o aumento da eficiência de operações espaço-temporais sobre regiões de interesse em séries de dados matriciais.

Tabela 1 – Síntese dos trabalhos relacionados: objetivos, métodos e limitações.

Trabalho	**Objetivo Principal**	**Método / Abordagem**	**Limitações Identificadas**
R*-Grove [11]	Reduzir a fragmentação e acessos à memória secundária.	Particionamento espacial baseado em R-tree com restrições de divisão.	Foco estritamente espacial; ineficiente para operações que cruzam a dimensão temporal.
DSTree [13]	Suportar consultas espaço-temporais em séries matriciais.	Estrutura híbrida: Árvore de intervalos (tempo) + Quadtree (espaço).	Blocos armazenam instantes isolados; não explora a relação temporal no particionamento físico.
Hu et al. [14]	Processamento de tensores N-dimensionais em larga escala.	Particionamento uniforme em arquitetura Hadoop (blocos de 64/128 MB).	Blocos excessivamente grandes geram sobrecarga de leitura para consultas de pequenas regiões.
Li et al. [47]	Otimizar a execução de consultas em dados 3D.	Distribuição estratégica e indexação sobre o ecossistema Hadoop.	Baixo desempenho em consultas que requerem apenas subconjuntos específicos do bloco (ROI reduzida).
GDAL ChronosDB [15, 9]	Abstração de formatos e processamento *in situ*.	Particionamento estático (TIF/JP2) ou via tensores (NetCDF/HDF5) com margens de sobreposição.	Natureza estática: qualquer mudança no método de particionamento exige a recodificação completa dos dados.
Esta Proposta Dympas	**Otimizar a organização física para operações espaço-temporais dinâmicas.**	**Indexação hierárquica e particionamento espaço-temporal dinâmico/ajustável.**	—

3 DYNAMIC MATRIX PARTITIONING SYSTEM

Este capítulo descreve a abordagem *Dympas* (***Dy****namic* ***M****atrix* ***Pa****rtitioning* ***S****ystem*), que é uma nova proposta de particionamento de dados matriciais visando minimizar o tempo de recuperação de dados de regiões de interesse em um determinado intervalo de tempo. A proposta organiza o particionamento no espaço e no tempo, a fim de equilibrar o tempo de recuperação dos dados e a redução do desperdício de dados recuperados da memória secundária conforme a demanda de consultas espaço-temporais. Isso é alcançado por meio da reorganização dos dados de forma dinâmica com as operações de *split* e *unsplit*, o que a torna uma abordagem flexível, superando limitações dos métodos de particionamento estáticos identificados na literatura.

3.1 Demonstração Prática da Metodologia

Esta seção apresenta um cenário de análise de dados geoespaciais para ilustrar o funcionamento do Dympas. São descritos o cenário inicial, o particionamento espacial de referência, a aplicação das operações de *split* espaço-temporal e os efeitos dessas operações no desempenho da recuperação dos dados.

3.1.1 Configuração do Cenário Geoespacial

Para fins experimentais, o conjunto de dados é parametrizado da seguinte forma:

- **Dimensão Temporal:** Composta por uma série de n instantes temporais, dos quais selecionamos quatro instantes subsequentes ($T_{n1}, T_{n2}, T_{n3}, T_{n4}$) para representar o intervalo de interesse.
- **Dimensão Espacial:** Cada instante temporal é representado por uma matriz bidimensional que abrange uma região geográfica contendo o talhão alvo.
- **Limiar de Particionamento:** O sistema utiliza blocos de tamanho fixo na memória secundária para otimizar a transferência de dados. O particionamento (nível 0) visa alinhar o tamanho físico do bloco para um equilíbrio de tempo de leitura e quantidade de dados com relevância para uma consulta.

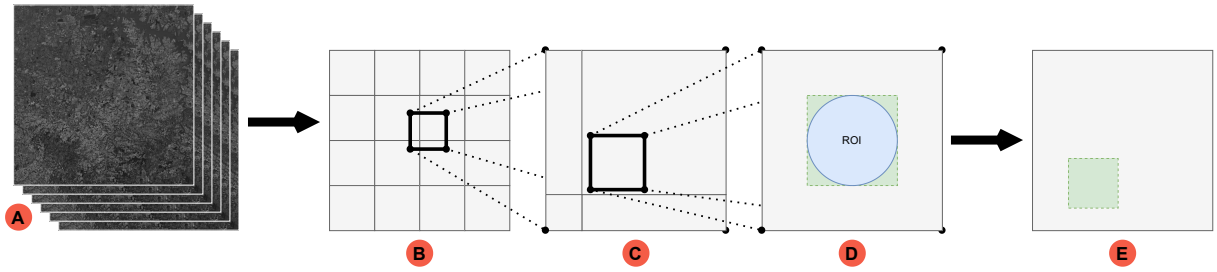

Figura 13 – Representação da configuração de um cenário geoespacial

A Figura 13 ilustra a configuração do cenário. Em **A**, observa-se a série espaço-temporal original. O item **B** detalha um único instante temporal sob um **particionamento de nível 0** (estritamente espacial). A ampliação em **C** e **D** revela a área de busca (o MBR) em verde, que circunda o talhão de interesse (em azul). Por fim, o item **E** destaca o bloco atômico de dados que contém a região de interesse, evidenciando a unidade mínima de acesso à memória secundária.

3.1.2 Estado Inicial: Particionamento nível 0 (Exclusivamente Espacial)

No nível de particionamento inicial (nível 0), a divisão dos dados é puramente espacial e iterativa até que o limiar de tamanho de bloco seja atingido. Como demonstrado na Figura 14, cada instante temporal é tratado como uma unidade isolada para fins de persistência. Cada bloco resultante atua de forma atômica: a recuperação de qualquer dado contido nele exige a leitura integral do bloco na memória secundária.

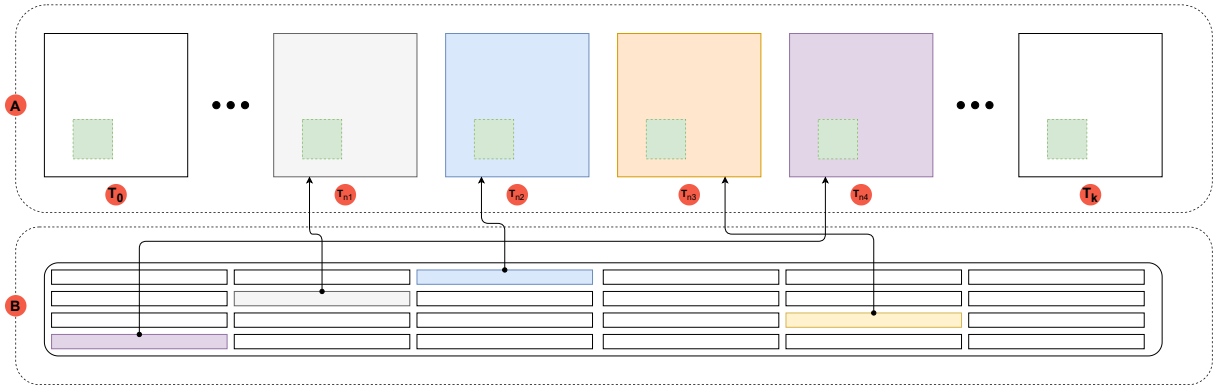

Figura 14 – Representação dos dados decodificados e a representação dos dados persistidos para particionamento de nível 0.

A Figura 14 ilustra a ineficiência deste modelo para consultas temporais. Em **A**, é identificado o bloco que contém o MBR (verde) em todos os instantes (T_{n1} a T_{n4}). Em **B**, é representado o mapeamento em memória secundária, onde os dados são armazenados

em segmentos distintos. Para satisfazer uma consulta que abranja o período $T_{n1} - T_{n4}$, para realizar a consulta são requeridas quatro operações de leitura distintas (uma para cada bloco temporal).

Existe uma baixa seletividade semântica, em que uma fração significativa dos dados recuperados (áreas em cinza, azul, laranja e roxo na Figura 14 A) é descartada após a leitura, pois não pertence à MBR de interesse. Dado que o acesso à memória secundária é ordens de grandeza mais lento que na memória primária, esse excesso de dados irrelevantes (*overhead* de I/O) degrada o desempenho da consulta.

3.1.3 Otimização via Operação de *Split* Espaço-Temporal

A metodologia Dympas introduz a operação de *split* espaço-temporal para aumentar a quantidade de dados relevantes por bloco. O particionamento de **nível 1** consolida dois instantes temporais subsequentes ($2^1 = 2$ instantes) em uma nova estrutura de bloco. Esta lógica é aplicada ao cenário ($T_{n1} - T_{n4}$), em que duas operações de *split* são aplicadas, a primeira fazendo a junção de $T_{n1} - T_{n2}$ e a segunda $T_{n3} - T_{n4}$.

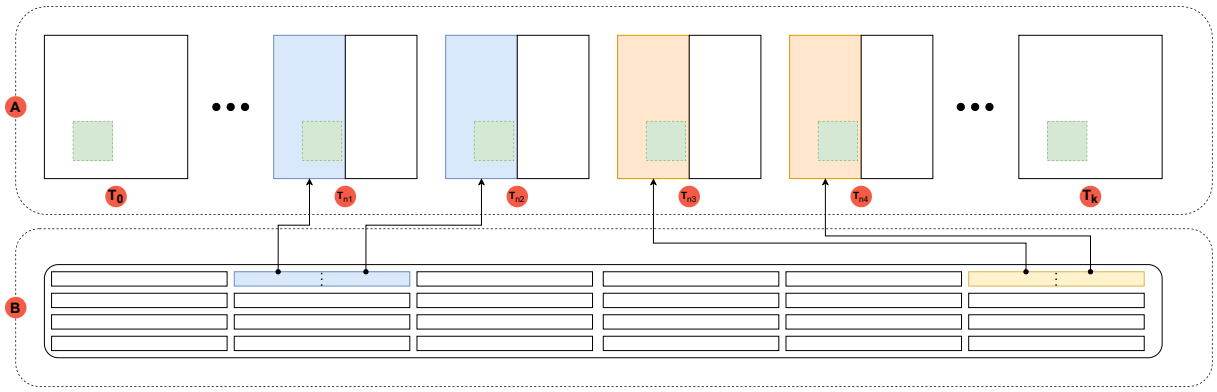

Figura 15 – Representação do resultado da aplicação de duas operações de *split* de nível 1.

Como ilustrado na Figura 15, o resultado é uma reestruturação da persistência física. Agora, são necessários apenas dois segmentos de memória secundária para recuperar o mesmo intervalo temporal, representando uma redução de 50% no número de acessos e metadados de leitura em comparação ao nível 0.

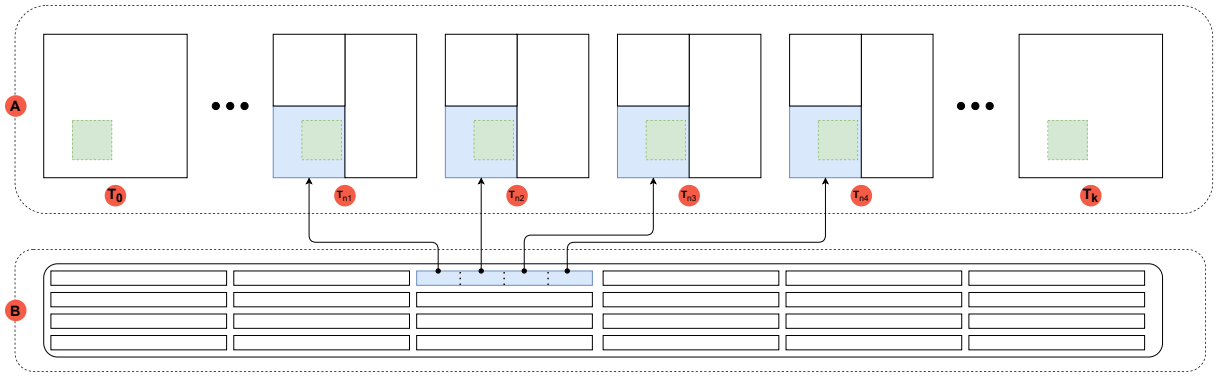

Figura 16 – Representação do resultado da aplicação da operação de *split* de nível 2.

Entretanto, este cenário ainda permite uma otimização adicional, conforme ilustrado na Figura 16. Nela, observa-se a aplicação da operação de *split* de nível 2, que requer a presença de quatro instantes temporais subsequentes ($2^2 = 4s$). Com esta operação, os dois blocos resultantes do nível 1 (na região e período de interesse) sofrem um novo particionamento, desta vez no sentido horizontal, complementando o particionamento vertical anterior.

Como resultado detalhado em **A**, obtém-se um único bloco espaço-temporal que consolida os dados dos quatro instantes temporais. Em **B**, que representa a persistência em memória secundária, é possível notar que os dados que antes estavam fragmentados agora formam um bloco único, persistido em um único segmento de memória. Essa configuração resulta em uma redução de 75% na quantidade de dados recuperados da memória secundária em comparação ao particionamento de nível 0 (exclusivamente espacial) para responder à mesma consulta, sem qualquer perda de informação semântica.

3.1.4 Considerações sobre a Eficiência Metodológica

A partir deste demonstrativo, é possível perceber o potencial de uso da metodologia proposta. A aplicação destas otimizações permite que pesquisadores realizem consultas mais eficientes para a análise de dados matriciais espaço-temporais.

O caso base apresentado (monitoramento agrícola) exemplifica um campo onde esse tipo de consulta é recorrente e pode extrair grandes benefícios da abordagem Dympas. A redução significativa na recuperação de dados desnecessários reflete diretamente na diminuição do tempo de resposta das consultas. Esse ganho é valioso, visto que a natureza deste tipo de análise envolve volumes massivos de dados, onde a eficiência no acesso à memória secundária é determinante para a viabilidade da pesquisa.

3.2 Operações de *Split* e *Unsplit*

Os dados matriciais são inseridos no Dympas para processamento em memória primária, antes da sua persistência em memória secundária. Guiado por um limiar configurável que define o limite da região correspondente a um bloco, o sistema atua sobre cada matriz da série temporal de forma individual, realizando divisões exclusivamente espaciais sucessivas. Adota-se a estratégia de particionamento baseada em KD-tree, que alterna recursivamente cortes horizontais e verticais. A escolha dessa estrutura deve-se ao seu crescimento controlado e à capacidade de desfazer partições para satisfazer restrições estruturais. Por fim, aplica-se a política de compressão aos dados de cada região, armazenando-os em seus respectivos blocos em memória secundária.

O limiar não pode ser excessivamente pequeno para se tirar proveito da largura de banda na transferência de blocos da memória secundária. Com isso, operações espaço-temporais frequentemente consideram ROIs menores do que a região de um bloco em dados matriciais de um conjunto de instantes de tempo subsequentes, demandando acesso, ao menos, a um bloco por instante de tempo. Para otimizar as operações nesses casos, a proposta subdivide espacialmente a região de um número de instantes de tempo subsequentes ($2^{\text{nível}}$) e reorganiza os dados para acomodar dados de cada sub-região dos instantes em um único bloco. Desta forma, o *split* espaço-temporal consiste em operação de divisão de área e aumento temporal de regiões persistidas em um bloco. Inversamente, o *unsplit* espaço-temporal consiste em aumento de região e redução temporal de regiões persistidas em um bloco. Pode-se traçar um paralelo entre estas operações e um sistema tridimensional, onde as dimensões espaciais e temporal se ajustam de forma inversa, mantendo constante o volume total de dados.

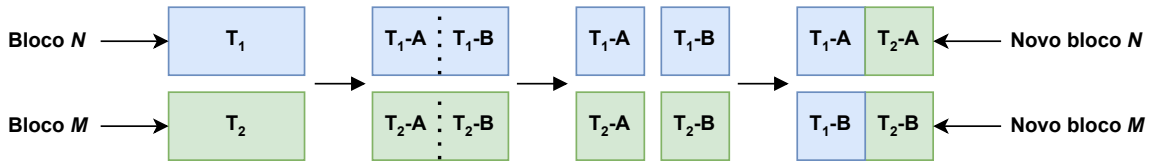

Figura 17 – Funcionamento do *split* espaço-temporal em blocos de dados.

A Figura 17 ilustra o processo de *split*, onde, inicialmente, os dados de uma mesma região em instantes subsequentes T_1 e T_2 são armazenados cada um em um bloco (N e M, respectivamente). Ao detectar-se uma oportunidade de otimização espaço-temporal, o *split* divide a região em duas partes, A e B, reorganiza os blocos preservando o tamanho para armazenar os dados de A nos instantes T_1 e T_2 em um mesmo bloco e faz o mesmo para B. Desta maneira, uma consulta com um ROI circunscrito em uma das sub-regiões, por exemplo, em A, que utilize os dois instantes de tempo, passa a acessar apenas um bloco (novo bloco N).

Este processo é repetido recursivamente conforme a demanda da carga de trabalho, permitindo agrupar em blocos dados de vários instantes de tempo de regiões de área reduzida, mas que ainda cubram as ROIs das operações espaço-temporais executadas. A organização parte do nível de particionamento inicial, o nível 0, contém blocos com área na norma do limiar e com somente um instante temporal. Quando um *split* é aplicado, é necessário aplicar a operação a $2^{\text{nível}}$ instantes temporais consecutivos cronologicamente. Toda a série deve estar em um nível k de particionamento, para ser possível gerar um novo nível de particionamento. Já a operação de *unsplit* reverte um *split*. Dado um estado onde existe um *split* nível X, onde $X \geq 1$, em uma região A. Ao realizar o *unsplit* sobre A, ele retorna para o estado anterior com nível $X - 1$.

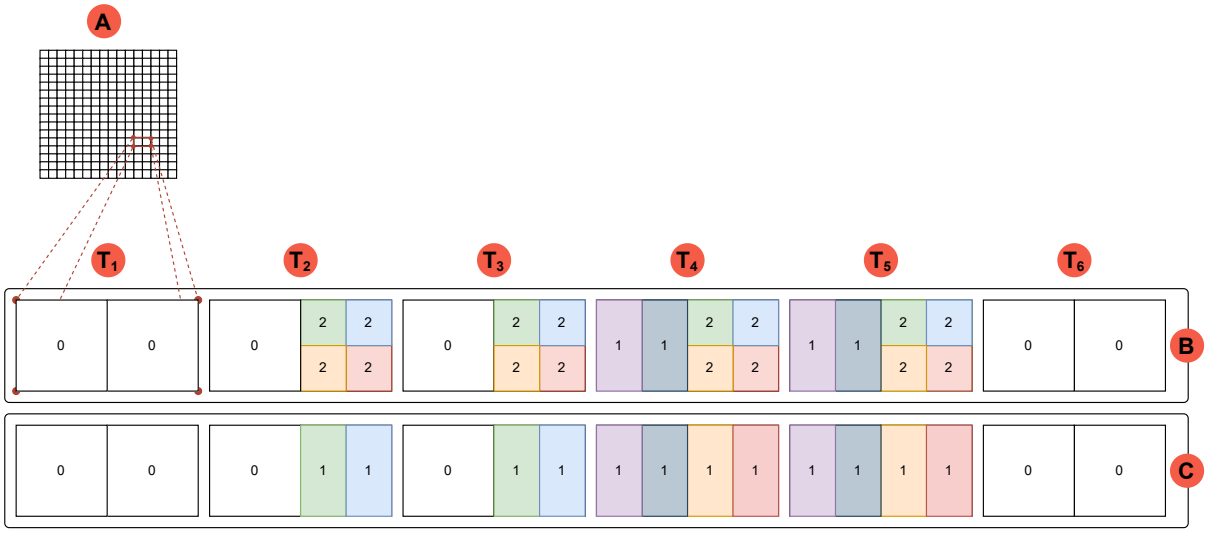

Figura 18 – Funcionamento do particionamento proposto em diferentes níveis.

A Figura 18 apresenta um exemplo de particionamento espaço-temporal com aplicação dos operadores *split* e *unsplit*. A Figura 18 A ilustra o dado matricial em um instante de tempo qualquer. Ao lado (Figura 18 B), são apresentados os blocos correspondentes a duas regiões adjacentes da matriz, mas considerando um intervalo de instantes de tempo $[T_1, T_6]$. Os blocos indicam que foi aplicado um *split* de nível 1 em $[T_4, T_5]$ (regiões demarcadas com 1 na figura) e outro *split* de nível 2, no intervalo $[T_2, T_5]$ (regiões demarcadas com 2). Os blocos brancos, representam o particionamento de nível 0, enquanto os particionamentos com nível ≥ 1 e de mesma cor representam dados que estão em um mesmo bloco. Para alcançar o particionamento de nível 2, foram realizados três *splits* em sequência, sendo dois de nível 1, em $[T_2, T_3]$ e $[T_4, T_5]$, seguidos de um de nível 2, no intervalo $[T_2, T_5]$.

Caso a carga de trabalho indique a necessidade de um *unsplit*, i.e., as consultas estão utilizando dados de um número de instantes de tempo significativamente menor do que o número de instantes agrupados em um bloco, causando acessos a dados de muitos

períodos não utilizados, o *unsplit* pode ser invocado para retroceder o particionamento ao nível acima. Por exemplo, um *unsplit* sobre o particionamento de nível 2 no intervalo $[T2, T5]$ no exemplo, resulta em particionamentos de nível 1, em $[T_2, T_3]$ e $[T_4, T_5]$, conforme ilustrado na Figura 18 C.

Embora os operadores *split* e *unsplit* contribuam para otimizar as consultas ao reduzir a leitura de dados desnecessários, seu potencial só é alcançado por meio de um gerenciamento de blocos de dados eficientes. A razão para essa dependência reside na estrutura dos blocos que, por conterem múltiplos instantes de tempo, são candidatos ideais para uma gestão de cache que evite acessos redundantes à memória secundária.

3.3 Estruturas de Armazenamento e Indexação

Uma vez estabelecida a estratégia de organização de dados, que visa otimizar as consultas através do rearranjo físico da informação, o próximo passo é descrever o sistema de indexação e sua complexidade computacional. Este sistema é o componente responsável por localizar eficientemente um bloco de dados a partir de uma região espacial e um período. Para este fim, a abordagem Dympas adota uma estrutura hierárquica de três níveis, conforme ilustrado na Figura 19:

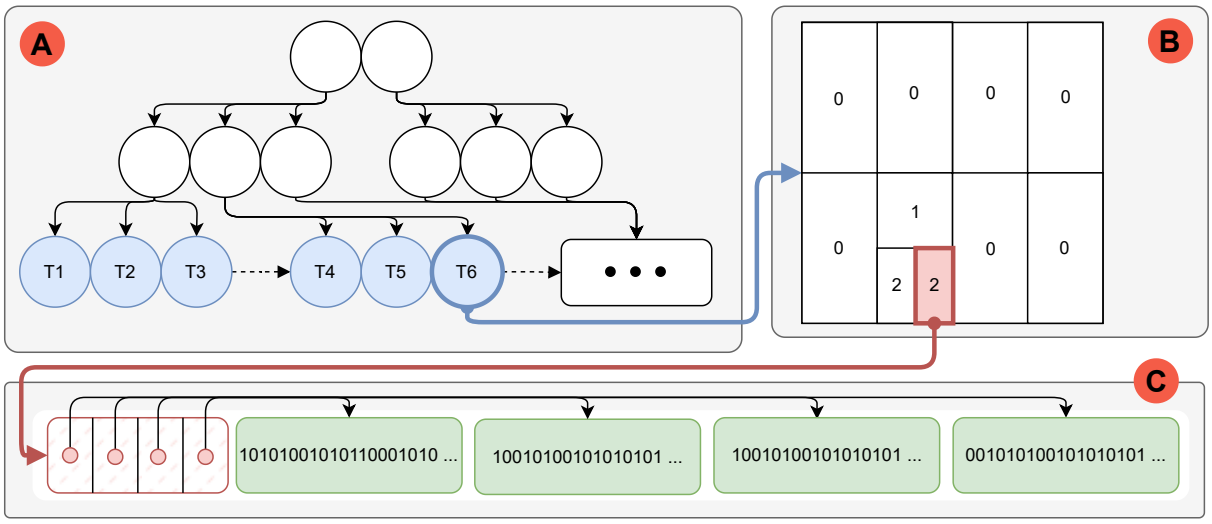

Figura 19 – Estrutura da abordagem proposta.

1. B^+tree (Figura 19A) para indexação temporal de cada instante temporal das matrizes inseridas na estrutura, onde cada nó folha tem um *timestamp* T_i;

2. KD-tree (Figura 19B), para cada nó folha da B^+tree existe uma KD-tree correspondente ao particionamento atual referente à matriz no instante temporal T_i, onde cada nó tem um UID (*unique identifier*) referente a um bloco persistido na memória secundária com os dados referentes àquela região;

3. Hash Index (Figura 19C), como um índice *in loco* no bloco, onde, para cada UID, existe um bloco que tem um ou mais instantes temporais.

A escolha da B^+tree se deve à sua eficiência para buscar dados ordenáveis, como instantes temporais, bastando percorrer a árvore somente uma vez até um nó folha em tempo $\mathcal{O}(\log N_1)$ [54], onde N_1 é a altura da árvore, e percorrer os nós folhas até a busca cobrir todo o intervalo. Assim, para localizar um período temporal de tamanho K_1 tem-se um tempo de busca de $\mathcal{O}(\log N_1 + K_1)$ [39].

O uso da estrutura KD-tree justifica-se por sua capacidade de particionar o espaço bidimensional com um padrão rigorosamente definido, onde a orientação de cada divisão se inverte na subsequente. A versão aplicada impõe ainda uma restrição adicional, onde cada divisão é realizada precisamente na metade do tamanho da dimensão correspondente (Corolário 1). Essa abordagem resulta em uma complexidade de busca dada por $\mathcal{O}(N_2^{1-\frac{1}{d}} + K_2)$ [54], na qual d representa a dimensionalidade (neste caso, 2), N_2 indica a altura da árvore (ou número de particionamentos) e K_2 corresponde à quantidade de regiões que intersectam a área de busca.

A terceira camada de indexação utiliza *hash in loco* (Figura 19C). Cada bloco reúne metadados para o mapeamento *hash* e dados binários de uma região da KD-tree, abrangendo um ou mais instantes temporais. Ao efetuar o *hash* de um *timestamp*, identifica-se o trecho de dados correspondente ao ROI e ao *timestamp*. Se esse bloco contiver vários instantes, ele tende a ser reutilizado e, por isso, deve ser mantido em memória primária. Se o bloco estiver na memória secundária, a busca tem complexidade $\mathcal{O}(B)$ (com B representando o tamanho do bloco); se estiver em memória primária, a busca ocorre em $\mathcal{O}(1)$ [39].

Com isso, a complexidade total da estrutura para busca é:

$$\mathcal{O}\left((\log N_1 + K_1) + K_1 \cdot [(N_2^{1-\frac{1}{d}} + K_2) + (K_2 \cdot B)]\right).$$

Para cada um dos K_2 nós folha da KD-tree que intersectam com a ROI, é feito um acesso a bloco via *hash index in loco*, e o intervalo de tempo da consulta espaço-temporal requer o acesso a cada uma das K_1 KD-trees retornadas pela busca temporal na B^+tree. Ressalte-se que a implementação do Dympas reduz o custo médio significativamente com o uso de um mecanismo de *cache* de blocos. A redução do custo é particularmente relevante para blocos que armazenam dados de diversos instantes de tempo, demandando um único acesso à memória secundária para todos os instantes agrupados no bloco, tornando o termo $\mathcal{O}(K_2 \cdot B)$ desprezível nesses casos. Isso é devido ao bloco com o instante de interesse recuperado da memória secundária, assim os demais instantes requisitados daquele bloco têm o retorno diretamente da *cache*.

3.3.1 Equivalência Espacial

A Dympas propõe uma estrutura de indexação espacial baseada na KD-tree, impondo uma nova restrição em relação à sua formulação convencional. Esta restrição tem o intuito de gerar um particionamento previsível independente do período temporal. Ela determina que, ao se alternar os eixos de divisão, o corte deve ser efetuado exatamente no ponto médio do eixo escolhido. Com isso, é alcançada a propriedade de *equivalência espacial*, que assegura que, a partir de duas matrizes com as mesmas dimensões iniciais, dada uma mesma coordenada espacial em ambas, os nós que a intersectam em um mesmo nível de profundidade terão a mesma forma e perfeito alinhamento espacial (coordenadas espaciais exatas de origem e fim). Isso permite que as operações espaço-temporais sejam realizadas sem conflitos de sobreposição de blocos.

Teorema 1. *Sejam duas KD-trees (A e B), ambas construídas sobre matrizes de dimensão $M \times N$, onde $(M, N) \in \mathbb{N} \wedge (M, N) \geq 2$, e particionadas usando a propriedade de "dividir sempre no meio, alternando eixos vertical e horizontal" (com divisão inteira: $\lfloor \cdot \rfloor, \lceil \cdot \rceil$). Então, para toda profundidade d, quaisquer dois nós nessa profundidade ocupam regiões com mesma forma, mesmas dimensões e idêntico alinhamento espacial.*

Demonstração. Indução sobre d.

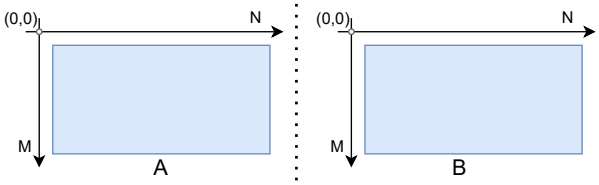

Figura 20 – Equivalência espacial - base

Base ($d = 0$). A profundidade 0 é a raiz: cada árvore cobre a matriz inteira $[0, M] \times [0, N]$. Idêntico em todas (Figura 20).

Hipótese (para $d = k$). Suponha que, em ambas as árvores, todo nó em profundidade k corresponde à mesma região $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, com $x_{\max} - x_{\min} \in \{\lfloor M/2^k \rfloor, \lceil M/2^k \rceil\}$ e idem para y.

Passo ($d = k + 1$). Cada nó em profundidade k é subdividido no ponto médio inteiro da dimensão escolhida:

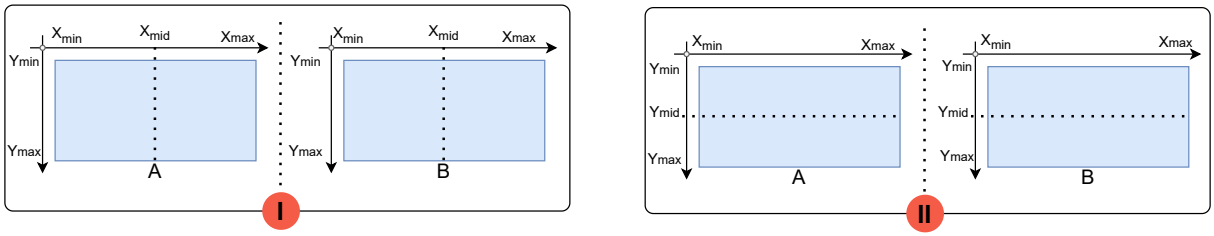

Figura 21 – Equivalência espacial - passo

- Se divide verticalmente em

$$x_{\text{mid}} = \left\lfloor \frac{x_{\min} + x_{\max}}{2} \right\rfloor .$$

Então as duas sub-regiões são $[x_{\min}, x_{\text{mid}}] \times [y_{\min}, y_{\max}]$, $[x_{\text{mid}+1}, x_{\max}] \times [y_{\min}, y_{\max}]$ (Figura 21 I).

- Se divide horizontalmente em

$$y_{\text{mid}} = \left\lfloor \frac{y_{\min} + y_{\max}}{2} \right\rfloor ,$$

gerando $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\text{mid}}]$, $[x_{\min}, x_{\max}] \times [y_{\text{mid}+1}, y_{\max}]$ (Figura 21 II).

Por hipótese, todos os nós de nível k têm o mesmo intervalo $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. A operação "dividir no meio" produz, a partir dessa região fixa, sempre dois pares de sub-intervalos cujas larguras são $\lfloor (x_{\max} - x_{\min})/2 \rfloor$ e $\lceil (x_{\max} - x_{\min})/2 \rceil$ (ou equivalente em y). Logo, em profundidade $k+1$, todos os nós têm regiões idênticas em forma, dimensão (diferença ≤ 1 se o comprimento for ímpar) e nas exatas coordenadas espaciais de origem e fim.

Portanto, por indução matemática, a geometria de qualquer nó em qualquer profundidade d é deterministicamente definida, o que prova o teorema. □

Corolário 1. *Como consequência direta do Teorema 1, se duas KD-trees obedecem às restrições de particionamento estabelecidas, então qualquer ROI fixa que intersecte a malha em uma determinada profundidade d abrangerá os mesmos nós (preservando idênticas formas, dimensões e alinhamento espacial) em ambas as árvores.*

3.4 Normas do Particionamento

No âmbito da otimização de consultas, os critérios para efetuar ou reverter o particionamento definem a organização dos dados, impactando diretamente o custo de acesso à memória secundária durante as buscas. Portanto, o Dympas estabelece 4 normas que controlam a aplicação de operação de otimizações. O rigoroso uso destas normas permite não somente a implementação de estratégias de particionamento que resultam

em otimizações eficientes, mas também a prevenção de particionamento desnecessários, os quais poderiam introduzir caraga de trabalho computacional sem ganhos proporcionais. Além de que a correta aplicação destas normas é um pré-requisito para utilização da estratégia proposta, assim, as tomadas de decisões operem sob condições que maximizem a eficiência do processamento de consultas espaciais e espaço-temporais.

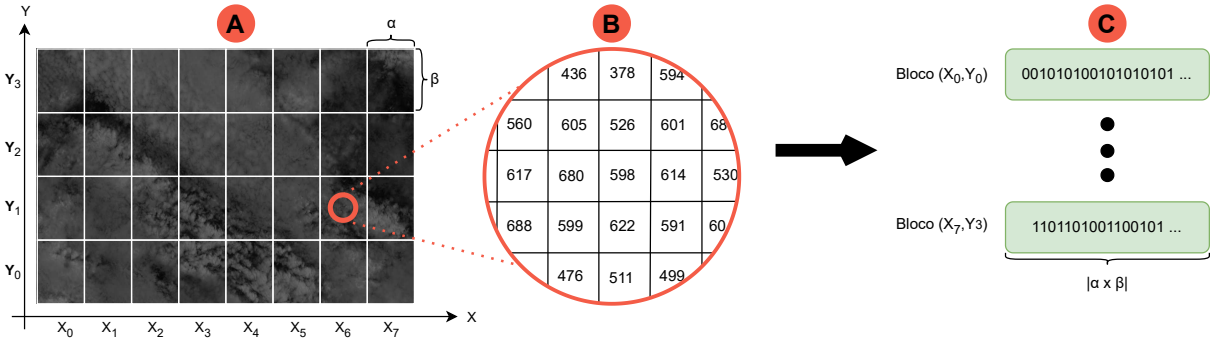

Figura 22 – Primeira norma do particionamento espaço-temporal.

Inicialmente, os parâmetros de entrada consistem em uma matriz de alta dimensionalidade, como um dado *raster* geoespacial (Figura 22 A), e um limiar. Esse limiar representa o valor mínimo do produto entre as dimensões de uma região mínima desejada. Isso permite a divisão da matriz em blocos de dimensões homogêneas, evitando que sejam gerados blocos com proporções distorcidas nas bordas. Situação típica quando os blocos fixos tem as dimensões pre-definidas imutáveis e as dimensões do dado de entrada não são múltiplas das dimensões pre-definidas. Assim, a **primeira norma** define que assim que uma matriz é inserida no sistema (Figura 22 B) é particionada seguindo as propriedades da KD-tree do Dympas. São realizados **particionamento espaciais subsequentes** até que o particionamento atual tenha α (largura) e β (altura) tais que $\delta = \alpha \times \beta$ satisfaça as condições: $(2 \times limiar) > \delta \quad \text{e} \quad \delta \geq limiar$. Dessa maneira, obtém-se uma estrutura de particionamento de nível 0 (Figura 22 C). Os dados deste particionamento são persistidos em memória secundária de forma exclusiva em um único bloco. Esse processo de particionamento é reiterado a cada novo dado matricial inserido na estrutura.

No particionamento de nível 0, apesar de haver uma redução do desperdício de recursos, as consultas são otimizadas para serem realizadas sobre um instante temporal, o que pode acarretar maior uso de recursos computacionais sem valor semântico em consulta espaço-temporais, o que é problemático em análises espaço-temporais, visto que esses dados são carregados da memória secundária. Portanto, maximizar a redução de carregamento destes dados é ponto central da estratégia proposta através da reorganização destes blocos em consultas espaço-temporais. Em particionamentos a partir do nível 0, são realizadas operações espaço-temporais, onde a otimização é realizada através da organização temporal dos dados dos blocos de um período. Para isso, deve garantir que os dados tragam uma

melhora no tempo quando é realizada alguma operação de recuperação neste período otimizado. Assim, as outras três normas atuam sobre este contexto e devem ser seguidas para obter o melhor nível de particionamento sobre uma região espaço-temporal.

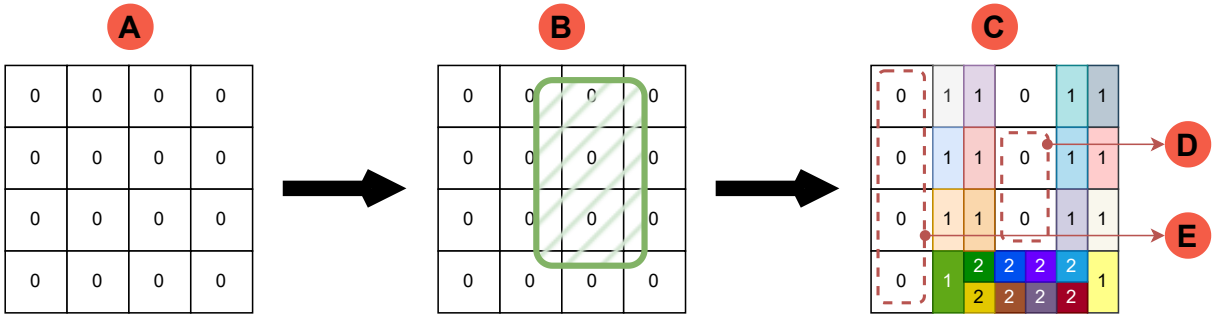

Figura 23 – Segunda norma do particionamento espaço-temporal.

A **segunda norma** estabelece que a otimização via particionamento deve ser aplicada exclusivamente às partições que apresentam **cobertura parcial** pela ROI. A Figura 23 ilustra a aplicação desta norma: Figura 23 A exibe um instante temporal de um contexto de consulta espaço-temporal antes da otimização; A Figura 23 B exibe o mesmo instante temporal, destacando a ROI sobrepondo parte dos particionamentos, todos inicialmente ao nível 0; e Figura 23 C exibe o resultado após a otimização. Conforme a norma, observa-se na Figura 23 C que os particionamentos originais não cobertos pela ROI (Figura 23 D) e aqueles totalmente cobertos (Figura 23 E) permanecem inalterados. Em contraste, os demais particionamentos, com cobertura parcial, foram efetivamente otimizados visando a melhor configuração de particionamentos mais eficiente para atender ao contexto da consulta.

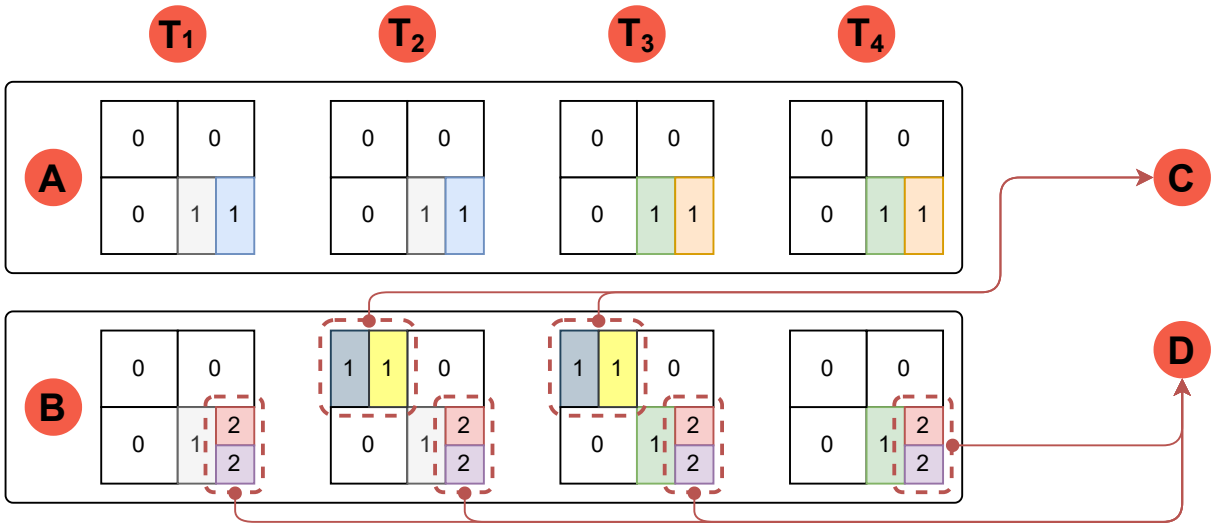

Figura 24 – Terceira norma do particionamento espaço-temporal.

A **terceira norma** estabelece as pré-condições para a execução de uma operação de *split*. Esta norma exige que, primeiramente, a partir do instante inicial da operação, exista uma série temporal com **comprimento suficiente** para permitir o particionamento espaço-temporal planejado. Em segundo lugar, e de forma simultânea, é exigido que todas as partições envolvidas na operação de *split*, pertençam ao **mesmo nível hierárquico**, sendo este nível exatamente uma unidade inferior àquele que se pretende alcançar após a conclusão da referida operação. A Figura 24 ilustra a aplicação desta norma. A Figura 24 A exibe um fragmento espaço-temporal sobre o período $[T_1, T_4]$, onde T_4 é o último instante temporal do sistema. A Figura 24 B exibe o resultado após a aplicação de duas operações de *split*: uma que gera os particionamentos de nível 1 no intervalo $[T_2, T_3]$ (detalhada na Figura 24 C) e outro gera os particionamento de nível 2 no intervalo $[T_1, T_4]$ (detalhada na Figura 24D).

Ao analisar a operação ilustrada na Figura 24 C, com particionamento nível 1 a partir de T_2 observa-se o cumprimento a norma: *(I)* **suficiência temporal**, onde a operação exige $2^1 = 2$ instantes temporais ($[T_2, T_3]$), o que é atendido pelo intervalo disponível; e *(II)* **uniformidade de nível**, onde os particionamentos originais envolvidos (Figura 24 A em $[T_2, T_3]$) possuem o particionamento de nível 0, sendo $1 - 1 = 0$, viabilizando o particionamento para o nível 1. De forma análoga, o *split* ilustrado em Figura 24 D (nível 2 a partir de T_1) demanda $2^2 = 4$ instantes ($[T_1, T_4]$) e particionamentos de nível 1 ($2-1 = 1$). Como tais condições são satisfeitas, a operação é válida. Contudo, a norma impediria uma operação de *split* para o particionamento de nível 2 a partir de T_2, pois a exigência de $2^2 = 4$ instantes (T_1, T_5) não seria cumprida, dado que T_4 é o último instante, violando assim a condição de suficiência temporal.

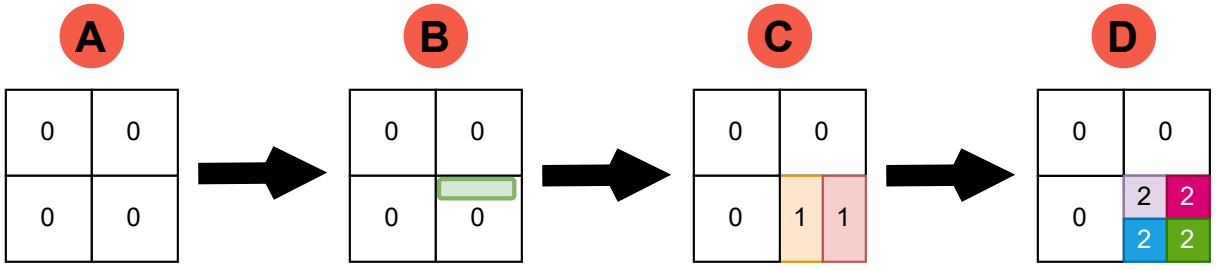

Figura 25 – Quarta norma do particionamento espaço-temporal.

Por fim, a **quarta norma** estabelece a aplicação de operações split subsequentes sobre uma mesma ROI. Ela define que tal sequência de otimizações somente deverá ser efetuada se o resultado implicar em alguma **redução efetiva da região espacial** (no contexto atual ou adjacente) necessária para satisfazer o contexto da consulta que justificou a otimização. Embora uma operação de *split* possa, em um passo intermediário, manter a área de dados acessada, a sequência completa só se justifica se a granularidade final permitir um acesso mais focado, diminuindo o volume de dados sem valor semântico recuperados.

Deste modo, a aplicação desta norma deve considerar o impacto do conjunto de operações *split* necessárias para atingir um nível de particionamento desejado na otimização do contexto de uma consulta, em vez de avaliar cada *split* de forma isolada, reconhecendo que o nível de particionamento afeta diretamente este critério.

A Figura 25 ilustra a aplicação desta norma via um exemplo onde partições de nível 0 são elevadas ao nível 2, necessitando, portanto, de um conjunto de operações *split*. A Figura 25 A ilustra um instante temporal presente no contexto da consulta que se deseja otimizar, enquanto a Figura 25 B ilustra este mesmo instante com a ROI sobreposta, cobrindo parcialmente uma partição de nível 0. Na Figura 25 C, pode-se observar o resultado após o *split* para o nível. Neste ponto, se análise fosse isolada, a otimização poderia ser rejeitada, pois a área recuperada para a consulta permanece a mesma. Contudo, a Figura 25 D demonstra o estado após a aplicação das operações *split* subsequentes até alcançar o nível 2. Neste caso, a região de dados efetivamente recuperada para a consulta é reduzida em comparação com o estado inicial. Portanto, como o resultado atende o critério redução, a sequência completa de *split* é considerada válida, consequentemente, executada.

Enquanto as normas apresentadas definem os alicerces teóricos do Dympas, sua implementação prática demanda uma metodologia específica. Portanto, é necessário introduzir as diretrizes gerais de alinhamento de *splits* e *unsplits* para a realização de um particionamento espaço-temporal que esteja conforme os princípios normativos do sistema.

3.5 Alinhamento de *Splits* e *Unsplits*

O Dympas implementa a técnica de particionamento espaço-temporal para organizar dados multi temporais em blocos de tamanho uniforme. O particionamento espaço-temporal é uma estratégia flexível, permitindo iniciar o particionamento em qualquer instante temporal dentro do domínio dos dados, desde que não haja conflitos de sobreposição entre os particionamentos. Para garantir a eficiência e evitar problemas, seguem-se algumas diretrizes específicas.

A primeira etapa consiste na definição do intervalo de particionamento, baseada nas características da consulta e nos requisitos de otimização (contexto de interesse). Após definir o intervalo, verifica-se se a região espacial que ele abrange é compatível com um ou mais particionamentos espaço-temporais. Caso aplicável, os particionamentos desnecessários são revertidos para viabilizar a criação de novos particionamentos, visando minimizar o número de blocos acessados em memória secundária.

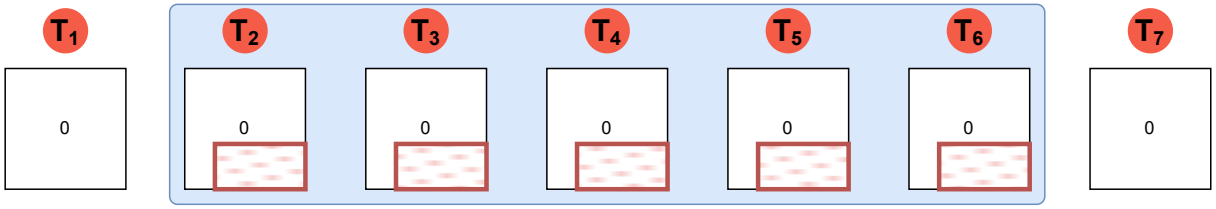

Figura 26 – Ilustração de um fragmento temporal, com o contexto de uma consulta.

A Figura 26 ilustra este conceito, mostrando um fragmento espaço-temporal com uma ROI delimitada ($[T_2, T_6]$). A análise do fragmento permite identificar a melhor forma de particionar os dados para atender à consulta, considerando a ROI (destacada em vermelho) e os blocos dentro do período de interesse.

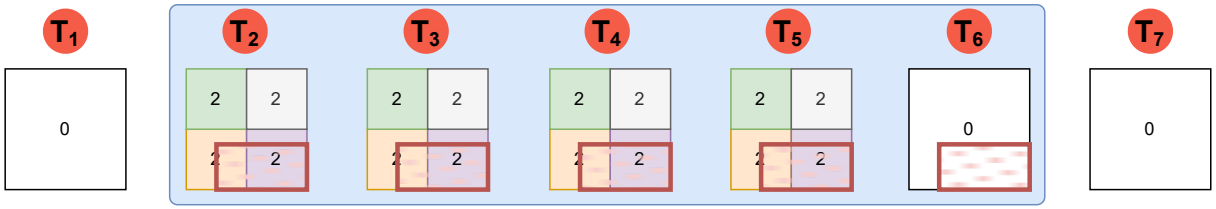

Figura 27 – Ilustração da otimização aplicada ao caso de particionamento espaço-temporal simples.

A Figura 27 ilustra a otimização aplicada ao caso de particionamento espaço-temporal simples. Considerando um contexto de interesse com cinco instantes temporais disponíveis, o maior particionamento que pode ser realizado, conforme as normas descritas na Seção 3.4, é o particionamento de nível 2 ($2^2 = 4$ instantes temporais). O particionamento é efetuado no primeiro instante temporal disponível dentro do contexto de interesse. Como não existem mais particionamentos espaço-temporais que podem ser realizados nesse contexto, a etapa de otimização é finalizada, resultando em uma redução de 5 blocos para 3 blocos.

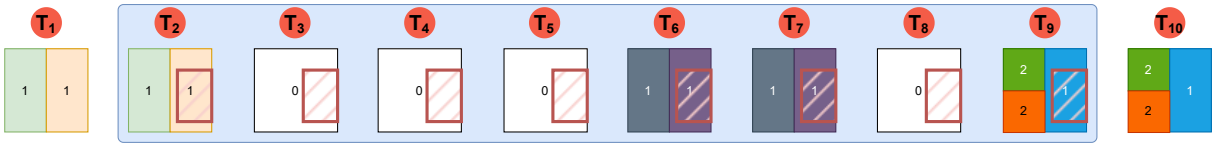

Figura 28 – Representação de um particionamento espaço-temporal de um determinado ROI.

Em algumas situações, pode existir particionamentos que cobrem parcial ou completamente o contexto de interesse. Nestes casos, é necessário um tratamento específico

para resolver as sobreposições. Primeiramente, identificam-se os particionamentos cobertos e verifica-se se são particionamentos utilizados; alguns particionamentos podem perder sua relevância ao longo do tempo, pois não são mais utilizados com seu potencial inicial. Deste modo, estes particionamentos podem ser revertidos através do operador *unsplit* (como visto na Seção 3.2), para permitir que possam ser realizados novos particionamentos espaço temporais.

A Figura 28 ilustra um fragmento espaço-temporal com o contexto de interesse. Este fragmento cobre o período $[T_1, T_{10}]$ sobre uma região espacial. Neste caso, existem 3 particionamentos sobre o fragmento, um de nível em $[T_1, T_2]$, outro de nível 1 em $[T_6, T_7]$, e mais um de nível 1 em $[T_9, T_10]$, por fim um de nível 2 que começa em T_{10} e avança por mais 4 instantes temporais. A princípio, seguindo as normas da Seção 3.4, temos o período $[T_3, T_5]$ como candidato a otimização.

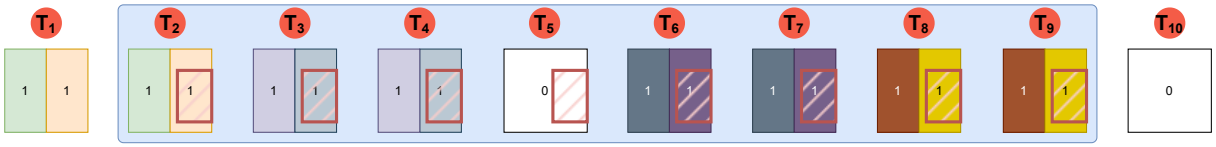

Figura 29 – Ilustração da otimização aplicada ao caso de particionamento espaço-temporal composto.

A Figura 29 ilustra a otimização aplicada ao caso de particionamento espaço-temporal composto. Neste contexto, verificou-se que os particionamentos espaço-temporais de nível 1 e outro de nível 2, ambos começando no instante temporal T_9, não se enquadravam como particionamentos relevantes. Então, estes particionamentos foram revertidos para particionamentos espaciais (nível 0). Portanto, existem dois períodos candidatos a otimização via particionamento: $[T_3, T_5]$ e $[T_8, T_9]$. Foram realizados dois particionamentos espaço-temporais, ambos de nível 1, em $[T_3, T_4]$ e $[T_8, T_9]$, resultando na redução do número de requisições de blocos, que inicialmente era de 7, para 5.

É importante salientar que a aplicação das normas (2º,3º e 4º) e as diretrizes é factível devido à equivalência espacial, pois, devido às restrições da KD-tree desta abordagem, temos particionamentos de mesmo nível em posições relativas com mesma forma e dimensão. Se duas KD-trees são construídas sobre matrizes de dimensão $M \times N$ usando a regra de "dividir sempre no meio" (alterando entre cortes verticais e horizontais com divisão inteira), então, para qualquer profundidade d, os blocos correspondem a regiões com a mesma forma, dimensões e alinhamento espacial.

3.6 Arquitetura do Dympas

Para garantir a efetividade da abordagem proposta, foi desenvolvido um sistema para gerenciar sua execução. A abordagem adotada, conforme ilustrado na Figura 30, visa otimizar o tempo de consulta com o uso adequado da memória primária e secundária conforme o necessário. Sua estrutura consiste em quatro módulos principais: `Interface`, `Orquestrador`, `Gerenciador de Índices` e `Gerenciador de Armazenamento`.

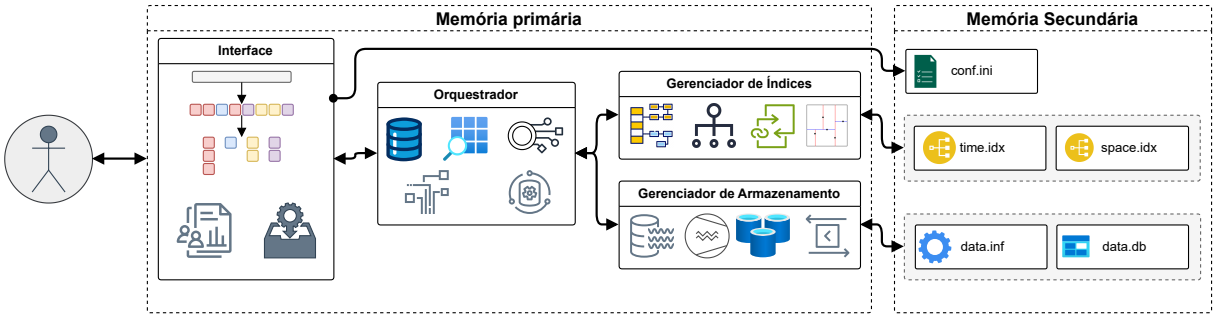

Figura 30 – Estrutura do Dympas.

O módulo `Interface` atua como ponto de entrada do sistema, responsável por receber, interpretar e validar os comandos do usuário. Suas atribuições incluem a inicialização dos recursos e a configuração de parâmetros operacionais, tais como: a alocação de memória para o cache; o tamanho máximo do armazenamento de blocos em memória secundária; a política de compressão (se ativa e o método a ser utilizado); o diretório de dados em memória secundária; o limiar de particionamento; e o tipo do pixel (inteiro de 1 byte, inteiro de 2 bytes, ponto flutuante de 4 bytes, etc.). Após a validação e estruturação desses parâmetros, a requisição é encaminhada ao módulo `Orquestrador` para processamento.

O módulo `Orquestrador` é o núcleo lógico do sistema, responsável por executar operações centrais como *split* e *unsplit*. Ele conduz as tarefas de alto nível, como: buscas, otimizações espaço-temporais, inserção de dados e preparação de resultados de consultas. Para isso, aciona os demais módulos de forma coordenada. A execução dessas operações é viabilizada pela delegação de responsabilidades: o `Orquestrador` requisita os índices necessários ao `Gerenciador de Índices` e solicita os blocos de dados correspondentes ao `Gerenciador de Armazenamento`.

O módulo `Gerenciador de Índices` controla o ciclo de vida das estruturas de indexação espaço-temporal. Sua operação consiste na carga dos índices da memória secundária para a primária na inicialização e efetuar sua persistência, atualizando os arquivos em memória secundária em cenários de modificação ou novas estruturas. A principal função deste módulo é fornecer ao `Orquestrador` a estrutura de índice requisitada para cada operação. Essa estrutura segue uma organização hierárquica: um índice temporal

por contexto do sistema (Dympas), cujos nós folha apontam para seus respectivos índices espaciais.

O módulo `Gerenciador de Armazenamento` atua como uma camada de abstração para o armazenamento físico, sendo o responsável pela gestão dos blocos de dados matriciais em memória secundária. Ele oferece uma interface transparente ao `Orquestrador`, que opera requisitando blocos por meio de um Identificador Único (UID). A partir deste UID, o módulo executa todo o ciclo de vida do bloco: realiza o mapeamento (via hash) para localizar o dado (onde ele começa e seu tamanho), gerencia sua compactação/descompactação conforme as configurações e, por fim, retorna, persiste ou remove os dados de interesse. De forma adicional, implementa uma camada de cache para otimizar o desempenho, mitigando a latência de I/O ao evitar acessos repetidos à memória secundária, o que é eficaz para dados espaço-temporais utilizados na abordagem proposta.

3.7 Gerenciamento de Blocos

O gerenciamento de blocos de dados é um componente para otimizar o desempenho de consultas no Dympas. Durante uma consulta espaço-temporal, o sistema reconstrói uma série cronológica de matrizes. Sem uma política de cache eficiente, os blocos de dados necessários para compor matrizes em instantes de tempo subsequentes seriam requisitados repetidamente da memória secundária. Dado que o acesso à memória secundária constitui um conhecido gargalo de desempenho, o uso de cache é uma estratégia para minimizar o tempo total da consulta.

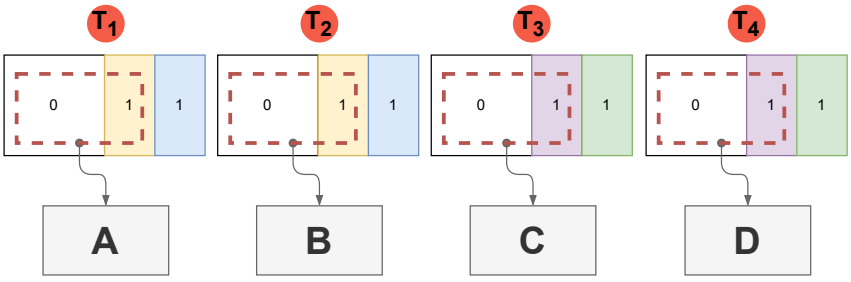

Figura 31 – Esquema de reconstrução de matrizes no Dympas.

A eficácia do cache se baseia na localidade temporal dos dados dentro dos blocos. A Figura 31 ilustra este princípio em uma consulta no intervalo $[T_1, T_4]$. Partições de nível 1 de mesma cor são armazenadas no mesmo bloco em memória secundária. Na reconstrução da matriz **A** (em T_1), o bloco amarelo é carregado. Este mesmo bloco contém dados necessários para a matriz **B** (em T_2). De forma análoga, o bloco lilás, carregado para a matriz **C**, também é utilizado na matriz **D**. Nesses dois casos, o uso de um cache evita acessos redundantes à memória secundária, validando a eficácia da abordagem.

Para implementar essa estratégia, cada bloco é unicamente identificado por um UID, através dele é usado um hash que mapeia sua localização e tamanho na memória secundária. O fluxo de recuperação de dados, ilustrado na Figura 32, segue uma política de cache explícita. Quando um bloco é solicitado, o sistema primeiro verifica sua presença na cache.

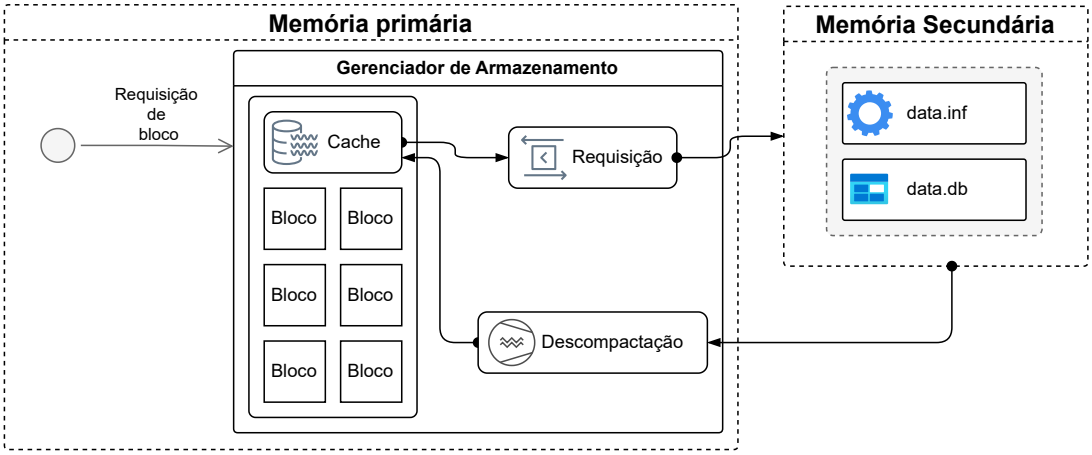

Figura 32 – Esquema do fluxo de recuperação de blocos no Dympas.

O processo de requisição de um bloco, identificado por seu UID, é gerenciado da seguinte forma:

- **Cache Hit:** Se o bloco existe na cache, ele é retornado diretamente ao solicitante e promovido ao início da estrutura de dados, seguindo a política de menos usado recentemente (LRU, do inglês ***L****east* ***R****ecently* ***U****sed*). A adoção da política LRU fundamenta-se na natureza das consultas, nas quais o particionamento espaço-temporal eleva a probabilidade estatística de reuso dos blocos mais recentes. Isso ocorre porque as consultas de janelas temporais favorecem o acesso às informações contidas nestes blocos em operações subsequentes à geração do dado atual.
- **Cache Miss:** Se o bloco não se encontra na cache, o sistema o localiza na memória secundária via tabela de hash. O bloco é lido e, se aplicável(conforme a configuração do sistema), descompactado. Em seguida, o bloco processado é inserido no topo da cache e retornado ao solicitante (módulo `Orquestrador`).

A eficácia do gerenciamento de blocos, como descrito, fundamenta-se na busca de dados de múltiplos instantes de tempo em um mesmo bloco. Contudo, tal arranjo dos dados em memória secundária não é arbitrário, mas uma consequência direta da estratégia de particionamento lógico seguindo uma estratégia pré-estabelecida. Portanto, para que a otimização via cache seja efetiva, o processo de particionamento deve obedecer às normas e diretrizes de alinhamento.

3.8 Algoritmos

Com a estrutura do sistema Dympas apresentada, esta seção se aprofunda na definição algorítmica de seus três métodos operacionais principais, responsáveis pelo desempenho e a consistência do sistema. Para assegurar a clareza e a reprodutibilidade, cada método é descrito por um algoritmo. O primeiro, a busca, focado na recuperação otimizada das informações de interesse requeridas. O segundo, o *split*, é responsável por aplicar um conjunto de particionamentos subsequentes em uma mesma região espaço-temporal e tem objetivo de refinar regiões de interesse e acelerar as consultas. Por fim, o *unsplit* é utilizado para reverter um particionamento espaço-temporal, seja para liberar recursos de otimizações que se tornaram desnecessárias ou para viabilizar novas partições prioritárias.

3.8.1 Busca

A busca espaço-temporal coordena a recuperação de dados do sistema. O algoritmo proposto opera em duas fases sequenciais: (I) varredura temporal e (II) análise espacial algoritmo 1. Inicialmente, uma B^+-tree filtra o período estipulado para identificar os *timestamps* com dados matriciais disponíveis. Cada *timestamp* atua como chave de seleção para a *KD-tree* correspondente. Esta estrutura executa a busca espacial para determinar quais partições e respectivos UIDs intersectam a região de interesse.

Algoritmo 1: Algoritmo de busca.

Input: ROI, start, end
Output: results

```
1  results ← [];
2  times ← B+Tree.findPeriod(start, end);
3  foreach T_i ∈ times do
4  |   spaceIndex ← getKDTree(T_i);
5  |   UIDs ← spaceIndex.search(ROI);
6  |   ROIFrame ← emptyMatrix(ROI);
7  |   foreach UID ∈ UIDs do
8  |   |   block ← DiskManager.load(UID); // Verifica a cache
9  |   |   chunk ← block.hashIndex.get(T_i);
10 |   |   ROIFrame.add(chunk);
11 |   end
12 |   results.append(ROIFrame, T_i);
13 end
14 return results;
```

Na sequência, para cada instante de tempo, o algoritmo verifica se o bloco de dados associado ao UID reside na memória principal. Caso contrário, o bloco é transferido da memória secundária. Uma busca baseada em *hash* determina o *offset* e o tamanho dos

dados no bloco, em que são inseridos na matriz de saída correspondente ao ROI. O processo iterativo abrange todos os instantes requisitados, reaproveitando blocos já alocados em memória que contenham múltiplos *timestamps*. Esta estratégia reduz a redundância de recuperação de dados da memória secundária e minimiza o custo de I/O da operação.

3.8.2 *Split*

A operação *split* visa aplicar particionamentos espaço-temporais em ROIs com alta demanda de acessos espaço-temporais. O algoritmo genérico é demonstrado no algoritmo 2, e permite a realização de múltiplos particionamentos sucessivos. Este método define um intervalo temporal para os particionamentos, mas não entra nos detalhes para verificar a validade dos particionamentos existentes. Os particionamentos são realizados de forma iterativa, em níveis de particionamento e períodos de particionamento.

Algoritmo 2: Algoritmo de *splits* consecutivos.

```
Input: start, level, topLeft, bottomRight
1  length ← 2^level;
2  dateRange ← getPeriod(start, length);
3  spaceIndexes ← loadSpaceIndexes(dateRange);
4  foreach i ∈ [0, level) do
5  |  serieLength ← 2^(i+1);
6  |  for j ← 0 to length − 1 step serieLength do
7  |  |  spaceNodes ← loadNodes(spaceIndexes, j, serieLength, topLeft,
   |  |   bottomRight);
8  |  |  nodesPerSplice ← getNodesPerSlice(spaceNodes);
9  |  |  subDataRange ← copy(dataRange + j, dataRange + j + serieLength);
   |  |     // Faz a cópia de um intervalo de datas
10 |  |  serieNodes ← transposeNodes(spaceNodes);
11 |  |  foreach nodes ∈ serieNodes do
12 |  |  |  split(nodes, subDataRange);// Split unitário
13 |  |  end
14 |  end
15 end
```

A Figura 33 ilustra a aplicação de particionamentos subsequentes em um fragmento espaço-temporal. Para alcançar um particionamento de nível 2, que requer $2^2 = 4$ instantes temporais sucessivos, é necessário considerar o período temporal inicia (T_1) como ponto de partida. Como demonstrado em Figura 33A, as regiões com particionamentos válidos não geram conflitos com o particionamento de nível 2. Em Figura 33B, a primeira etapa de particionamento envolve um particionamento de nível 1 no intervalo $[T_3, T_4]$. Finalmente é aplicado um particionamento de nível 2 no intervalo $[T_1, T_4]$, ilustrado na Figura 33C. Desta forma, múltiplos *splits* sucessivos são realizados, aumentando o grau de particionamento a cada iteração.

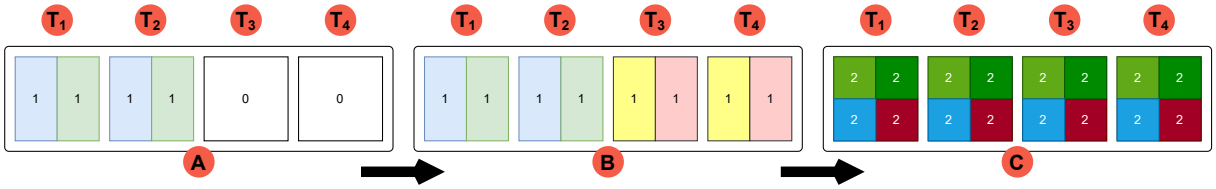

Figura 33 – Aplicação de *splits* subsequentes em um fragmento espaço-temporal.

O algoritmo 3 demonstra o *split* unitário que verifica se a geometria dos particionamentos envolvidos otimiza a mesma região com posição relativa nos diferentes períodos temporais. Em seguida, é efetuado o particionamento dos dados (algoritmo 4), gerando dois novos blocos com o novo particionamento. Posteriormente, é realizado o particionamento lógico no indexamento espacial para garantir o mapeamento correto dos novos blocos. Por fim, é realizada a atualização das informações nos novos blocos.

Algoritmo 3: Algoritmo de *splits* unitário.

Input: nodes, dates

```
1 checkNodeSerieGeometry(nodes);
2 [UID1, UID2] ← splitData(nodes, dates);
3 [bb1, bb2] ← divideBoundingBox(nodes);
4 foreach node ∈ nodes do
5 |   [firstNode, secondNode] ← splitKDSpaceNode(node);
6 |   firstNode.UID ← UID1;
7 |   secondNode.UID ← UID2;
8 end
```

O algoritmo 4 é responsável por realizar o particionamento e organização dos dados persistidos em memória secundária. São definidos os parâmetros do particionamento, as delimitações dos novos particionamentos e o período temporal. Os blocos envolvidos no particionamento ($2^{\text{nível}}$) são requisitados para a memória primária, onde são divididos conforme os parâmetros. Após cada dado ser particionado, é inserido em um dos dois novos blocos, informando seu *timestamp* para realizar a indexação hash *in loco*. Por fim, os dois novos blocos são persistidos e seus respectivos UIDs são retornados pela função.

3.8.3 *Unsplit*

A operação de *unsplit* reverte particionamento espaço-temporais (*splits*) prévios. O objetivo principal consiste em consolidar áreas provenientes de dois particionamentos espaço-temporais, reorganizando os dados de forma que a quantidade de informações em cada bloco seja mantida. Especificamente, a operação visa criar dois novos particionamentos, ambos espaço-temporais ou apenas espaciais.

Algoritmo 4: Algoritmo para fazer o *split* dos dados.

```
Input: nodes, dates
1  [width, height] ← getDimention(nodes);
2  [bb1, bb2] ← divideBoundingBox(nodes);
3  UIDs ← [];
4  hashIndex1, hashIndex2;
5  for i ← 0 to (nodes.size() − 1) do
6  |  date ← dates[i];
7  |  node ← nodes[i].UID;
8  |  data ← diskManager.Read(UID);
9  |  temporalInstantDate ← Data.getData(date);
10 |  [date1, size1] ← getSubMatrix(data, width, height, bb1);
11 |  [date2, size2] ← getSubMatrix(data, width, height, bb2);
12 |  hashIndex1.insert(data1, size1, date);
13 |  hashIndex2.insert(data2, size2, date);
14 end
15 UID1 ← hashIndex1.save();
16 UID2 ← hashIndex2.save();
17 return [UID1, UID2];
```

Dados dois particionamentos espaço-temporais que começam em δ e termina em γ com tamanho espacial k. Os particionamentos são reorganizados de forma que suas áreas são somadas e a quantidade de dados com diferentes instantes temporais é reduzido pela metade. Então, obtém-se dois particionamentos (espaço-temporal ou apenas espacial) com dados que representam uma área de tamanho $2 \times k$. Onde o primeiro começa no período δ e termina em $\frac{\gamma+\delta}{2}$. Já o segundo particionamento começa em $\frac{\gamma+\delta}{2} + 1$ e termina em γ. Essa operação permite reverter operações de *split*, voltando ao estado anterior ao *split*. Portando, a estratégia é mesclar a área de dois particionamentos e reorganizar os dados de forma que os blocos contenham a mesma quantidade de informação.

O algoritmo 5 ilustra de forma genérica o funcionamento do algoritmo *unsplit*. Este algoritmo aplica-se unitariamente, ou seja, quando uma região é identificada como espacial, ele é revertido, assumindo que se trata de um particionamento espaço-temporal.

Na entrada do algoritmo 5 é fornecido um instante temporal dentro do período de um particionamento espaço-temporal em que se deseja aplicar o *unsplit*, juntamente com uma região espacial onde são encontrados os particionamentos que serão operados. O parâmetro "level" define o objetivo da operação, ou seja, um nível que se pretende alcançar após a aplicação de um *unsplit*. Ao identificar os particionamentos a serem revertidos, é realizada a remoção das subárvores do indexamento espacial, consolidando os dados e organizando os blocos. Em seguida, os dados são persistidos, e os índices de indexação são atualizados, finalizando a operação *unsplit*.

A Figura 34 demonstra um exemplo do processo de *unsplit*, onde um fragmento

Algoritmo 5: Algoritmo de *unsplits* unitário.

```
Input: start, level, topLeft, bottomRight
1  nodes ← searchKDSpaceNodes(topLeft, bottomRight);
2  for j ← 0 to nodes.size() − 1 do
3  |   block ← diskManager.read(nodes[i].UID);
4  |   dateRange ← block.getDataRange();
5  |   spaceIndexes ← loadSpaceIndexes(dataRange);
6  |   for j ← 0 to spaceIndexes.size() − 1 do
7  |   |   spaceIndexes[j].removeSubTreeByLevel(topLeft, bottomRightm, level);
8  |   |   spaceIndexes[j].save();
9  |   end
10 |   indexDataManager.write();
11 end
```

espaço-temporal compreendido no intervalo $[T_1, T_4]$ é dividido em três estados distintos (A-C), sendo o estado A representação original. Especificamente, a Figura 34 A ilustra o particionamento de nível 2, parte de uma estratégia de aplicação do operador *unsplit* duas vezes, visando a obtenção dos particionamentos espaço-temporais para nível 0 ao final das operações.

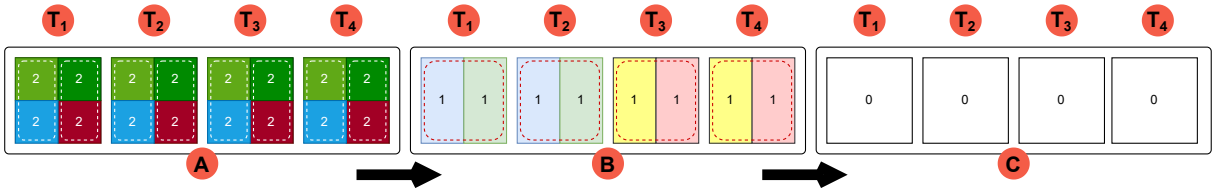

Figura 34 – Aplicação de *unsplits* subsequentes em um fragmento espaço-temporal.

Inicialmente, observa-se a criação de quatro particionamentos espaço-temporais de nível 2, originados em T_1 e progredindo até T_4. Posteriormente, é necessária a definição do agrupamento dos particionamentos, seguindo a orientação original do sistema (vertical ou horizontal, conforme o nível). Por exemplo, se um particionamento α originou β usando a orientação vertical, ao aplicar o *unsplit* em β, é utilizada a orientação vertical para agrupar os resultados, gerando quatro particionamentos de nível 1, dois iniciados em T_1 e dois em T_3. Essa reorganização permite que cada região espacial contenha informações temporais correspondentes há apenas dois instantes.

A etapa subsequente, segunda execução do operador *unsplit*, envolve a junção dos particionamentos de nível 1 em pares, resultando na formação de blocos de nível 0, onde cada bloco contém dados referentes a um único período temporal. Após essa consolidação, os particionamentos são encerrados e as árvores KD-tree associadas são atualizadas, além de ajustes na alocação dos blocos de dados em memória secundária.

4 EXPERIMENTOS

Este capítulo detalha os procedimentos adotados para a avaliação do método proposto (Dympas). É descrita a carga de trabalho dos testes (*workload*), os conjuntos de dados e etapas de preparação, a configuração do ambiente de testes e as etapas de análise dos resultados.O objetivo central é avaliar quantitativamente o particionamento espacial (particionamento nível 0 no Dympas), normalmente aplicados em GIS, e comparar com as estratégias propostas do Dympas de particionamento espaço-temporais (particionamentos nível > 0). Além disso, são apresentadas as métricas definidas para quantificar o desempenho dos testes realizados, para permitir a reprodutibilidade e a consistência dos experimentos.

A carga de trabalho foi estruturada em duas etapas. Inicialmente, avaliou-se o comportamento do Dympas em um Cenário de Uso Real, simulando condições operacionais típicas. Na sequência, realizou-se o Cenário Sintético, explorando variações controladas de parâmetros para analisar a robustez e o desempenho do sistema sob diversas condições de estresse

4.1 Cenário de Uso Real

Esta etapa descreve a configuração de um ambiente experimental projetado para simular condições operacionais reais, onde a aplicação de consultas espaço-temporais representa um diferencial estratégico. A delimitação experimental é baseada em análises geoespaciais de índices de vegetação voltados à agricultura de precisão. Para viabilizar a análise, os dados brutos foram submetidos as etapas de pré-processamento, visando aproximar das operações realizadas em um caso de uso real. Adicionalmente, foram delimitadas ROIs condizentes com os padrões de sensoriamento remoto. Esta abordagem permite uma avaliação comparativa rigorosa entre as consultas convencionais em GIS (nível 0 do Dympas) e as consultas espaço-temporais, possibilitando a quantificação do ganho de desempenho e da precisão nos resultados obtidos.

4.1.1 Carga de Trabalho

A carga de trabalho usada nos testes é composta por operações espaço-temporais típicas em tarefas de análise e extração de características para aprendizado de máquina. As operações recuperam dados de uma dada região de interesse ao longo de uma série de instantes de temporais consecutivos. As operações são subdividas em três tipos: (*I*) janela aleatória (JA); (*II*) janela deslizante (JD); e (*III*) instante temporal (IT).

As operações de janela aleatória recuperam dados de um ROI fornecido de intervalos

variados de instantes de tempo, representando consultas isoladas para análise espaço-temporal. Já as consultas de janela deslizante recuperam dados de um mesmo ROI em instantes subsequentes de tempo, de forma iterativa, por exemplo, para calcular estatísticas móveis, como o acumulado de precipitação dos últimos 15 dias para cada dia de um intervalo de interesse. Uma aplicação prática dessa operação é o acompanhamento meteorológico em regiões agronômicas, o que possibilita a extração de informações valiosas para orientar o manejo de patologias como a ferrugem asiática. Essa abordagem pode gerar um impacto econômico significativo, ao reduzir os custos com o controle dos patógenos por meio de intervenções mais precisas, além de aumentar a produtividade. Para calcular a precipitação acumulada em 15 dias em uma ROI para um dado dia, é preciso consultar uma janela de 15 instantes de tempo. Por exemplo, a precipitação acumulada do dia 16 exige operar sobre os dados do intervalo de dias $[T_1, T_{15}]$. De forma similar, a precipitação acumulada do dia 17 opera sobre o intervalo $[T_2, T_{16}]$, o que corresponde a uma operação semelhante à anterior, mas deslizando a janela temporal em um dia. As consultas por janela deslizante permitem um uso agressivo da *cache* de blocos, pois a mesma ROI é examinada à medida que o intervalo temporal é varrido sequencialmente do início ao fim, com acesso a um único bloco novo da memória secundária a cada deslizamento da janela. Em contrapartida, as consultas por janela aleatória até podem beneficiar-se da *cache* de blocos, mas isso ocorre de forma menos frequente. Por fim, as operações de instante temporal recuperam dados de um ROI fornecido para um instante de tempo.

A carga de trabalho é baseada um conjunto de dados de sensoriamento remoto do Sentinel-2 [55] e em geometrias de talhões agrícolas registrados na Agência Nacional de Águas (ANA), para simular operações em um ambiente real. Foi utilizada a região de código `T22KDV` do Sentinel-2, que cobre uma área de $100km \times 100km$ do norte do estado do Paraná, de maio de 2017 até agosto de 2024. Foi usada a banda espectral de 490nm com resolução espacial de 10 metros. Os dados foram filtrados com a restrição de terem menos de 10% de nuvem, o que é uma decisão muito comum em situações reais, resultando em uma série temporal com 68 arquivos do tipo *raster* no intervalo. Cada *raster* tem dimensão de 10980×10980 pixeis e precisão de 2 bytes por pixel. Para as ROI, foram utilizados os dados da geometria de talhões cadastrados na ANA contidos na região `T22KDV` do Sentinel-2, totalizando 78 talhões. As ROIs foram definidas como o MBR dos talhões. O tamanho médio das ROIs é 90×90 pixeis, sendo que a maior ROI tem as dimensões de 141×37 pixeis e a menor 30×37 pixeis.

As operações, tanto de janela aleatória quanto deslizante, foram definidas para períodos de 7 meses. Contudo, o número de rasters da série contida em diferentes intervalos de 7 meses varia, pois há períodos com maior ou menor ocorrência de nuvens, variando o número de *rasters* com 10% ou mais de nuvem descartados. Embora em aplicações reais em muitos casos busquem-se alternativas de interpolação de dados faltantes, devido à cobertura por nuvens, para o experimento não foi considerada essa possibilidade, pois há

interesse em que haja variação no tamanho da janela, para gerar uma carga de trabalho mais diversa. A quantidade de *rasters* dentro dessas janelas varia de 2 (muita ocorrência de nuvens) a 7 (sem descarte por cobertura de nuvens), com um tamanho médio de 5,24 *rasters*. A cada nova janela, há um acréscimo de 1 mês no início, em relação à janela anterior. Assim, para cada ROI, os dados permitiram produzir 82 janelas de maio de 2017 até agosto de 2024.

Nas consultas de janela aleatória, foram geradas combinações exaustivas de ROI e período temporal da janela. Já para as operações com janelas deslizantes, a consulta é realizada sobre um ROI sequencialmente sobre todas as janelas até cobrir toda a abrangência temporal dos dados, resultando em 82 consultas, uma para cada ROI. As consultas de instante temporal geram combinações de todos os instantes de tempo para todos os ROIs

4.1.2 Configuração dos Experimentos

O objetivo principal dos testes foi avaliar a variação do tempo das consultas usando o método proposto em comparação ao particionamento exclusivamente espacial. Por particionamento exclusivamente espacial, entenda-se o particionamento inicial adotado na proposta desse trabalho, em que o dado matricial de cada instante de tempo é subdividido iterativamente usando a estratégia da KD-tree (particionamentos verticais e horizontais alternados) até alcançar um limiar pré-definido, que estabelece o tamanho da região a ser depositada de forma comprimida em um bloco (cf. Seção 3.2). Nos resultados, o particionamento exclusivamente espacial é denotado como *split nível 0*.

Os testes consideraram variações no tamanho de bloco, com base no limiar de particionamento, para analisar o comportamento das estratégias mediante de diferentes tamanhos de bloco. Os limiares avaliados foram 5625, 22500, 90000 e 360000, além de avaliar os dados compactados com a metodologia `zstd` e não compactados.

A carga de trabalho foi submetida para as diferentes configurações de limiar, considerando níveis crescentes de otimização espaço-temporal. Ressalte-se que a proposta desse trabalho permite o ajuste dinâmico do particionamento espaço-temporal, contudo, neste trabalho, as operações de *split* foram invocadas manualmente e de forma sucessiva, como tarefas de otimização de bancos de dados executadas por um administrador com base em sua análise e expertise a respeito da carga de trabalho. Isto é, a primeira execução da carga de trabalho considerou o particionamento exclusivamente espacial (*split* nível 0). Em seguida, foi invocado o *split* espaço-temporal em todos os blocos com região interceptada por algum ROI, subdividindo os blocos envolvidos em um *split* de nível 1 (agrupando dados de 2 instantes de tempo), e a carga de trabalho foi executada novamente. O mesmo procedimento foi realizado para *splits* de nível 2 e 4, agrupando dados de 4 e 16 instantes de tempo, respectivamente.

Para avaliar o benefício da *cache* de blocos, foram considerados casos típicos em dois extremos: casos sem qualquer aproveitamento de *cache* e casos com ótimo aproveitamento de *cache*. Para avaliar os casos sem aproveitamento de *cache*, foram utilizadas operações de janela aleatória, limpando a *cache* ao término de cada operação. Esse caso é típico quando se deseja executar uma operação espaço-temporal para realizar uma análise isolada, seja para uma região ou um intervalo não consultados anteriormente, portanto, os blocos necessários não deverão estar em *cache*. Um exemplo de operações no outro extremo são as operações de janela deslizante, que são altamente beneficiadas pela *cache*. Para avaliar este caso, nos experimentos com operações de janela deslizante não foi feita limpeza da *cache* de blocos. Eventuais retiradas de blocos da *cache* foram realizadas exclusivamente pelo mecanismo de liberação de *frames*, que foi implementado no Dympas utilizando a política LRU.As consultas de instante temporal, representa consulta pontuais, para verificar o desempenho das consultas ao adaptar ao conjunto de dados, através *splits*, para consultas espaço-temporais.

4.2 Cenário Sintético

O cenário sintético foi elaborado para testar o sistema Dympas em situações que os dados do cenário real não permitiu. Enquanto o cenário real foca no uso prático próximo a um caso de uso, o cenário sintético serve para ampliar os testes e averiguar a robustez do sistema e entender como ele se comporta na mudança drástica do tamanho das áreas consultadas, a quantidade de dados e o tempo das análises. Essa abordagem ajuda a verificar se a lógica de particionamento espaço-temporal continua eficiente e rápida mesmo em condições extremas ou variadas.

4.2.1 Carga de Trabalho

As operações aplicadas na carga de trabalho para este cenário seguem a mesma lógica apresentada no cenário de uso real (Subseção 4.1.1), porém são aplicadas a uma base de dados distinta para estender a análise experimental. Para isso, foram utilizados os mesmos limiares de particionamento: 5625, 22500, 90000 e 360000. As ROIs foram categorizadas em três grupos de tamanho para testar diferentes escalas de consulta: o Tipo 0 (pequenas, de 1 a 266 pixels), o Tipo 1 (médias, de 270 a 1134 pixels) e o Tipo 2 (grandes, de 1136 a 8309675 pixels). O número de matrizes processadas por janela temporal foi controlado para permitir testes com intervalos fixos de 2, 5, 9, 14 e 20 instantes. Para cada configuração, avaliou-se o desempenho comparando dados não compactados com dados compactados por meio do algoritmo `zstd`, mantendo o rigor das medições realizadas anteriormente.

A carga de trabalho baseia-se em dados de sensoriamento remoto do Sentinel-2 [55] e em geometrias de imóveis do Sistema de Cadastro Ambiental Rural (SICAR).

Foi utilizada a região de código `T22KGU` (Paranapanema–SP), abrangendo o período de janeiro de 2017 a abril de 2025. Utilizou-se a banda espectral de 865 nm com resolução espacial de 20 metros. Foram processados 570 dados matriciais, cada um com dimensão de 5.490×5.490 pixels e precisão de 2 bytes por pixel. Para as ROIs, as geometrias do SICAR representam propriedades rurais reais, o que permite verificar como o sistema lida com formas geográficas variadas em uma escala muito maior do que os talhões da ANA utilizados no cenário real.

4.2.2 Configuração dos Experimentos

O objetivo principal destes testes foi validar os mesmos parâmetros avaliados no primeiro cenário, mas expandindo as análises para situações que os dados reais, por sua natureza, não abrangem. Com essa abordagem, foi possível verificar o comportamento do Dympas sob condições de uso mais severas. Uma das principais diferenças em relação ao cenário anterior é o uso de janelas temporais de tamanhos fixos em uma consulta (variando em diferentes testes). Assim, permite observar o desempenho do sistema em análises temporais de longa duração, como ocorre no cálculo de acumulados históricos. Além disso, a variação controlada no tamanho das ROIs permite analisar como o particionamento se comporta em diferentes densidades. Em regiões pequenas (Tipo 0), o sistema realiza menos particionamentos; já em regiões grandes (Tipo 2), o particionamento ocorre prioritariamente nas bordas das geometrias, pois o centro da área de interesse já está totalmente coberto por blocos de dados, eliminando a necessidade de novos *splits*.

Os conjuntos de dados também foram ajustados para permitir uma visão detalhada do comportamento do sistema com parâmetros técnicos diferentes dos usados no cenário de uso real. A principal modificação é o uso de uma resolução espacial menor (20 metros), o que gera arquivos matriciais com dimensões reduzidas em comparação ao primeiro cenário. Essa mudança é estratégica, pois permite realizar uma análise focada no tempo de inserção dos dados, na velocidade das operações de particionamento e, principalmente, no impacto da compactação `zstd` sobre diferentes volumes de dados. Assim, os experimentos buscam demonstrar que o método proposto mantém sua eficiência mesmo quando as características dos dados e das áreas de consulta são alteradas expressivamente.

4.3 Especificações Experimentais e Ambiente de Teste

A Tabela 2 sumariza as configurações adotadas em ambos os cenários, evidenciando as distinções entre os conjuntos de dados reais e sintéticos. A avaliação experimental foca na análise de desempenho considerando: persistência em memória secundária (com e sem compressão), variações na dimensão das ROIs, extensões das janelas temporais, limiares de particionamento e classes de consultas matriciais. Uma consulta de uma janela equivale à recuperação de todos os dados matriciais (um para cada instante de tempo que

existe dados) no período requisitado, não a métrica de cada dado recuperado. No cenário real, uma consulta de janela varia a quantidade de dados recuperados para cada período. Para o cenário sintético, esses dados são definidos conforme o tamanho da janela, que representa exatamente a quantidade de dados matriciais recuperados. O instante temporal é a recuperação de um único dado matricial.

Tabela 2 – Comparativo entre os conjuntos de dados

Parâmetro	**Caso de Uso Real**	**Caso Sintético**
Conjunto de dados de Geometrias	ANA	SICAR
Conjunto de dados de Dados Matriciais	SENTINEL-2	SENTINEL-2
Região	T22KDV	T22KGU
Banda Espectral	490nm	865nm
Quantidade de Matrizes	68	570
Resolução Espacial	10m	20m
Precisão por Pixel	2 bytes	2 bytes
Variedade de ROIs	1 tipo	3 tipos
Tamanho de Janelas	Média de 5,24	2, 5, 9, 14, 20
Dimensão da Matriz	10980 × 10980 pixels	5490 × 5490 pixels
Níveis de Particionamento	0, 1, 2 e 4	0, 1, 2, 3 e 4
Limiares de Particionamento	5625, 22500, 90000 e 360000	5625, 22500, 90000 e 360000

Ademais, todos os testes foram realizados em um mesmo ambiente de desenvolvimento, implementados na mesma base de código em C++ e com execução em um mesmo equipamento. Especificamente, o sistema implantado e testado utiliza memória secundária do tipo memória secundária rígido. O computador utilizado é equipado com processador Intel Core i7-9700, com 32GB de RAM DDR4, HDD SATA 2TB de 7.200RPM e sistema operacional Ubuntu GNU/Linux 24.04 64 bits. O ambiente foi exclusivamente utilizado para estes testes para evitar interferências no gerenciamento dos dados do sistema. Além disso, a *cache* do sistema de arquivos utilizado pelo sistema operacional foi esvaziada antes de cada execução. Para o cálculo de redução de tempo, foi utilizada a fórmula:

$$\text{Redução } (\%) = \left(\frac{V_i - V_f}{V_i} \right) \times 100 \tag{4.1}$$

Onde:

- V_i: Valor (tempo) inicial.

- V_f: Valor (tempo) final.

São utilizadas 2 formulas para o cálculo de taxa de uso de memória. A primeira fórmula utilizada para medir a proporção direta entre o estado final e o inicial:

$$T_{geral} = \left(\frac{M_f}{M_i}\right) \times 100 \tag{4.2}$$

A segunda fórmula é utilizada para garantir a estabilidade numérica em casos onde $M_i \to 0$ (como em cenários de alto uso de cache):

$$T_{norm} = \left(\frac{M_f}{M_f + M_i}\right) \times 100 \tag{4.3}$$

Onde:

- M_i: Memória Inicial (ou valor de referência).
- M_f: Memória Final (ou valor de uso/saída).

4.4 Resultados e Discussão

Esta seção apresenta os resultados da análise experimental do sistema Dympas, integrando os resultados obtidos tanto no cenário real quanto no sintético para oferecer uma compreensão abrangente do impacto do método proposto. A aplicação da metodologia resultou em uma melhoria de desempenho relevante, com reduções globais no tempo de consulta que atingiram 81% nos testes realísticos e até $92,3\%$ no contexto controlado (sintético). Este alto rendimento mostra que o particionamento espaço-temporal, quando devidamente calibrado, consegue reduzir de forma drástica o custo de recuperação de séries de dados matriciais, embora o grau de otimização varie conforme a previsibilidade e a extensão temporal da carga de trabalho.

A análise da infraestrutura revela uma relação elementar entre a economia de armazenamento e a agilidade no processamento e persistência dos dados matriciais. O emprego do algoritmo de compressão `zstd` proporcionou uma redução na ocupação de memória secundária, alcançando uma eficiência volumétrica de até $41,9\%$ no cenário real e $23,77\%$ no sintético. Entretanto, essa economia impõe um custo computacional durante a persistência, elevando o tempo médio de inserção em aproximadamente seis vezes no cenário real e em $69,07\%$ no sintético. Assim, caracteriza o sistema como altamente dependente da largura de banda de I/O e do poder de processamento para operações de descompressão em tempo real.

Em relação à recuperação de dados espaço-temporais, os resultados demonstram que a eficácia da estrutura é dada pela interação entre o nível de particionamento e a natureza da consulta. Para consultas de janela deslizante, a taxa de recuperação e o desempenho temporal escalam positivamente com o incremento do nível de particionamento, atingindo patamares de eficiência superiores a 94% no Nível 4, devido ao benefício do reúso intensivo de *cache* em séries temporais. Entretanto, consultas de janela aleatória e instantes temporais encontram seus pontos de equilíbrio em níveis intermediários ou iniciais, visto que o particionamento para níveis mais elevados introduz um overhead de gerenciamento de blocos de dados e metadados que pode degradar a latência em até $19,7\%$ quando comparada a níveis menores.

Por fim, o limiar de particionamento se mostrou como o parâmetro individual de maior impacto na escalabilidade do sistema. Enquanto o limiar de 360000 apresentou os resultados mais robustos e eficientes para a complexidade do cenário real, o limiar de 90000 se apresentou como o ponto de equilíbrio ideal no cenário sintético, equilibrando a agilidade de ingestão de dados com a baixa latência na recuperação em todos os regimes avaliados. Essa necessidade de calibração estratégica indica que o desempenho máximo do Dympas não é estático, mas sim o resultado de um ajuste fino entre o nível do particionamento e a densidade de compactação dos blocos físicos.

4.4.1 Caso de Uso Real

Inicialmente serão descritos os resultados baseados nos testes conduzidos em um cenário realístico de aplicação da metodologia proposta. Deste modo, permite avaliar os resultados em condições que ocorrem em análise geoespacial que faz uso de consultas espaço-temporais.

4.4.1.1 Persistência e Particionamento

Esta seção caracteriza os aspectos de infraestrutura do Dympas, com foco em persistência e particionamento. São analisados o consumo de memória secundária, o tempo médio de ingestão de dados matriciais e a latência do particionamento espaço-temporal. Os resultados detalhados encontram-se no Apêndice (Seção A.1).

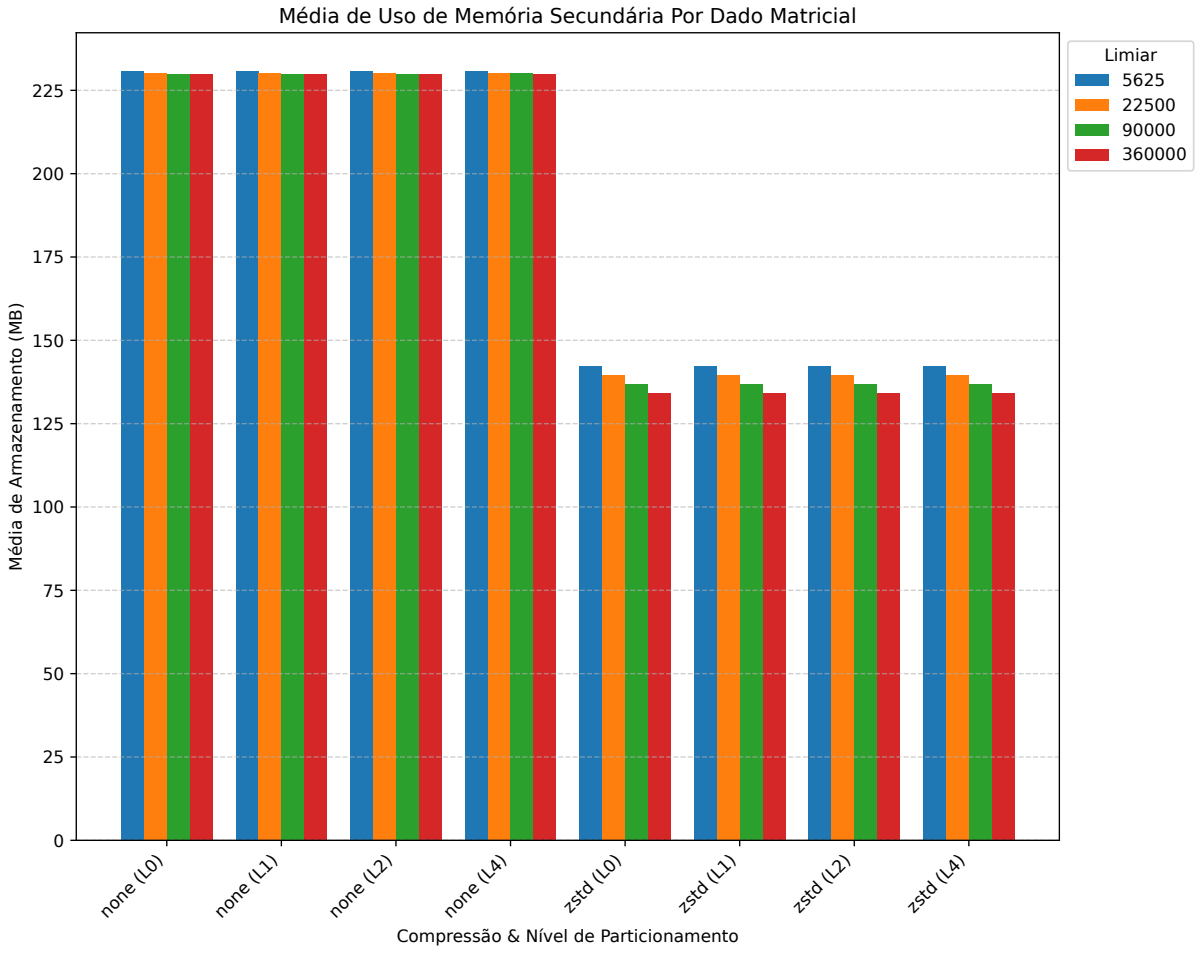

Figura 35 – Tamanho médio de uso da memória secundário por dado matricial inserido no Dympas.

A análise dos resultados apresentados na Figura 35 revela uma diferença significativa entre os métodos de compactação de dados (`zstd`) e o não comprimido (`none`). Para cada

inserção de conjunto de dados e particionamentos realizados, foi feita a medição do consumo de memória secundária por dado matricial, obtendo-se a média para cada configuração de tipo de compactação, nível de particionamento e limiar. O uso do algoritmo `zstd` resultou em uma economia substancial de memória secundária. Essa redução de armazenamento é expressiva em cenários de dados massivos. Conforme o Apêndice A (Tabela 3), o tamanho médio dos dados comprimidos varia entre $134,07$ MB e $142,34$ MB, em contraste com os aproximadamente 230 MB do formato não comprimido. Para o limiar de 360000 com nível 4 de particionamento, a redução atingiu $41,7\%$. No entanto, essa compressão impõe um custo computacional adicional, resultando em um aumento no tempo de inserção, conforme detalhado no Apêndice A (Tabela 4).

O consumo médio para dados sem compressão (`none`) foi de aproximadamente $230,20$ MB, enquanto os dados comprimidos (`zstd`) ocuparam, em média, $137,95$ MB. Isso representa uma redução média de $40,07\%$ ao utilizar a compactação, independentemente do limiar ou nível de particionamento. A configuração de maior ocupação foi observada no cenário sem compressão com limiar de 5625 e nível de particionamento 4 ($230,74$ MB), enquanto o cenário de maior eficiência volumétrica ocorreu com limiar de 360000 e nível 4 utilizando `zstd` ($134,07$ MB), totalizando uma redução máxima de $41,9\%$. O aumento do limiar de 5625 para 360000 nos dados sem compressão resultou em uma redução de apenas $0,32\%$ no tamanho do arquivo, indicando que o tamanho do bloco isoladamente tem pouco impacto no volume total quando não há compressão.

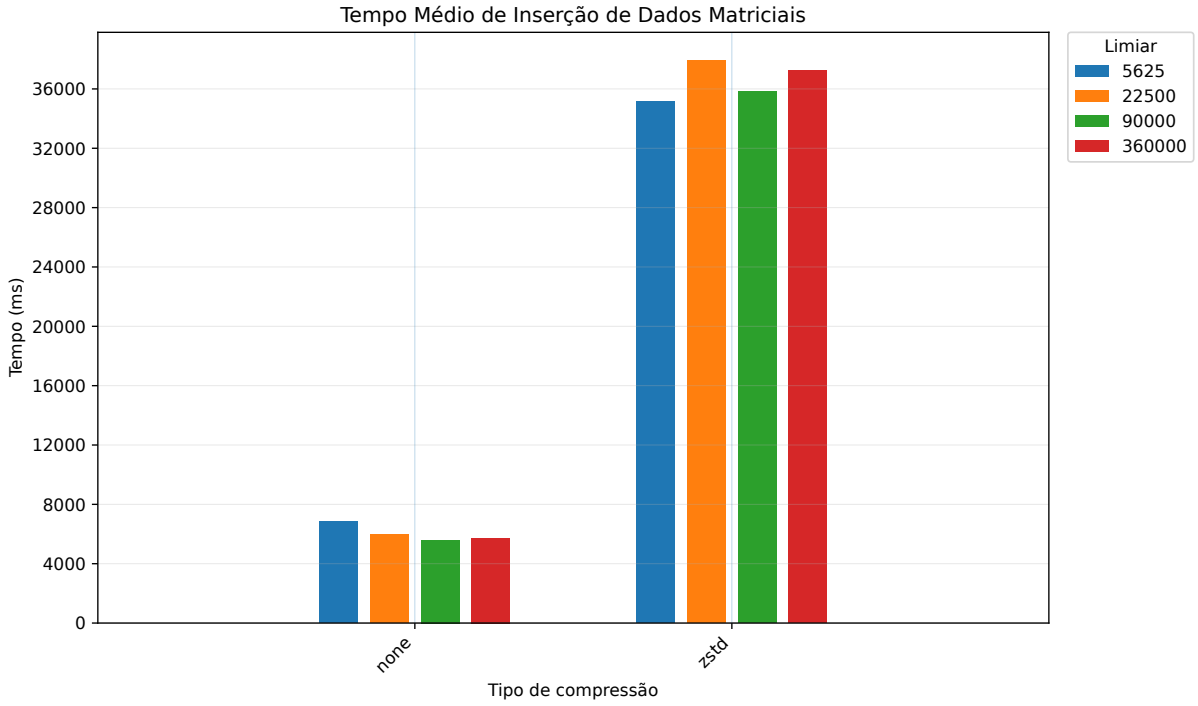

Figura 36 – Tempo médio de inserção de dados matriciais no Dympas por tipo de compressão e limiar.

A Figura 36 apresenta o tempo médio necessário para a inserção dos dados matriciais no Dympas. O tempo gasto para a persistência de cada dado matricial é exibido em milissegundos (ms), comparando o uso de dados comprimidos (`zstd`) e não comprimidos (`none`). Para cada método, foram avaliados os limiares de 5625, 22500, 90000 e 360000. Como neste contexto tempos menores indicam melhor desempenho, observa-se que os dados não comprimidos apresentaram uma redução de até $85,23\%$ no tempo de inserção em comparação aos dados comprimidos. O maior tempo de execução foi registrado na configuração com limiar de 22500 utilizando `zstd` (37918 ms), enquanto o menor tempo ocorreu com o limiar de 90000 sem compressão (5600 ms). No cenário sem compressão, o aumento do limiar de 5625 para 90000 gerou uma melhora de $18,07\%$, embora o tempo tenha voltado a subir levemente no limiar de 360000.

A comparação entre as de inserção e limiar revela que o particionamento dos dados possui um custo pequeno em relação ao processo de compressão. Em termos médios, a organização lógica dos blocos representa apenas entre $5,40\%$ e $6,27\%$ do tempo de persistência dos dados. Pode se notar que o impacto do algoritmo `zstd` é o principal fator que define o desempenho, elevando o custo de inserção em aproximadamente 6 vezes quando comparado ao formato não comprimido `none`. No entanto, a baixa proporção do tempo de particionamento diante da inserção indica que o principal gargalo do sistema está na etapa de persistência e nos cálculos de compactação, e não na lógica de divisão da estrutura de dados.

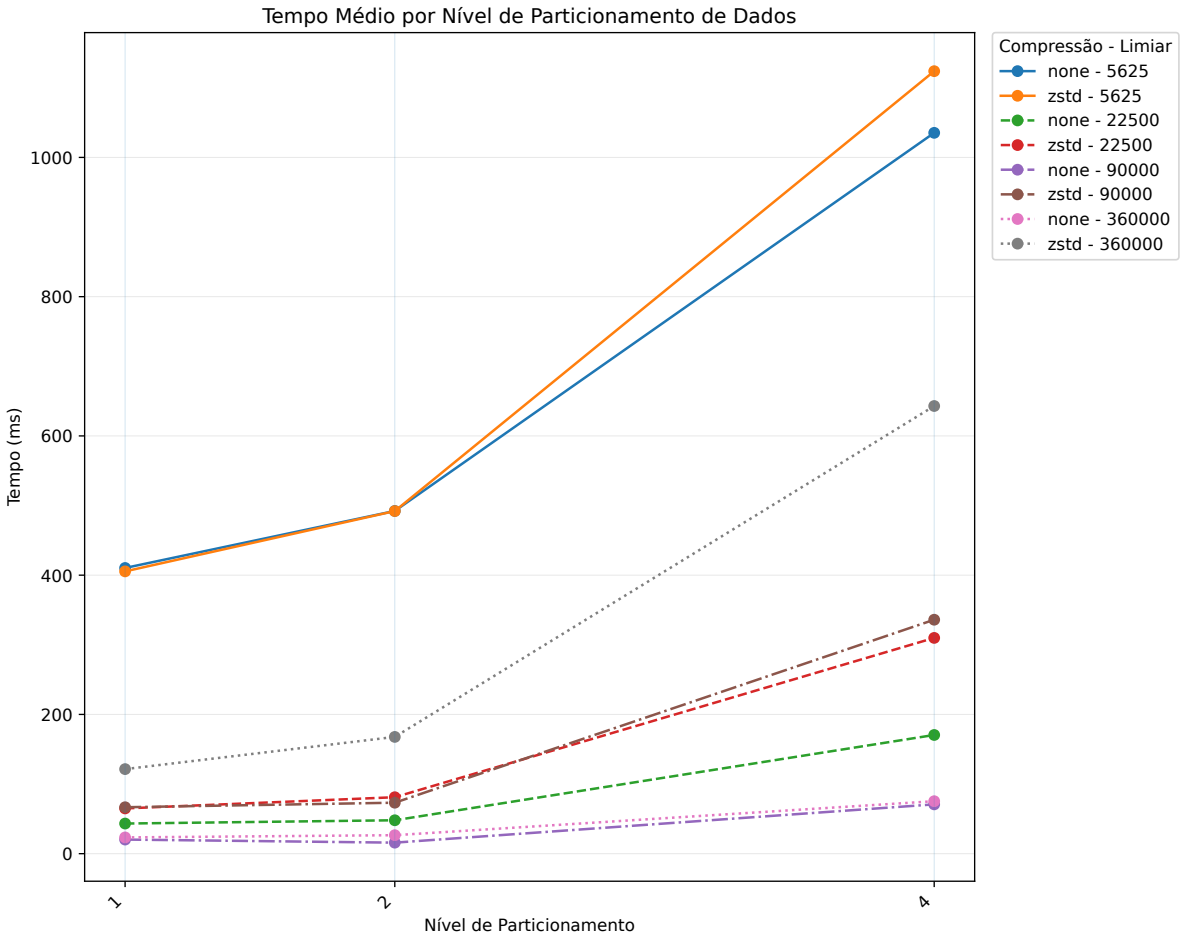

Figura 37 – Tempo médio de particionamento dos dados matriciais no Dympas para o caso sintético por nível de particionamento, tipo de compressão é limiar.

O tempo necessário para a transição entre níveis de particionamento, apresentado na Figura 37, evidencia que o limiar e a compressão são os principais determinantes no custo da operação de *split*. O maior tempo de execução registrado foi de $1123,85$ ms (limiar de 5625, com compressão `zstd`, no nível 4), enquanto o melhor desempenho foi de $15,84$ ms (limiar de 90000, sem compressão, no nível 2). Em média, o tempo de transição para os dados sem compressão foi $37,4\%$ menor quando comparado aos dados processados com o método `zstd`. Esse resultado reforça que a necessidade de descompactar e recompactar os blocos durante o particionamento aumenta o custo computacional do Dympas.

Em todos os cenários avaliados, o nível 4 de particionamento apresentou o maior *overhead*, sugerindo que o custo de particionamento aumenta à medida que a estrutura se expande temporalmente. Diferente do que foi observado no consumo de memória secundária, o tempo de particionamento se mostrou sensível ao limiar definido. No cenário com `zstd`, elevar o limiar de 90000 para 360000 causou um aumento expressivo no tempo (de $336,02$ ms para $642,96$ ms no nível 4), indicando que blocos maiores tornam o processo de subdivisão

mais custoso. Além disso, o uso de um limiar baixo (5625) também gerou aumento no tempo, devido ao aumento da quantidade de blocos gerenciados.

A análise conjunta dos resultados de uso de memória secundária e particionamento (*split*) revela um equilíbrio necessário entre o consumo de memória secundária, o tempo de inserção e o esforço de particionamento. O uso da compactação `zstd` mostrou-se eficiente ao reduzir o espaço ocupado em aproximadamente 40%, mas introduziu um custo alto no tempo de inserção, que chegou a ser seis vezes superior ao formato sem compressão. O limiar de particionamento aparece como o parâmetro que dita tanto a eficácia da compactação quanto a velocidade das operações de escrita e particionamento. Assim, a escolha das configurações depende diretamente das prioridades de uso, priorizar a economia de armazenamento ou a rapidez no processamento inicial dos dados.

Essa forte relação entre os três fatores analisados mostra que o desempenho ideal do Dympas depende de uma escolha estratégica. Enquanto o aumento no volume de dados torna a compressão mais atraente para economizar espaço, o uso de limiares muito altos ou muito baixos pode tornar as operações de *split* mais lentas, sobretudo quando combinadas com o uso de `zstd`. Portanto, a otimização deve considerar o ajuste dessas estratégias conforme as características de cada conjunto de dados e quantidade de aplicação de *split*. Manter o equilíbrio entre o uso de memória secundária, o tempo de resposta e a capacidade de crescimento do sistema para que o método proposto seja eficiente em diferentes cenários de uso.

4.4.1.2 Análise de Desempenho da Recuperação Espaço-Temporal

Esta seção descreve as análises de consultas espaço-temporais, decompondo os resultados em diferentes cenários experimentais. São avaliadas consultas agrupadas, janela deslizante, janela aleatória e instante temporal em razão do nível de particionamento. Os resultados detalhados encontram-se no Apêndice (Seção A.2).

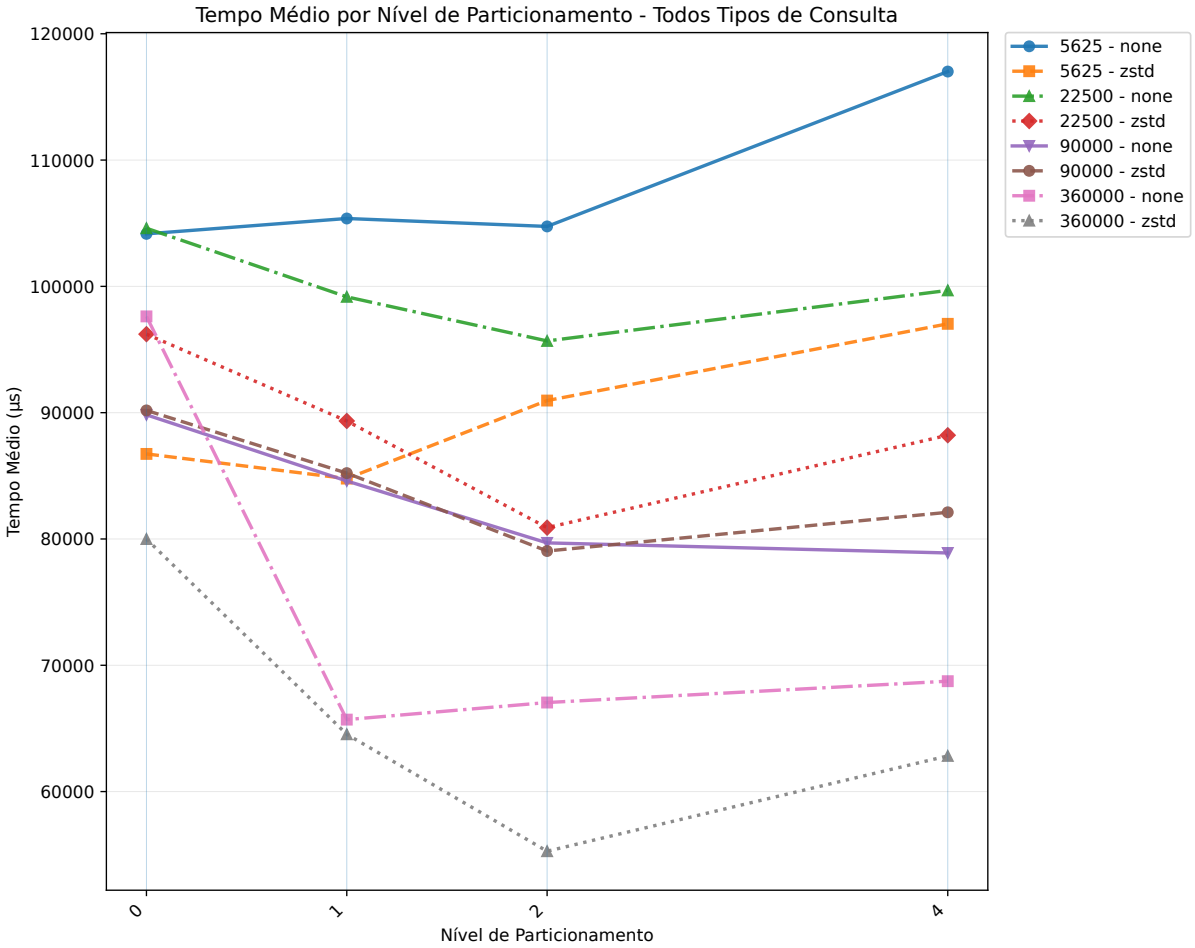

Figura 38 – Tempo médio das consultas (janela aleatória, janela deslizante e instante temporal) dividido por nível de particionamento, limiar e tipo de compressão.

A Figura 38 exibe o resultado consolidados das consultas unificadas, uma média de todos os tipos de consultas realizadas (janela deslizante, aleatória e instante temporal). O tempo de resposta é avaliado como um indicador de desempenho, com valores menores indicando melhor desempenho. O melhor desempenho foi observado no nível 2 de particionamento, utilizando o limiar de 360000 e compressão `zstd`, atingindo o tempo de $55,28$ ms. Em contrapartida, o pior tempo registrado foi de $117,01$ ms no nível 4, com limiar de 5625 e sem o uso de compressão. Entre esses dois extremos, obteve-se uma redução de tempo global de $52,76\%$. Os dados detalhados que serviram de base para esta análise podem ser observados no Apêndice A (Tabela 6).

O impacto do particionamento mostra que o nível 2 apresentou os resultados mais consistentes para os maiores limiares. Comparando o melhor caso do nível 0 ($79,99$ ms) com o melhor caso do nível 2 ($55,28$ ms), observa-se uma redução de $30,89\%$. O limiar de 360000 demonstrou ser o mais eficiente em quase todos os níveis, superando significativamente o limiar de 5625. No nível 2, a escolha do limiar 360000 em vez de 5625 gerou uma redução

de 47, 22% no tempo de consulta. A compressão `zstd` mostrou-se benéfica em cenários de volumes maiores, proporcionando uma redução adicional de 17, 5% no nível 2 com limiar 360000 (de 67, 05 ms para 55, 28 ms). Conclui-se que o aumento do limiar para 360000, aliado ao particionamento de nível 2, otimiza a recuperação ao equilibrar a carga de leitura e o processamento.

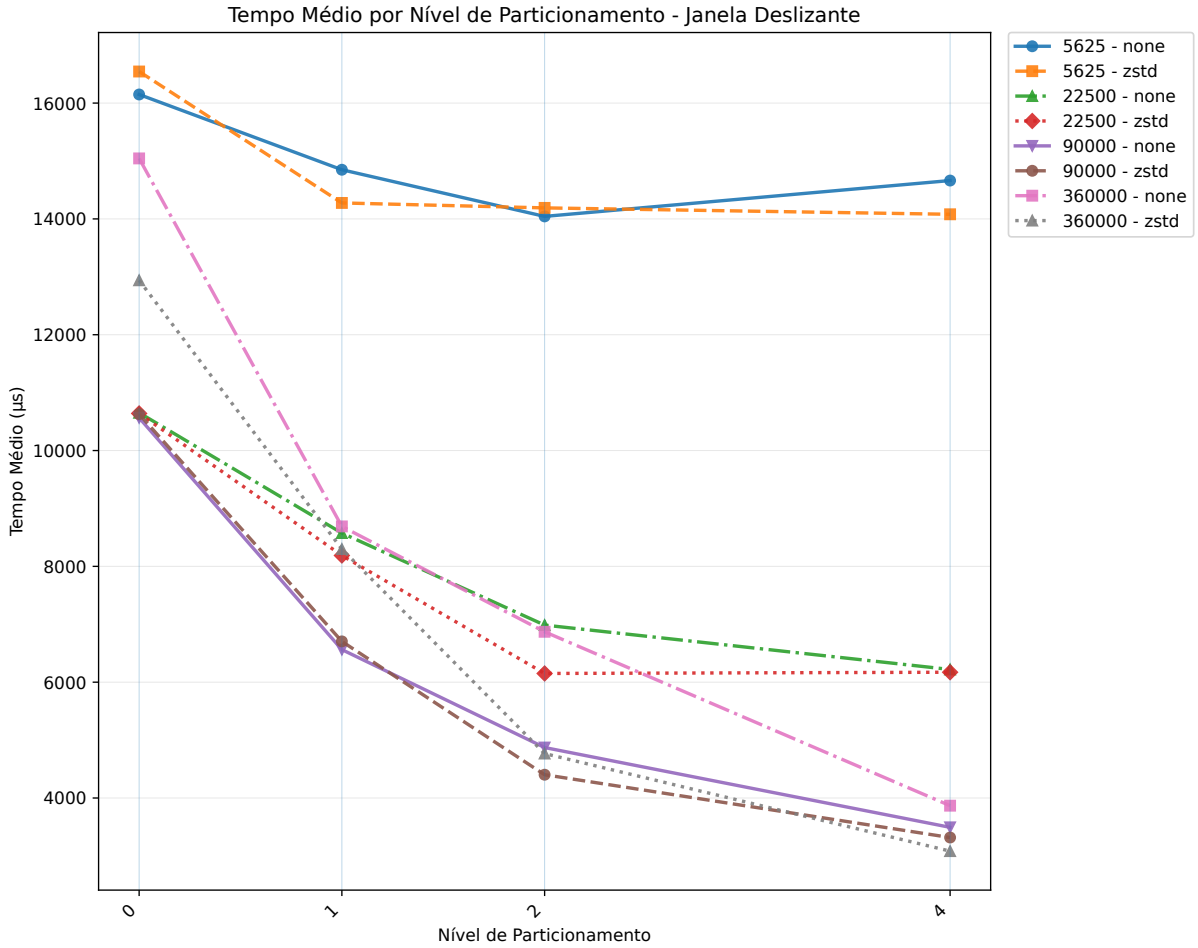

Figura 39 – Tempo médio das consultas de janela deslizante (JD) dividido por nível de particionamento, limiar e tipo de compressão.

A Figura 39 exibe os resultados das consultas de janela deslizante (JD). A análise do tempo é realizada em relação o nível de particionamento para diferentes combinações de limiares e tipo de compressão. Portanto, tempos menores indicam melhor desempenho. O melhor desempenho é observado no nível 4, com limiar de 360000 e compressão `zstd`, atingindo o expressivo tempo de 3, 08 ms. O pior tempo foi de 16, 54 ms para o nível 0, com limiar de 5625 e compressão `zstd`. Assim, é alcançada uma redução de tempo global de 81,38%. Os dados detalhados constam na Tabela 12.

O impacto do particionamento é evidenciado para a JD, onde a transição do nível 0 para o nível 4 (ambos no melhor limiar e compressão) reduziu o tempo de 10, 63 ms para

3, 08 ms, uma redução de 71, 02%. O limiar 360000 consolidou-se como a melhor escolha, permitindo uma redução de 78, 1% no nível 4 em comparação ao limiar 5625 no mesmo nível. A compactação `zstd` no nível 4 com limiar 360000 garantiu um ganho de 20, 3% em relação ao modo não compactado (3, 87 ms para 3, 08 ms). Isso indica que, para consultas de janela deslizante, o particionamento agressivo (nível 4) combinado a limiares altos e compressão eficiente minimiza o *overhead* de I/O.

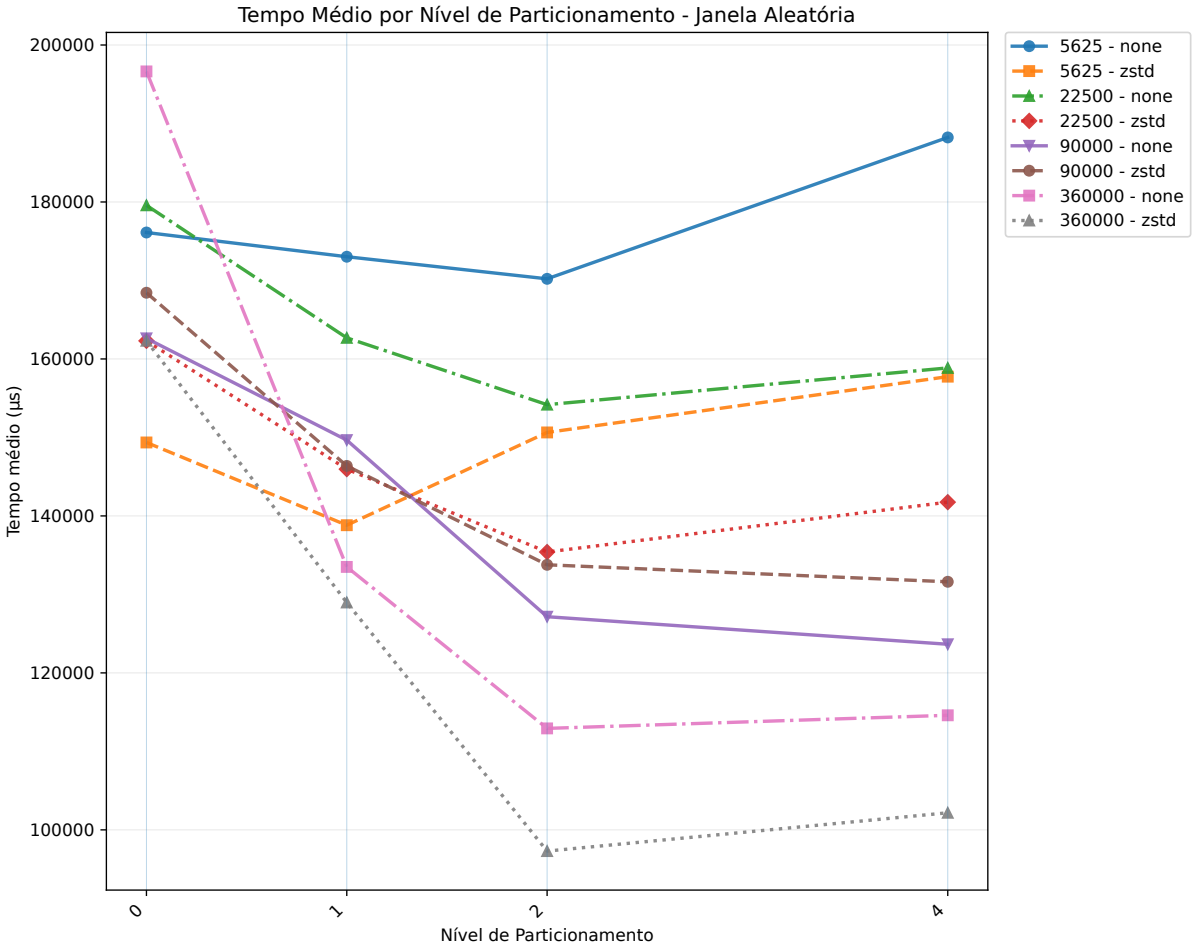

Figura 40 – Tempo médio das consultas de janelas aleatórias dividido por nível de particionamento, limiar e tipo de compressão.

A Figura 40 apresenta os resultados para janelas aleatórias (JA). A análise do tempo é realizada em relação o nível de particionamento para diferentes combinações de limiares e tipo de compressão. Cada consulta representa uma janela de dados matriciais, ou seja, cada consulta pode variar a quantidade de dados conforme o período a quantidade de dados matriciais recuperados. O melhor desempenho ocorreu no nível 2, limiar 360000 com `zstd`, atingindo 97, 29 ms. O pior tempo registrado foi de 196, 63 ms no nível 0, limiar 360000 sem compressão. A redução global obtida foi de 50, 52%. Os dados detalhados estão na Tabela 19.

Diferente de outras consultas, em JA o nível 2 superou o nível 4. Comparando o melhor cenário do nível 0 (149, 36 ms no limiar 5625) com o melhor do nível 2 (97, 29 ms), a redução foi de 34, 86%. No nível 2, o impacto do limiar é notável: a mudança de 5625 para 360000 reduziu o tempo em 35, 4% sob compressão `zstd`. Neste tipo de consulta, no nível 2 com limiar 360000, o uso de compressão (`zstd`) reduziu o tempo de 112, 93 ms para 97, 29 ms (ganho de 13, 85%). Os dados sugerem que o acesso aleatório é beneficiado por um particionamento intermediário, evitando a particionamento excessivo do nível 4.

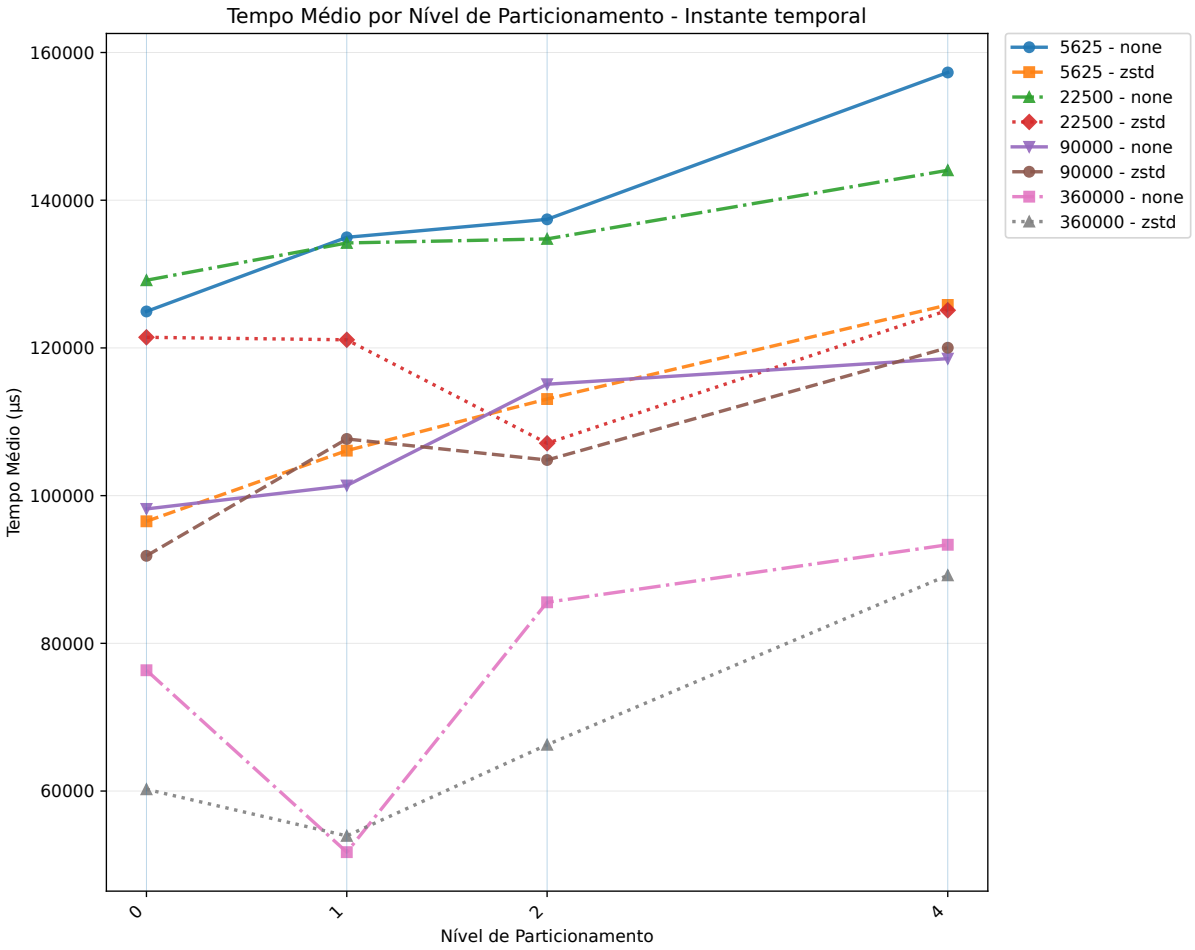

Figura 41 – Tempo médio das consultas de instante temporal dividido por nível de particionamento, limiar e tipo de compressão.

A Figura 41 detalha o desempenho para instante temporal (IT). A análise do tempo é realizada em relação o nível de particionamento para diferentes combinações de limiares e tipo de compressão. Neste caso, cada consulta também representa a recuperação de todos os dados em uma janela, que pode variar conforme o período. O melhor tempo foi de 51, 72 ms no nível 1, com limiar 360000 e sem compressão. O pior desempenho foi de 157, 30 ms no nível 4, limiar 5625, sem compressão, resultando em uma redução global de 67, 12%. Os detalhes podem ser vistos na Tabela 26.

Para consultas IT, o particionamento de nível 1 com limiares altos mostrou-se o mais eficaz. A redução ao comparar o melhor caso do nível 0 (60, 24 ms) com o nível 1(51, 72 ms) foi de 14, 14%. O impacto do limiar foi o fator mais determinante, no nível 1, aumentar o limiar de 5625 para 360000 reduziu o tempo de 134, 97 ms para 51, 72 ms (redução de 61, 68%). A compressão `zstd` apresentou resultados mistos para IT, chegando a gerar um leve overhead no nível 1 com limiar 360000 (aumentando de 51, 72 ms para 53, 93 ms). Isso sugere que, para buscas pontuais de instantes temporais, o custo de descompressão pode não compensar a redução no volume de dados recuperados quando o particionamento e o limiar já estão otimizados.

Com base nos resultados apresentados, o ajuste do limiar é o fator de maior impacto individual no desempenho, com o valor de 360000 apresentando os resultados mais eficientes em todas as modalidades de consulta. A compressão `zstd` atuou como um otimizador na maioria dos cenários, grande parte em consultas de janelas (JD e JA), onde a redução do volume de dados lidos da memória secundária compensou o custo computacional da descompressão. No entanto, o caso das consultas de instante temporal (IT) evidencia uma limitação nas consultas, o overhead da descompressão pode anular os ganhos de I/O, o que indica que a compressão deve ser aplicada de forma estratégica.

Além disso, a análise evidencia que o nível de particionamento ideal é dependente da natureza da carga de trabalho. Enquanto consultas de janela deslizante (JD) apresentaram escalabilidade quase linear com o aumento de particionamento espaço-temporal, atingindo o ápice no nível 4, as consultas aleatórias (JA) e de instante temporal (IT) encontraram seu ponto de equilíbrio em níveis intermediários (1 e 2). Isso indica que particionamento excessivo dos ROIs degrada as consultas para certos padrões de acesso.

4.4.1.3 Análise de Consultas de Janela Deslizante

Esta seção descreve as análises de consultas de janela deslizantes, decompondo os resultados em diferentes cenários experimentais. São avaliadas as taxas de recuperação relacionando a memória secundária à resposta e do desempenho temporal das consultas. Todas as métricas são analisadas em função do limiar, tamanho de janela e do tipo de ROI. Os resultados detalhados encontram-se no Apêndice (Seção A.3).

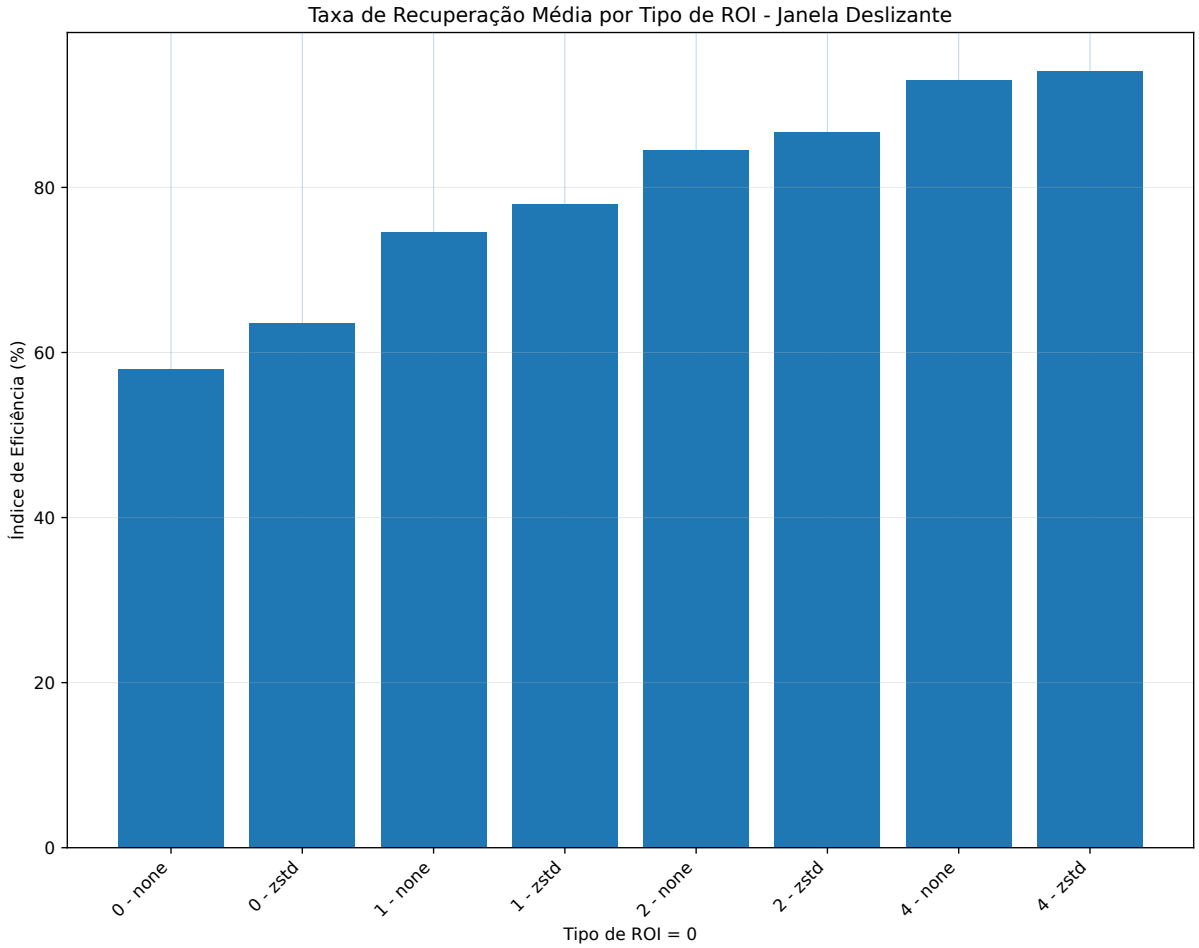

Figura 42 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tipo de ROI, nível de particionamento e tipo de compressão.

A Figura 42 ilustra a eficiência de recuperação, definida pela razão entre o volume de dados úteis e o volume total recuperado da memória secundária. A taxa de recuperação é diretamente proporcional à granularidade do particionamento. No nível 0, a taxa de recuperação para o ROI analisado variou entre $57,98\%$ e $63,57\%$, indicando um *overhead* de leitura considerável. Com a progressão para o nível 4, a eficiência convergiu para patamares superiores a 93%, atingindo $94,06\%$ sob compressão `zstd`. Os dados detalhados encontram-se na Tabela 10.

O incremento na profundidade da árvore de particionamento mitiga o impacto da dimensão do ROI na eficiência de busca. A estruturação ao nível 4 promove a equalização do desempenho, minimizando a leitura de dados adjacentes irrelevantes. Adicionalmente, o algoritmo `zstd` demonstrou manter a precisão seletiva da estrutura, apresentando ganhos marginais de eficiência em relação ao modo não compactado em todos os níveis testados. Esse resultado decorre da recuperação dos dados em formato comprimido, o que reduz

o volume de dados transferidos da memória secundária para obter a mesma informação original.

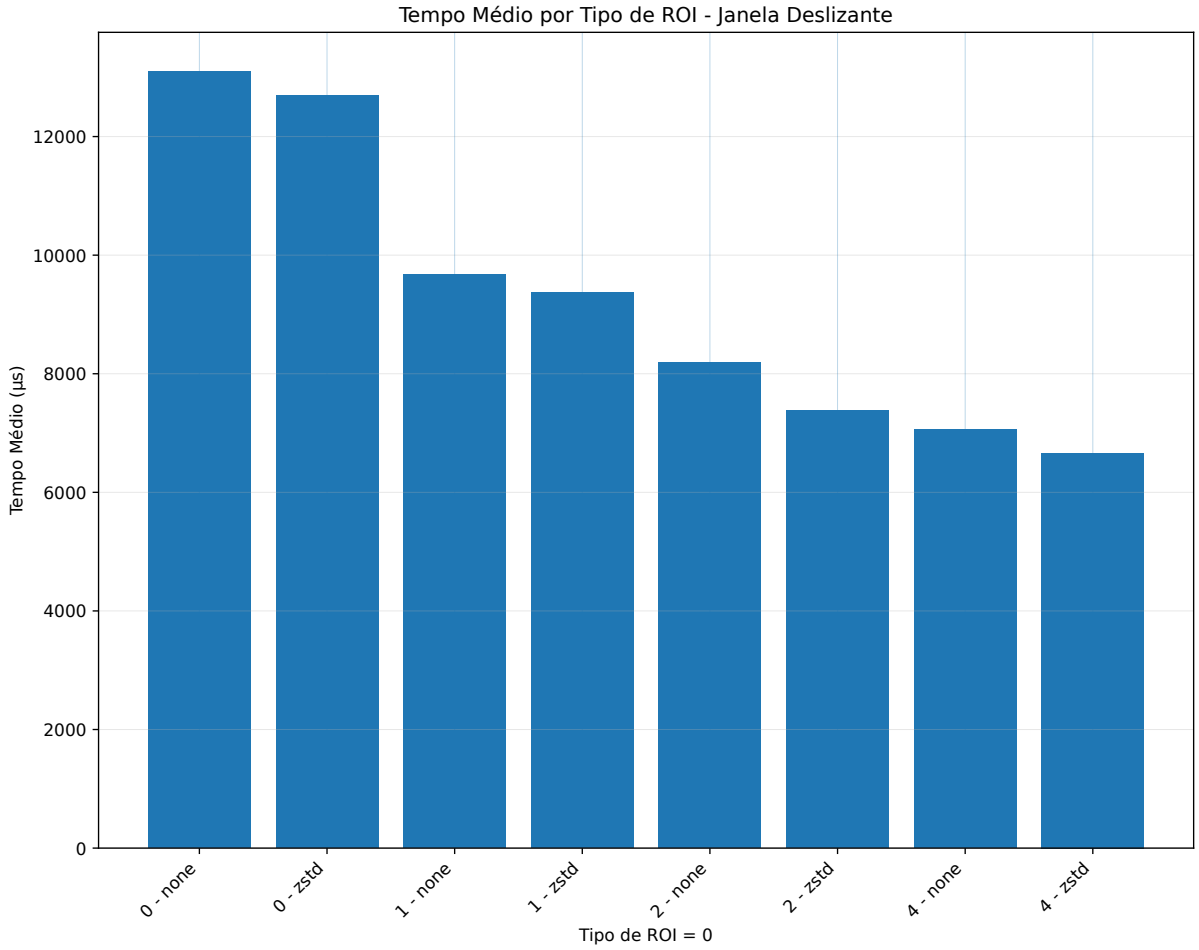

Figura 43 – Média de tempo das consultas agrupadas por tipo de ROI por nível de particionamento e tipo de compressão. Neste cenário de teste não existe categorização de tamanhos de ROI, assim, o tipo 0 representa todas as consultas. Apesar disso, a natureza do conjunto de dados da ANA mantém um padrão de tamanho de geometrias de interesse, como já foi descrito (Subseção 4.1.1).

A Figura 43 ilustra a correlação do nível de particionamento do dado matricial com o tempo de resposta das consultas de janela deslizante. A transição do nível 0 para o nível 4 resultou em uma redução de latência de $13,10$ ms para $6,66$ ms (sob compressão `zstd`), representando um ganho de performance de aproximadamente $49,1\%$. Este resultado demonstra a eficácia do particionamento na redução do volume de processamento de dados sem semântica para a consulta atrelada ao alto reaproveitamento espaço-temporal dos dados com uso do *cache*. Os valores detalhados estão disponíveis na Tabela 11.

A análise dos tempos de resposta sob compressão `zstd` demonstra que o sistema é limitado pela largura de banda de I/O. Em todos os níveis de particionamento, os dados

comprimidos apresentaram latências inferiores aos dados não comprimidos, evidenciando que a economia no tempo de transferência da memória secundária sobrepõe o custo computacional de descompressão em memória primária.

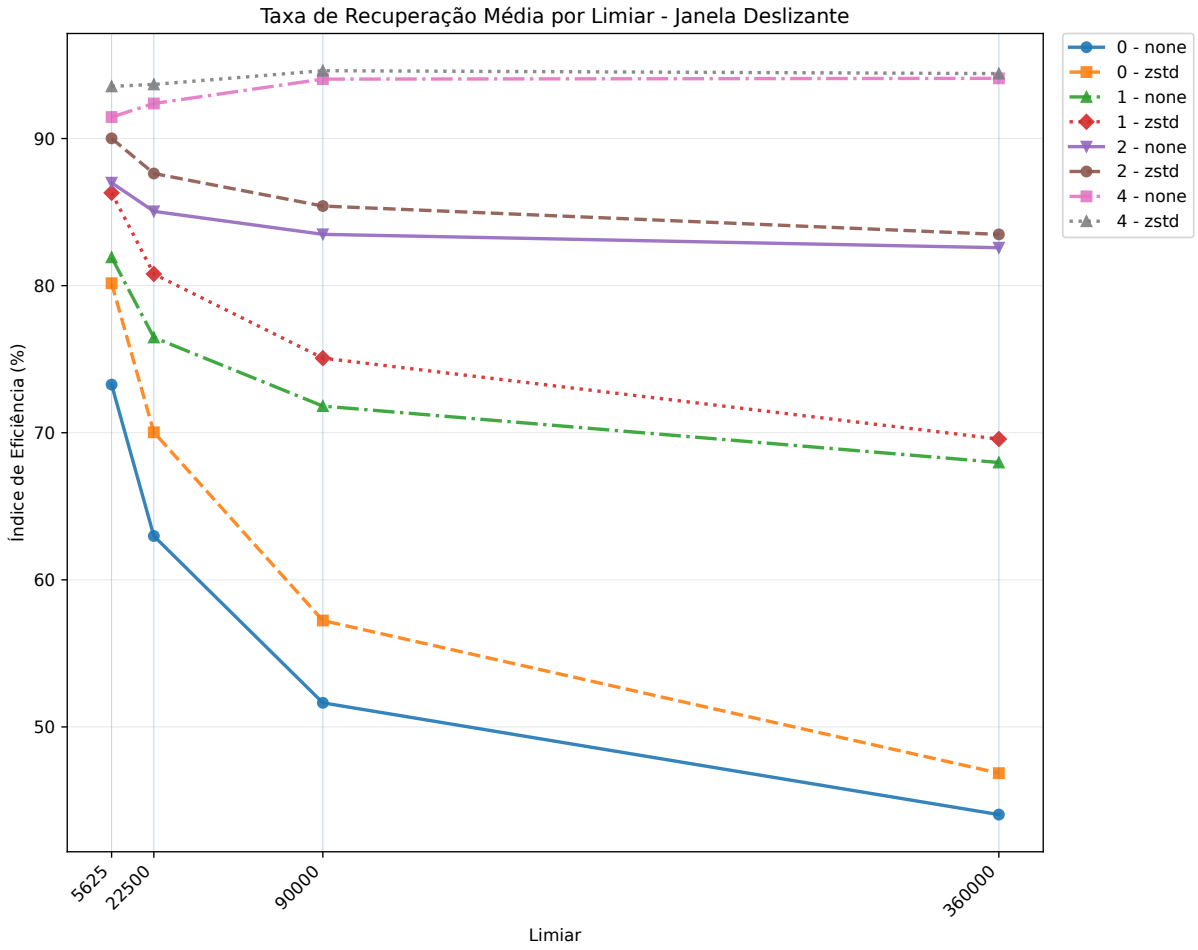

Figura 44 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por limiar, nível de particionamento e tipo de compressão.

A Figura 44 exibe a sensibilidade da taxa de recuperação em relação ao limiar de particionamento. No nível 0, pode ser observado uma disparidade significativa, o limiar 5625 atingiu uma eficiência de $73,27\%$, enquanto o limiar 360000 apresentou apenas $44,04\%$. Este padrão demonstra que limiares reduzidos favorecem uma filtragem mais precisa nos estágios iniciais da estrutura. Os dados detalhados podem ser observados na Tabela 13.

Neste contexto os resultados mostram que a convergência nos níveis superiores de particionamento. No nível 4, a eficiência de todos os limiares estabilizou acima de 93%. A convergência da taxa de aproveitamento é observada proporcionalmente ao incremento do nível de particionamento. Nesse sentido, a relevância dos dados compactados decresce conforme o aumento do nível de particionamento e do limiar. Assim, os resultados indicam

que a profundidade espaço-temporal da abordagem compensa a baixa taxa de recuperação de limiares elevados, garantindo alta performance de recuperação independente da configuração inicial de criação dos blocos.

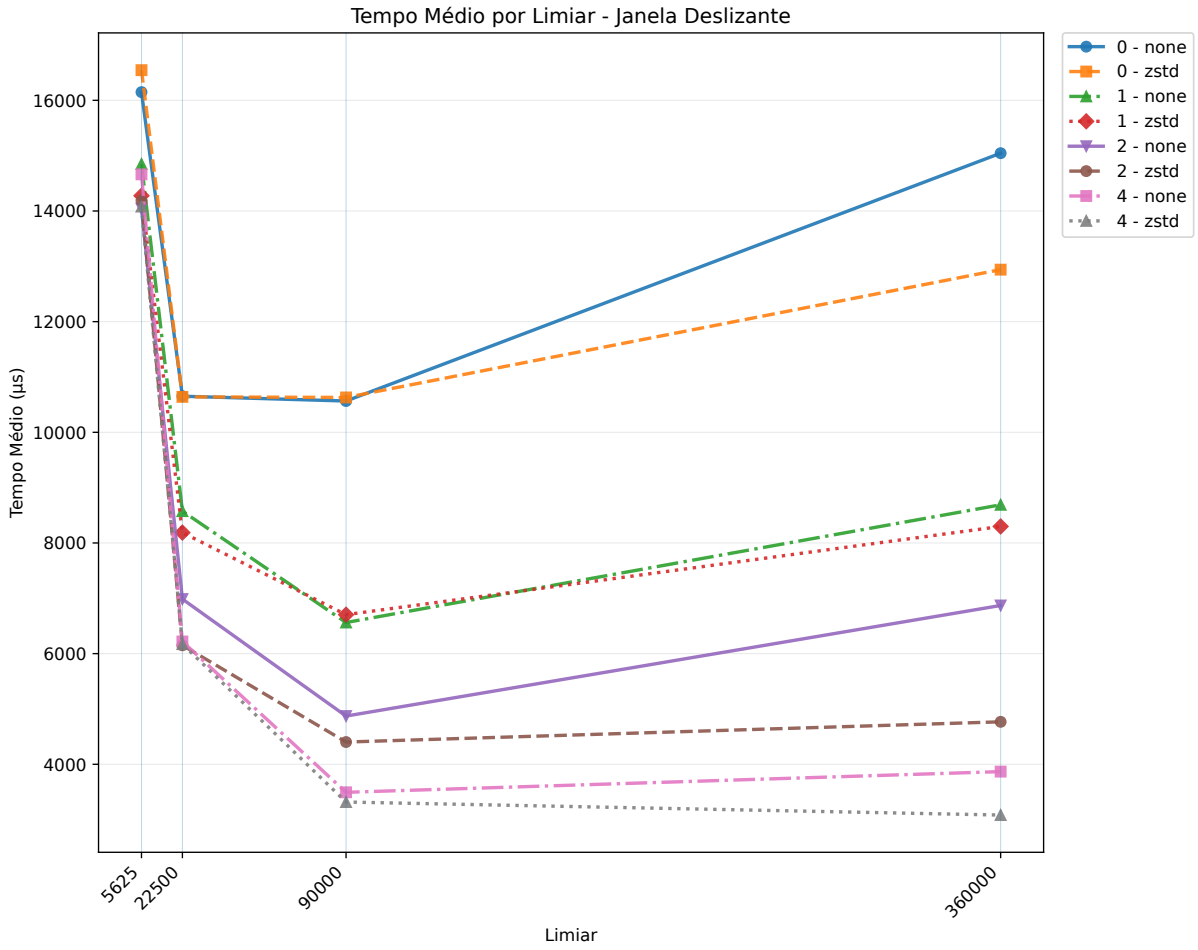

Figura 45 – Média de tempo das consultas agrupadas por limiar pelo nível de particionamento e tipo de compressão.

A Figura 45 exibe a avaliação do tempo de reposta das consultas agrupadas por limiar. A combinação de alto nível de particionamento e limiares elevados resultou nos menores tempos de resposta. O melhor desempenho foi registrado no nível 4, com limiar 360000 e compressão `zstd`, atingindo $3,08$ ms. Em comparação ao tempo de $15,04$ ms (nível 0, mesmo limiar, sem compressão), obteve-se uma redução de latência de $79,52\%$. Os dados detalhados encontram-se na Tabela 14.

Identificou-se que o limiar de 90000 e o de 360000 apresentam as melhores curvas de escalabilidade. Estes resultados evidenciam que o particionamento direcionado do Dympas apresenta desempenho superior ao particionamento exclusivamente espacial operando sob limiares reduzidos. Em contrapartida, o limiar 5625 demonstrou saturação prematura, com tempos próximos a 14 ms no nível 4, indicando que o excesso de pequenos blocos gera

um *overhead* de gerência que degrada a performance temporal, apesar da alta taxa de recuperação espacial.

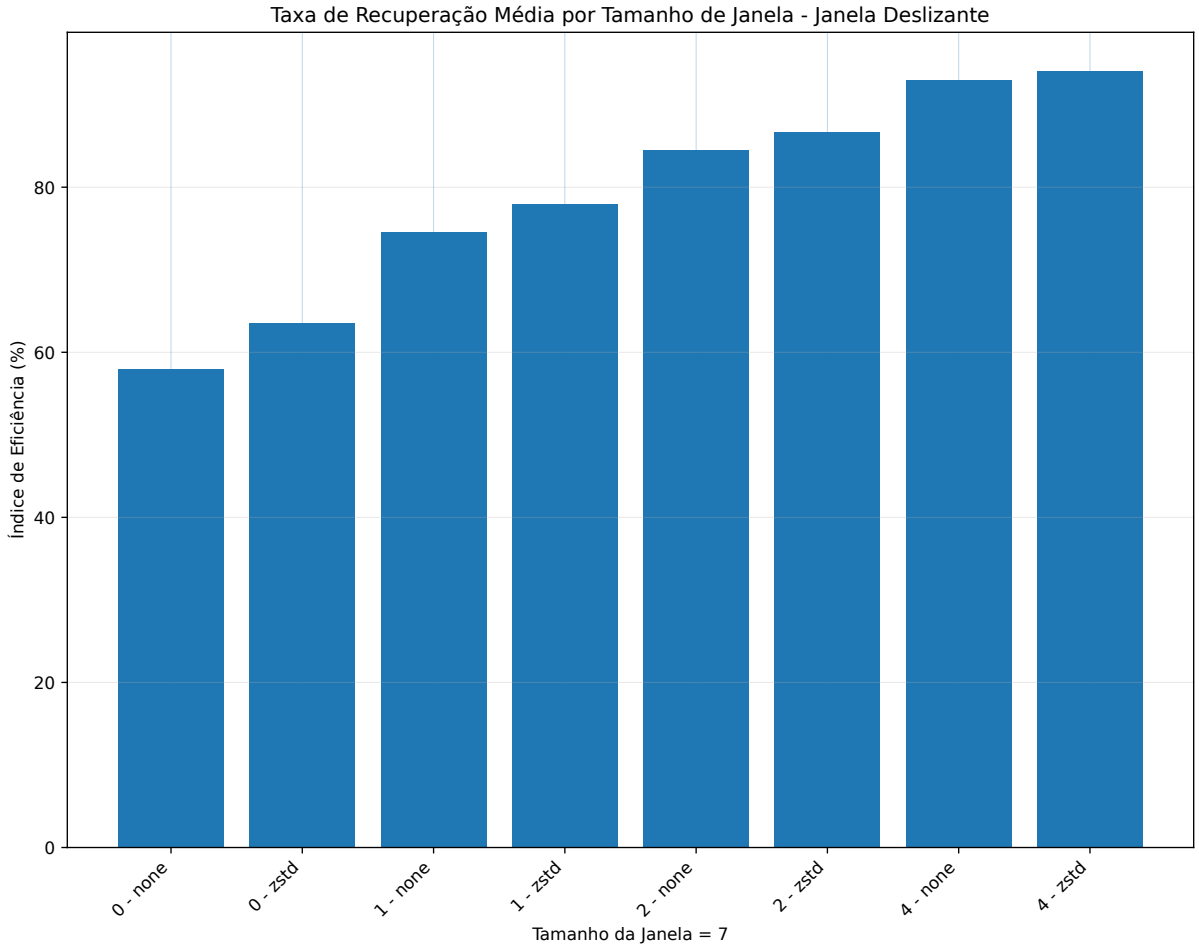

Figura 46 – Taxa média de recuperação de dados da memória secundária em relação à resposta da consulta por tamanho de janela, nível de particionamento e tipo de compressão. Tamanho da janela de 7 meses, mas a quantidade de dados matriciais varia conforme o período da janela.

A Figura 46 mostra os detalhes da eficiência de recuperação em memória secundária pelo tamanho da resposta em função da extensão da janela temporal. O cenário analisado tem janela com períodos de 7 meses, mas variando a quantidade de dados matriciais. A taxa de recuperação inicial no nível 0 foi de 57,98% sem compressão, foi registrado 63,57% com o uso de `zstd`. A progressão para o nível 4 de particionamento demonstrou alta eficácia na redução de leituras redundantes, com a taxa de recuperação atingindo 94,06%. Essa alta taxa de uso de dados, além da melhora da taxa de recuperação, é devido à natureza das consultas que permitem o uso frequente do *cache*. Os dados detalhados podem ser observados na Tabela 15.

Os resultados demonstram que a estrutura de particionamento espaço-temporal

ligado ao reaproveitamento de blocos em memória primária é o fator determinante para melhores aproveitamentos dos dados. Enquanto em níveis baixos de particionamento a eficiência é limitada, o refinamento estrutural permite uma recuperação de dados altamente seletiva, minimizando o tráfego de I/O independentemente da configuração de compressão.

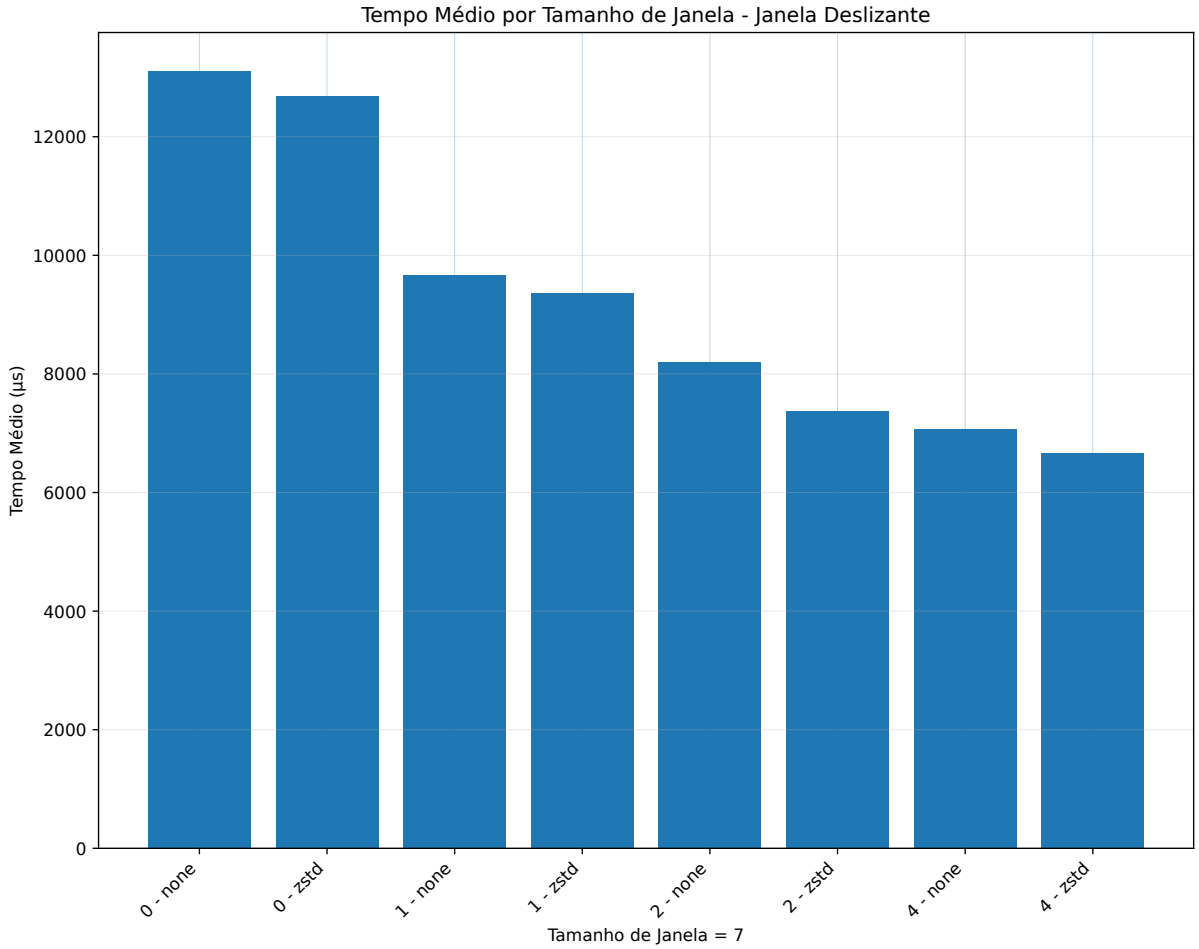

Figura 47 – Média de tempo das consultas agrupadas por tamanho de janela pelo nível de particionamento e tipo de compressão.

A Figura 47 mostra a avaliação do tempo de consulta em relação à extensão da janela no desempenho das consultas. O incremento do nível de particionamento (nível 0 para nível 4) promoveu uma redução da latência média de 13,10 ms para 6,66 ms sob compressão `zstd`, totalizando um ganho de performance global de 49,15%. Os dados detalhados podem ser observados na Tabela 16.

A performance sob compressão `zstd` destacou-se em todos os níveis de particionamento, apresentando tempos de resposta inferiores aos dados não comprimidos. Estes resultados induzem que a operação de consulta é limitada pela velocidade de leitura física dos dados, onde a redução do volume de dados transferidos compensa integralmente o esforço computacional despendido na descompressão em tempo real. Entretanto,

o incremento de desempenho maior relevância ocorre da aplicação do particionamento, pode ser observado um ganho mais expressivo na transição do nível 0 para o nível 1, independentemente do método de compactação empregado.

Em síntese, a análise dos testes realizados sobre o regime de janela deslizante revela que o particionamento espaço-temporal é o principal meio de otimização do sistema, elevando a taxa de recuperação para patamares próximos a 94% no nível 4. Embora o alto nível de particionamento melhore a latência e a eficiência recuperação de dados em memória secundária, ela introduz um ponto de inflexão em que o particionamento já não pode mais ser realizado e o *overhead* de gestão estrutural aumenta.

Os resultados consolidam a compressão `zstd` como uma estratégia eficiente para o Dympas. A diminuição do tempo de resposta nas configurações comprimidas caracteriza o sistema como limitado por I/O, evidenciando que a minimização do volume de dados transferidos assegura a escalabilidade para volumes massivos. Esse comportamento é otimizado pelo padrão de acesso das consultas, que favorece a utilização do *cache* e reduz a quantidade absoluta de requisições à memória secundária.

4.4.1.4 Análise de Consultas de Janela Aleatória

Esta seção descreve as análises de consultas de janela aleatórias, decompondo os resultados em diferentes cenários experimentais. São avaliadas as taxas de recuperação relacionando a memória secundária à resposta e do desempenho temporal das consultas. Todas as métricas são analisadas em função do limiar, do tamanho da janela e do tipo de ROI. Os resultados detalhados encontram-se no Apêndice (Seção A.4).

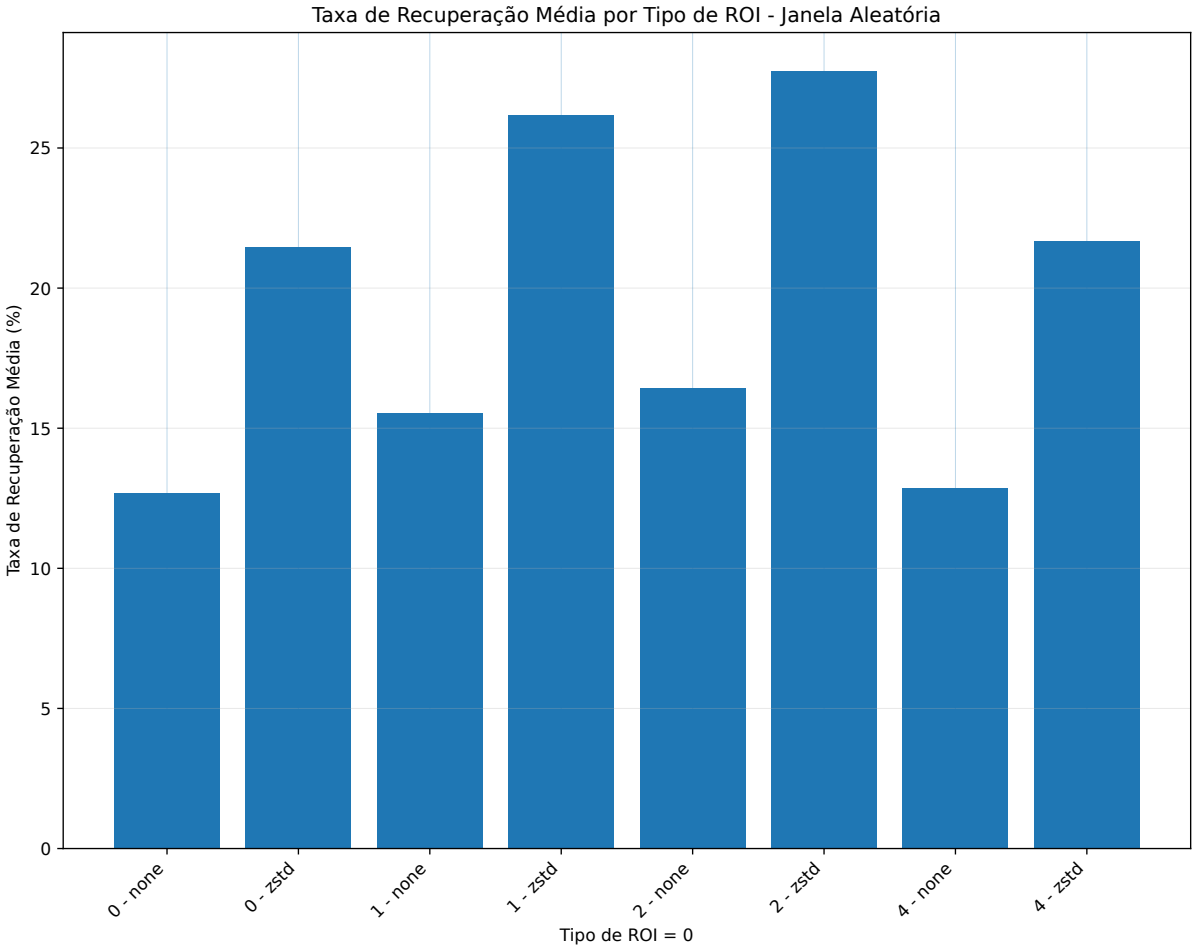

Figura 48 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tipo de ROI, nível de particionamento e tipo de compressão em janela aleatória.

A Figura 48 exibe os resultados da taxa média de recuperação de memória secundária pelo tamanho da reposta. Essa análise da taxa de recuperação em memória secundária para os ROIs revela que a eficiência de recuperação é sensível ao nível de particionamento, apresentando um comportamento de pico em granularidade intermediária (nível 2). No nível 0, a taxa média registrou $12,66\%$, elevando-se para o valor máximo de $27,73\%$ no nível 2 sob compressão `zstd`, o que representa um incremento de eficiência de $118,9\%$ em relação à base. Contudo, a transição para o nível 4 resultou em um declínio para $21,66\%$, indicando que limiares reduzidos impactaram o resultado geral, visto que o particionamento em níveis de alta granularidade não gera ganho incremental da taxa de recuperação.

O uso do `zstd` para compreensão demonstrou uma superioridade em todos os níveis, mantendo uma margem de taxa de recuperação significativamente superior ao modo `none`. Este resultado ocorre pela maior densidade de informação por bloco físico comprimido, permitindo que a recuperação de um único bloco retorne uma proporção maior de dados

úteis para a resposta. Os dados confirmam que, para janelas aleatórias, o nível 2 oferece a melhor relação entre o custo de recuperação e o volume de dados úteis processados neste contexto de testes.

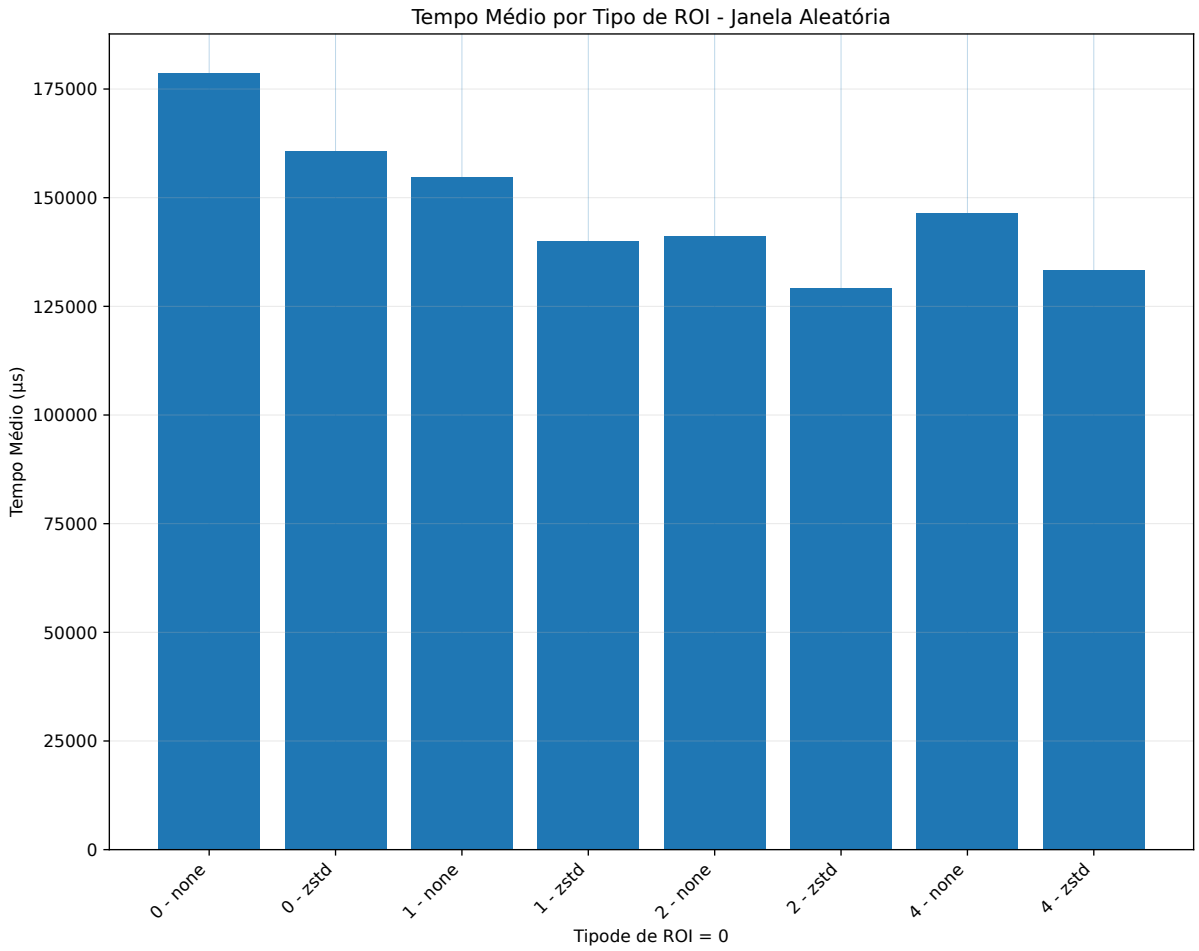

Figura 49 – Média de tempo das consultas de janela aleatória agrupadas por tipo de ROI por nível de particionamento e tipo de compressão.

A Figura 49 ilustra a avaliação do tempo médio das consultas por ROI. Os resultados mostram que o particionamento espaço-temporal é eficaz na redução do custo temporal, mas atinge um platô de performance. O tempo de resposta declinou de 178, 73 ms (nível 0, `none`) para a marca mínima de 129, 27 ms (nível 2, `zstd`), uma redução total de 27, 6%. Contudo, no nível 4, a latência elevou-se levemente para 133, 32 ms sob compressão, evidenciando que o particionamento com limiares reduzidos atinge uma escala em que o *overhead* de gerência passa a degradar o desempenho das consultas.

A superioridade temporal da configuração de dados comprimidos pelo `zstd` em todos os níveis de particionamento espaço-temporal evidencia que o tempo de descompressão é inferior ao ganho obtido na recuperação de dados da memória secundária. A economia

na transferência de dados compensa integralmente o tempo adicional introduzida pela descompressão.

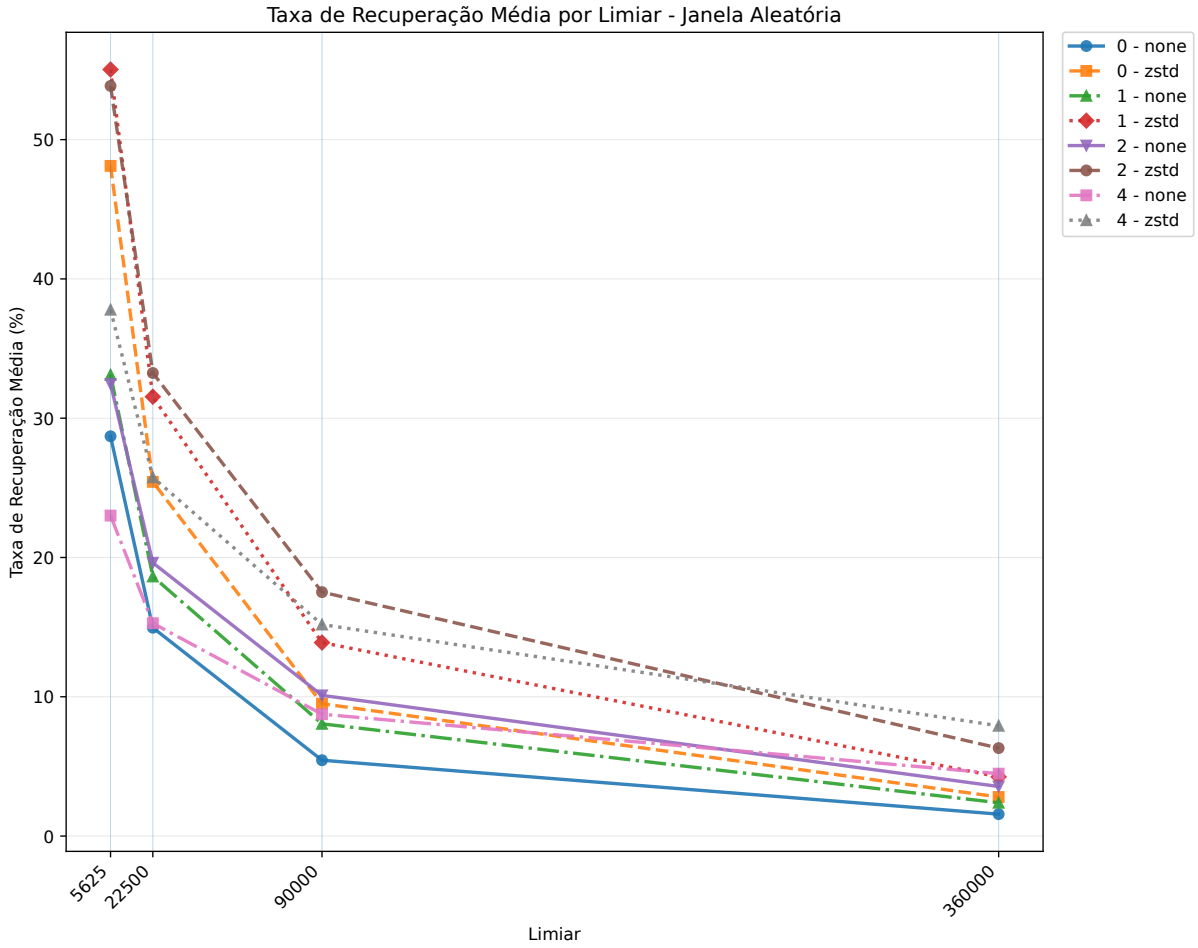

Figura 50 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta de janela aleatória por limiar, nível de particionamento e tipo de compressão.

A Figura 50 ilustra a taxa média de recuperação de dados da memória secundária em relação ao limiar para janela aleatória. Esta análise da taxa de recuperação em função do limiar expõe uma relação entre o tamanho do bloco inicial e a eficiência de recuperação. Limiares reduzidos (5625) apresentaram uma taxa de recuperação de até $55,03\%$ no nível 1 com `zstd`, enquanto o limiar mais elevado (360000) atingiu apenas $7,91\%$ no nível 4. Ademais, a aplicação de particionamentos de forma indiscriminada não assegura melhorias da taxa de recuperação. Esta afirmação pode ser observada no limiar 5625, em que o nível 4 sem compressão apresentou o menor índice de eficiência.

Entretanto, o particionamento espaço-temporal atua como uma estratégia eficaz de otimização para limiares elevados. Para o limiar 360000, a taxa de recuperação elevou-se de $2,80\%$ (nível 0) para $7,91\%$ (nível 4) sob compressão, representando um incremento de

182, 5% na eficiência da taxa de recuperação. Tais resultados indicam que, embora limiares reduzidos ofereçam maior taxa de recuperação, o refinamento em níveis espaço-temporais superiores compensa a baixa precisão de blocos com limiares mais elevados e particionamento direcionados ao contexto da busca. Os resultados com `zstd` demonstram um aproveitamento superior dos blocos, decorrente da redução do volume de transferência da memória secundária. Contudo, existe uma degradação da eficiência no menor limiar (5625), devido ao gerenciamento de mais blocos.

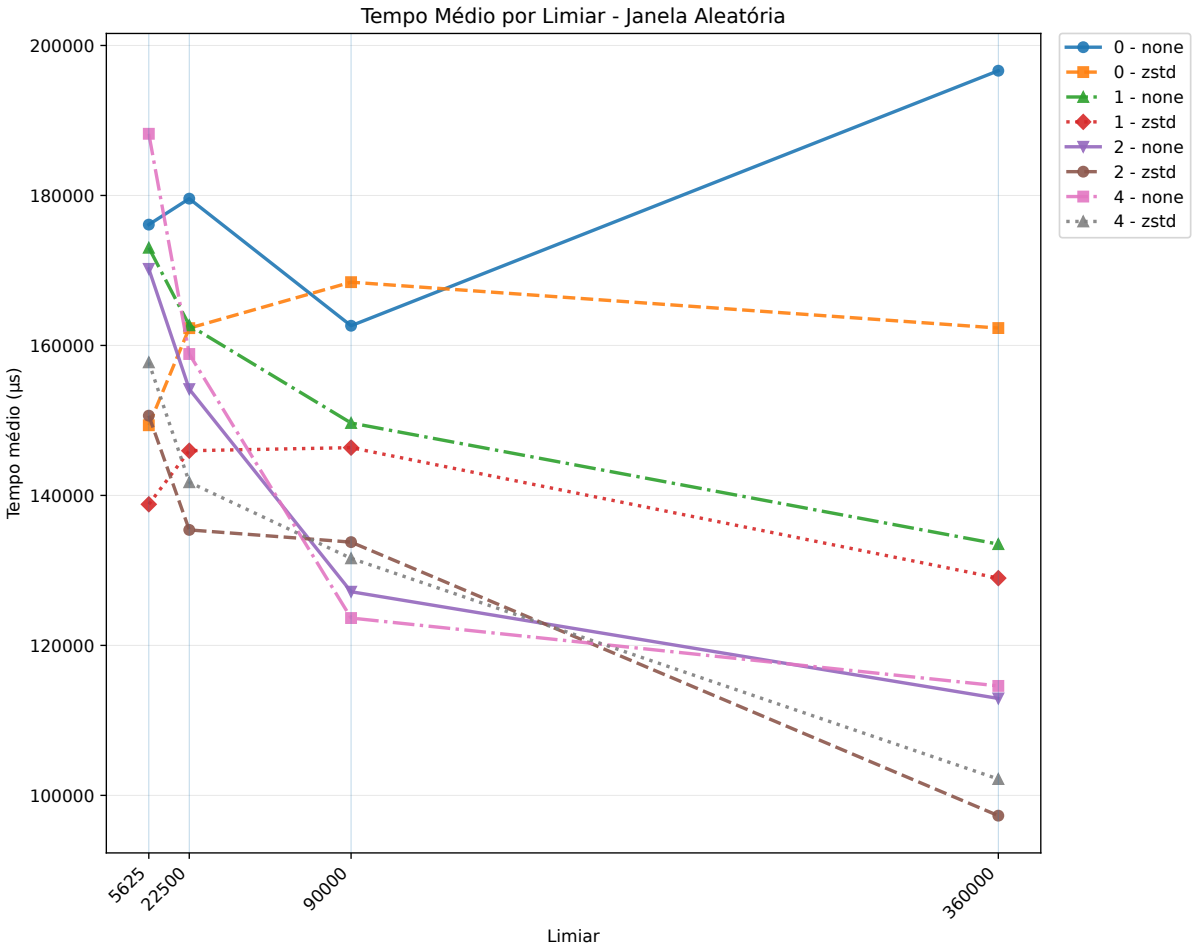

Figura 51 – Média de tempo das consultas de janela aleatória agrupadas por limiar pelo nível de particionamento e tipo de compressão.

A Figura 51 apresenta a análise do desempenho das consultas em função do limiar, destacando a configuração com limiar 360000 no nível 2 com `zstd` como o cenário de melhor desempenho, registrando 97, 29 ms. Limiares reduzidos (5625), apesar da alta taxa de recuperação, apresentaram latências superiores, atingindo o patamar de 150, 6 ms sob as mesmas condições (nível 2, `zstd`). Essa inversão de desempenho indica que o particionamento excessivo, junto a limiares baixos, introduz um custo de gerência estrutural em cenários com baixo aproveitamento de *cache*.

A transição do nível 0 para o nível 2 no limiar 360000 sob compressão promoveu uma redução de latência de 162,31 ms para 97,29 ms (ganho de 40,1% em performance). Os resultados indicam que, para consultas de janela aleatória, a estratégia mais eficiente consiste no uso de limiares elevados, o que permite um controle mais efetivo do particionamento espaço-temporal conforme o contexto dos dados e da consulta.

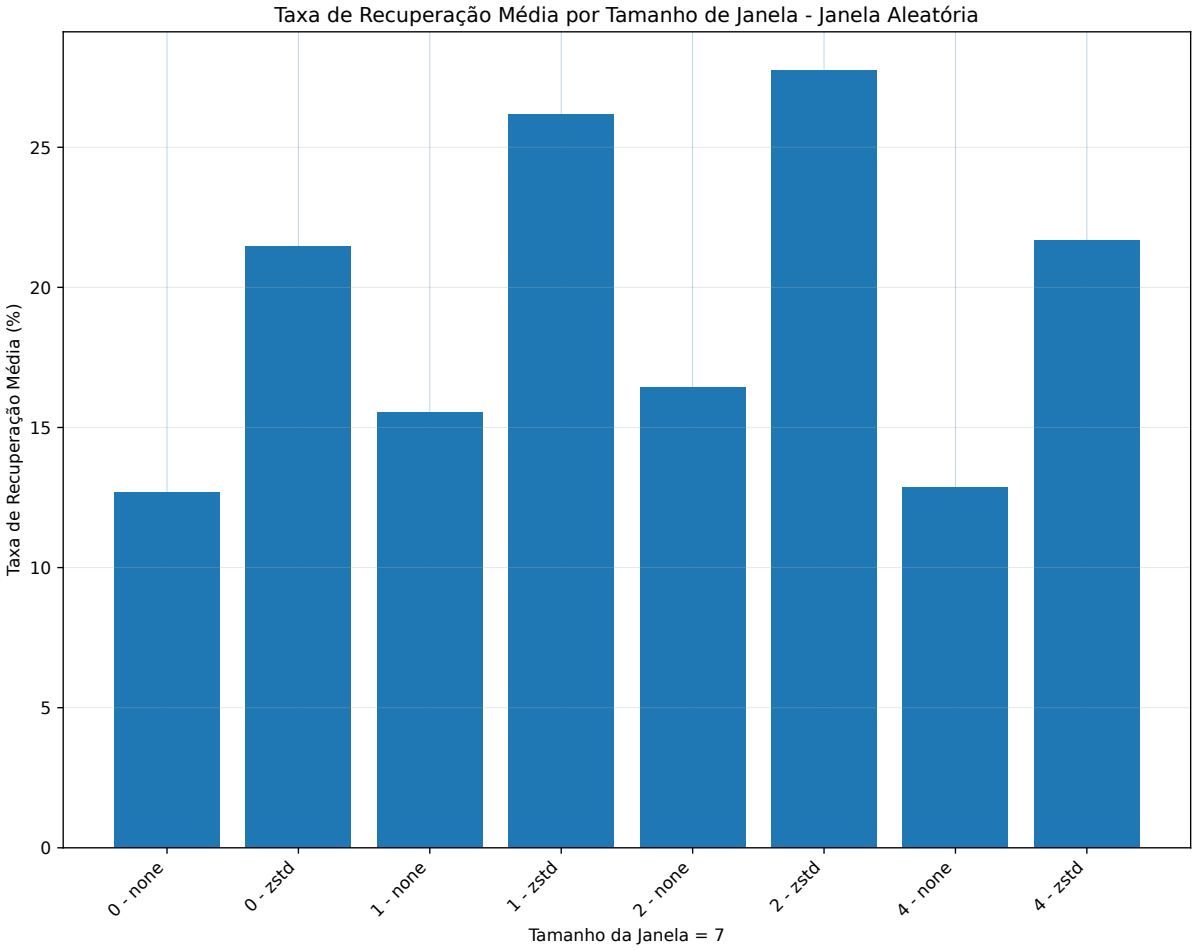

Figura 52 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta de janela aleatória por tamanho de janela, nível de particionamento e tipo de compressão.

A Figura 52 exibe a taxa de recuperação de dados da memória secundária pela resposta para janelas de 7 meses. Pode ser observado que a compressão apresentou desempenho superior sob os mesmos parâmetros de avaliação. No nível 0, a eficiência da taxa de recuperação com compressão registrou 21,46%, elevando-se para o ápice de 27,73% no nível 2. Esse comportamento mostra que o sistema atinge seu limite da taxa de recuperação (neste contexto de consultas) para janelas aleatórias em níveis intermediários de particionamento, degradando para 21,66% no nível 4, conforme observado nos resultados anteriores.

A configuração `zstd` manteve uma taxa de recuperação de cerca de 71% superior ao modo `none` no nível 4 (21, 66% e 12, 87%, respectivamente). Estes dados demonstram que, embora a estrutura espaço-temporal favoreça a eficiência, a compressão é o fator determinante para otimizar o volume de dados lidos da memória secundária. Diferentemente do regime de janelas deslizantes, que apresenta maior reúso de *cache*, o regime de janelas aleatórias não sustenta o mesmo patamar de eficiência.

O ganho mais expressivo sob compressão ocorre na transição entre o nível 0 e o nível 1 (20, 5%). Assim, neste contexto, as consultas apresentam maior reúso médio de dados em níveis superiores de particionamento. Contudo, o particionamento excessivo decorrente de níveis muito altos (nível 4, possivelmente, e níveis acima) pode degradar outros indicadores de performance, conforme evidenciado em análises anteriores.

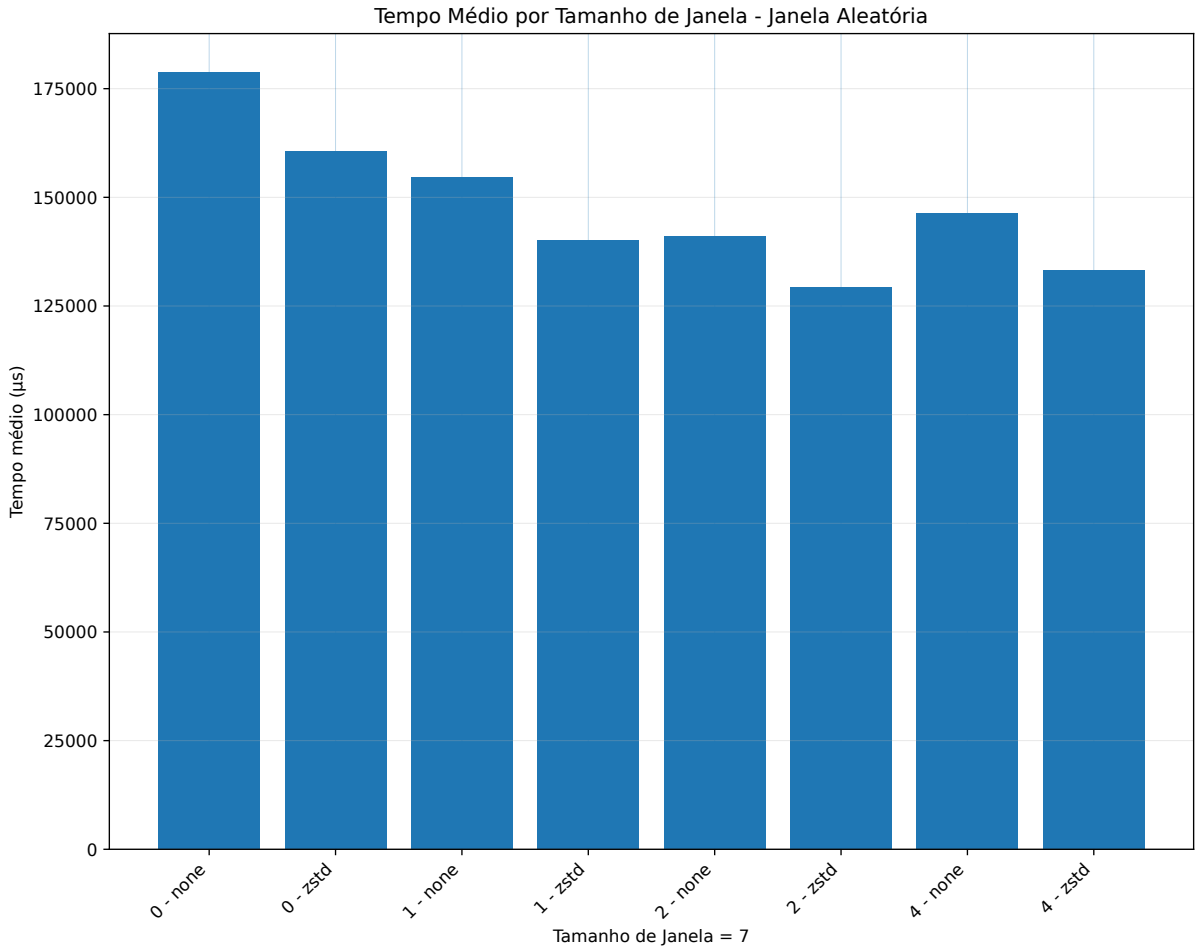

Figura 53 – Média de tempo das consultas de janela aleatória agrupadas por tamanho de janela pelo nível de particionamento e tipo de compressão.

A Figura 53 apresenta a análise do tempo de consulta para janelas aleatórias de sete meses. Os resultados revelam que a compressão `zstd` reduz a latência média em aproximadamente 10% em todos os níveis de particionamento, em comparação ao modo

não comprimido (`none`). O tempo de resposta declinou de 178, 73 ms (nível 0, `none`) para 129, 27 ms (nível 2, `zstd`). É possível notar uma sutil degradação no tempo médio de consulta no nível 4, evidenciando que o particionamento espaço-temporal ultrapassou o limite de eficiência para o tamanho da janela, impactando negativamente a latência.

O sistema é limitado pela transferência de dados da memória secundária, na qual a redução do volume via compressão é o fator relevante sob parâmetros equivalentes. Contudo, a otimização de maior relevância decorre da estrutura de particionamento espaço-temporal; níveis intermediários (nível 2) oferecem a melhor amortização dos tempos de resposta. A latência final no regime JA é, portanto, o resultado do equilíbrio entre a eficiência de transferência e o nível de particionamento adequado ao contexto.

Os resultados experimentais para o regime de janela aleatória indicam que a eficiência do sistema Dympas, neste contexto de análise, tem ponto de otimização em níveis intermediários de particionamento (nível 2). A elevação da latência para 133,32 ms no nível 4 (Figura 49), indica que o particionamento em níveis com alto limiar introduz um *overhead* não compensado pelo ganho da taxa de recuperação. Esse resultado decorre de consultas com janela com extensão temporal de dados matriciais inferior à extensão espaço-temporal do particionamento aplicado.

No que se refere ao desempenho temporal, a configuração mais eficiente utilizou o limiar de 360000 associado ao nível 2 e compressão `zstd`, registrando a latência mínima de 97, 29 ms (Figura 51). Este cenário representa um ganho de performance de 40, 1% em relação ao nível 0 devido ao particionamento direcionado ao contexto da consulta. Em contrapartida, limiares reduzidos (5625) apresentaram latências superiores, chegando a 150, 6 ms sob as mesmas condições. Os dados indicam que, para consultas de janela aleatórias, limiares maiores reduzem a recuperação de dados sem semântica para consulta. Enquanto o particionamento em níveis moderados entrega a otimização ideal para o contexto de busca desta análise.

4.4.1.5 Análise de Consultas de Instante Temporal

Esta seção descreve as análises de consultas de instante temporal, decompondo os resultados em diferentes cenários experimentais. São avaliadas as taxas de recuperação relacionando a memória secundária à resposta e do desempenho temporal das consultas. Todas as métricas são analisadas em função do limiar e do tipo de ROI. Os resultados detalhados encontram-se no Apêndice (Seção A.5).

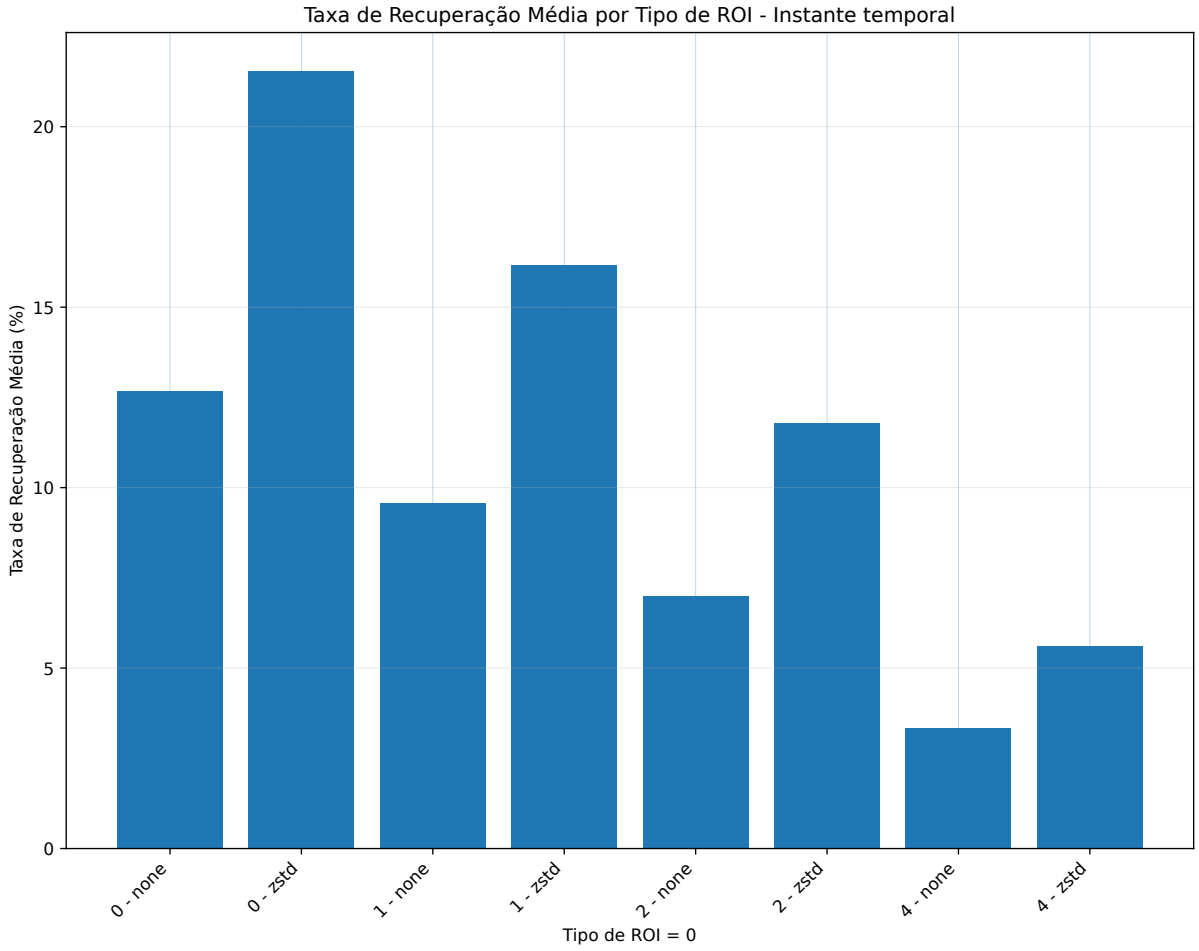

Figura 54 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tipo de ROI, nível de particionamento e tipo de compressão em instante temporal.

A Figura 54 apresenta a razão entre o volume recuperado da memória secundária e o tamanho da resposta para ROIs de consultas de instante temporal. Os resultados revelam uma correlação inversa entre o nível de particionamento e a eficiência de recuperação. No nível 0, a taxa de aproveitamento registrou $12,67\%$ no modo `none` e $21,53\%$ com `zstd`, enquanto a progressão para o nível 4 resultou em uma redução para $3,35\%$ e $5,60\%$, respectivamente.

A compressão `zstd` atenuou a perda de eficiência, mantendo uma densidade de informação aproximadamente 67% superior ao modo `none` no nível 4 ($5,60\%$ contra $3,35\%$). Contudo, a tendência indica que o aumento do nível de particionamento reduz a eficiência em consultas de instante temporal. O decréscimo na taxa de recuperação, correlacionado ao acréscimo de níveis espaço-temporais, pode decorrer de ROIs adjacentes, que ao serem particionados influenciam o particionamento desses ROIs adjacentes neste contexto de consulta.

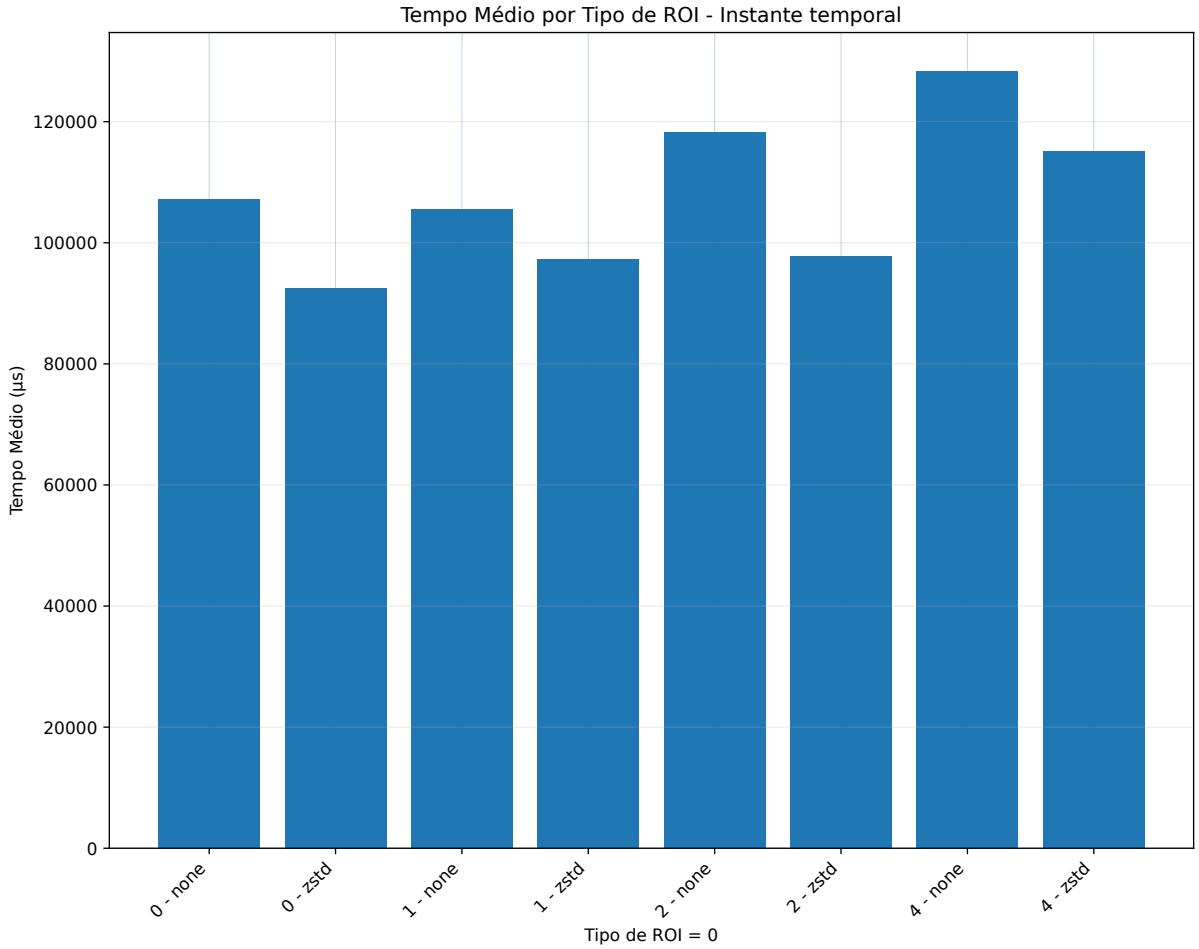

Figura 55 – Média de tempo das consultas de instante temporal agrupadas por tipo de ROI por nível de particionamento e tipo de compressão.

A Figura 55 exibe a análise do tempo médio de consulta para ROIs de instante temporal. Os resultados indicam que a latência média é influenciada pelo particionamento espaço-temporal. Os dados não comprimidos tem o tempo de resposta elevado de $107,16$ ms no nível 0 para $128,31$ ms no nível 4 ($19,7\%$). Sob compressão `zstd`, a latência elevou-se de $92,51$ ms para $115,03$ ms. Estes resultados mostram que no cenário de menor aproveitamento do sistema (consultas de instante temporal sob particionamento espaço-temporal) o acréscimo de tempo é baixo quando comparado aos ganhos observados nos regimes de janela aleatória e deslizante.

Apesar da tendência de elevação da latência, observa-se que a compressão `zstd` conteve a degradação temporal entre 10% e 15% em relação aos dados originais. No entanto, para este contexto específico, o ponto de operação ideal permanece no nível 0, configuração na qual o particionamento exclusivamente espacial minimiza o tempo de consultas.

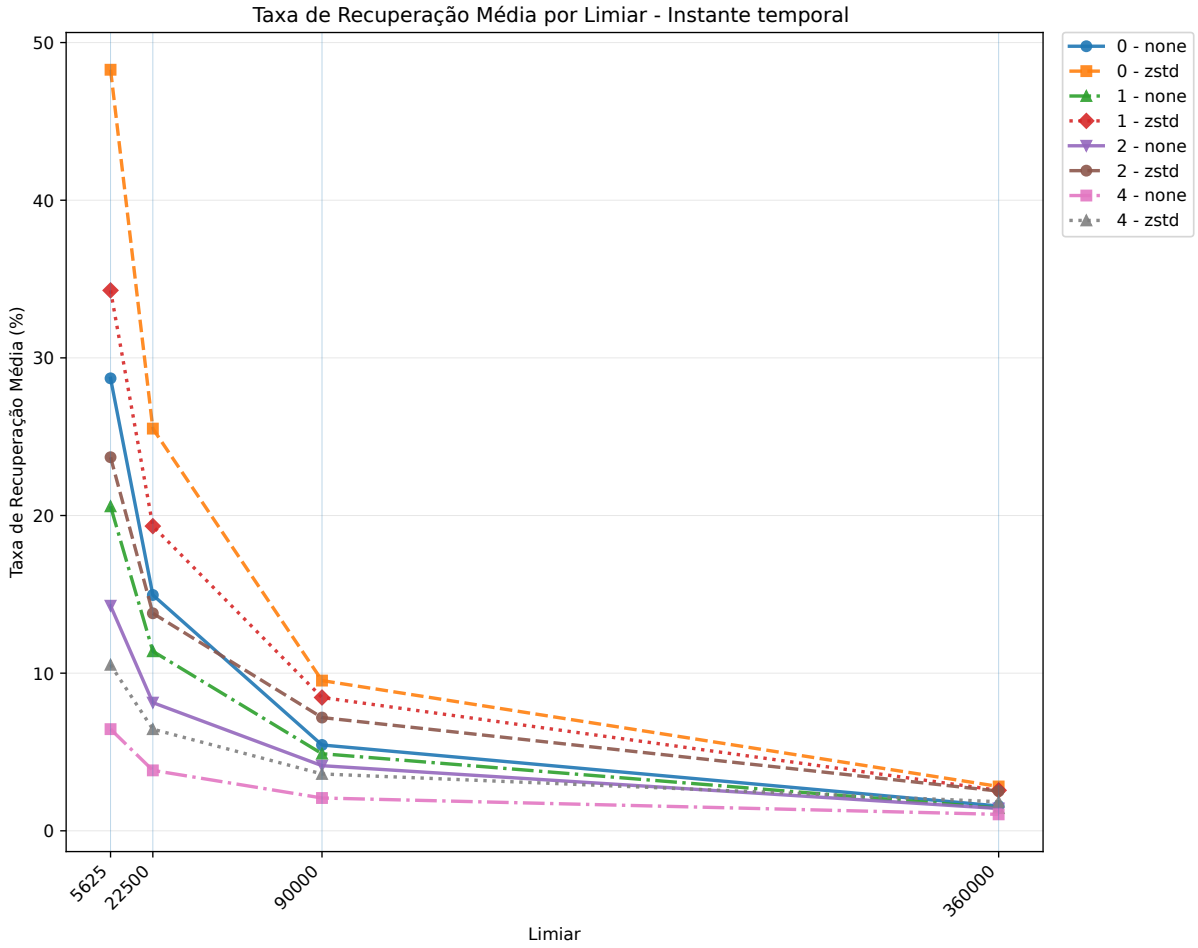

Figura 56 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta de instante temporal por limiar, nível de particionamento e tipo de compressão.

A Figura 56 exibe a razão média entre o volume de dados recuperado da memória secundária e o tamanho da resposta para consultas de instante temporal em função do limiar. A análise indica que a eficiência de recuperação é influenciada pela compressão de dados em todos os cenários sob parâmetros equivalentes. No nível 0, o limiar de 5625 sem compressão apresentou uma taxa de recuperação de $28,70\%$, enquanto a configuração com `zstd` atingiu $48,28\%$. Contudo, este limiar apresentou uma redução acentuada com o incremento do particionamento, declinando para $6,45\%$ (sem compressão) e $10,53\%$ (com compressão) no nível 4, o que representa perdas de $77,5\%$ e $78,2\%$, respectivamente.

No limiar de 360000, a redução de eficiência foi proporcionalmente menor, variando de $1,57\%$ para $1,04\%$. Os resultados mostram que, para este contexto de consultas, a combinação de baixa profundidade de particionamento com limiares reduzidos (5625) é a configuração que minimiza o tráfego de dados irrelevantes à resposta.

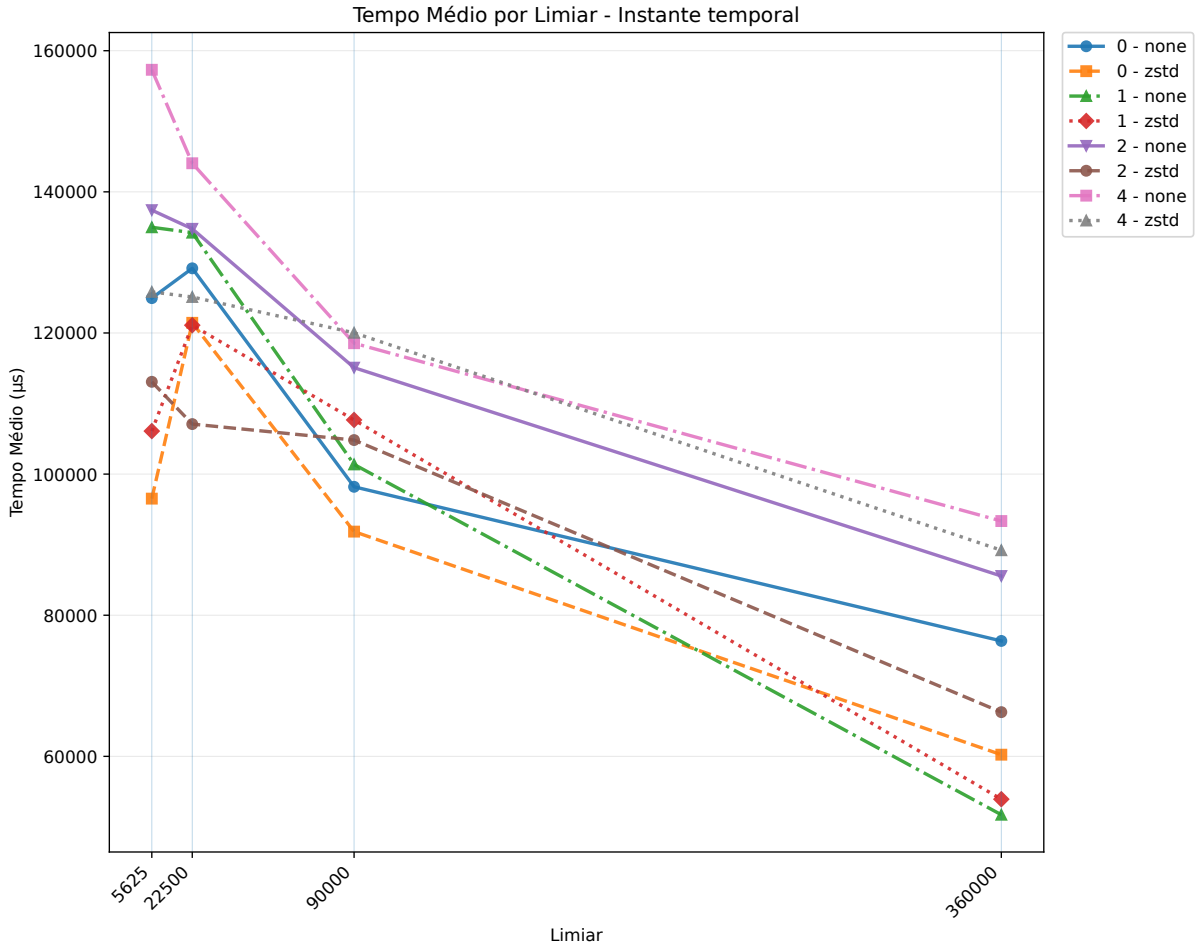

Figura 57 – Média de tempo das consultas de instante temporal agrupadas por limiar pelo nível de particionamento e tipo de compressão.

A Figura 57 exibe a análise do tempo médio de consulta para instantes temporais em função do limiar. Os resultados identificam a configuração com limiar 360000 no nível 1 como a de maior eficiência, tanto para dados sem compressão (51, 72 ms) quanto para dados comprimidos com `zstd` (53, 92 ms). Esse desempenho contrasta com o limiar de 5625, em que os tempos de consultas ultrapassam 124 ms em todos os níveis de particionamento. A vantagem de limiares maiores reside na redução do *overhead* no gerenciamento de blocos com múltiplos instantes temporais para o processamento de uma consulta de apenas um instante.

Pode se observar que o particionamento de nível 4 eleva a latência em todos os limiares analisados. Para o limiar 360000, o tempo de consulta aumentou de 51, 72 ms (nível 1) para 93, 34 ms (nível 4). Estes resultados indicam que, para consultas de instante temporal, a estratégia mais eficaz consiste na utilização de limiares elevados associados a baixos níveis de particionamento espaço-temporal. Isso pode ser justificado pela maior probabilidade de limiares elevados conterem a área de interesse de forma integral, reduzindo

a necessidade de recuperação de múltiplos blocos neste contexto de consulta.

A análise consolidada para o regime de Instante Temporal (IT) indica que, diferentemente dos regimes de janela (deslizante e aleatória), a eficiência do sistema é inversamente proporcional ao nível de particionamento espaço-temporal. Os dados revelam que o nível 0 (particionamento exclusivamente espacial) apresenta o melhor equilíbrio entre latência e taxa de recuperação. Conforme observado na Figura 54, o incremento para o nível 4 resultou em uma redução da taxa de recuperação de até $78,2\%$, evidenciando que o refinamento temporal excessivo é ineficaz para consultas que demandam apenas um ponto isolado no tempo.

Referente ao desempenho temporal, a configuração de maior eficiência foi obtida com o limiar de 360000 no nível 1, registrando aproximadamente 52 ms (Figura 57). Este resultado contrasta com o desempenho de limiares reduzidos (5625), que apresentaram latências superiores a 124 ms. Essa diferença se deve ao fato que blocos de maiores limiares possuem maior probabilidade de conter a ROI de forma integral, minimizando o custo de gerência de múltiplos blocos e metadados para uma única resposta de instante temporal.

Embora a compressão `zstd` tenha reduzido a perda na taxa de recuperação e reduzido a latência entre 10% e 15%, ela não altera a tendência de degradação provocada pelo particionamento espaço-temporal excessivo. Pode se concluir que, para o contexto de consultas de instante temporal, a estratégia de otimização mais eficaz consiste na utilização de limiares elevados associados a baixos níveis de particionamento. Nestas condições, embora o particionamento reduza a taxa de recuperação em instantes temporais, os ganhos de desempenho alcançados em buscas de séries temporais superam essa degradação, justificando a aplicação da metodologia proposta no balanço global do sistema dentro deste contexto de buscas.

4.4.1.6 Discussão do Cenário Real

A análise integrada dos resultados experimentais do Dympas para cenário real permite identificar um compromisso entre a eficiência de armazenamento e o desempenho temporal das operações de ingestão e recuperação. No que se refere à infraestrutura, a adoção do algoritmo de compressão `zstd` proporcionou uma economia média de $40,07\%$ no consumo de memória secundária, conforme ilustrado na Figura 35. Portanto, essa redução implica um custo computacional significativo durante a persistência, elevando o tempo de inserção em aproximadamente seis vezes quando comparado ao formato não comprimido, conforme evidenciado na Figura 36. Esse comportamento caracteriza o sistema como altamente dependente da largura de banda de I/O. Apesar do custo de descompressão, os dados compactados apresentaram tempos de consulta inferiores em cenários de janelas temporais extensas, onde a redução no volume de transferência da memória secundária entregam uma redução maior do que o custo computacional de descompressão do `zstd`.

O parâmetro de limiar de particionamento consolidou-se como o principal determinante da escalabilidade da estrutura espaço-temporal. O limiar de 360000 demonstrou ser a configuração mais robusta, superando limiares reduzidos como o de 5625, que, apesar de oferecerem maior taxa de recuperação no particionamento espacial inicial, introduzem um *overhead* de gerência de blocos, além de não permitir particionamento tão bem direcionados com limiares maiores. Deste modo, degrada o desempenho em níveis de particionamento mais profundos. Assim, o impacto do particionamento espacial é pequeno em comparação ao processo de compressão no tempo de persistência, representando menos de 7% do tempo total, o que valida a viabilidade da estrutura de particionamento espaço-temporal proposta para o Dympas.

Quanto à recuperação de dados, os resultados revelam que o nível ideal de particionamento é estritamente dependente da natureza da carga de trabalho. Para consultas de janela deslizante, a taxa de recuperação e o desempenho temporal escalam positivamente até o nível 4, atingindo uma taxa de recuperação de dados úteis superior a 94%, como detalhado na Figura 42. Esse ganho de eficiência é atribuído ao reúso efetivo do *cache*, permitindo que o uso otimizado dos blocos com dados espaço-temporais. Entretanto, as consultas de janela aleatória encontram seu ponto de inflexão no nível 2, em que o particionamento excessivo gera custo de acesso que não é compensado pelo incremento da taxa de recuperação, conforme observado na Figura 49.

A análise das consultas de instante temporal apresenta o cenário de maior restrição para a metodologia de particionamento espaço-temporal. Nestes casos, o desempenho é inversamente proporcional ao nível de profundidade da estrutura, sendo o nível 0 ou 1 o mais eficiente para minimizar a latência, conforme aponta a Figura 57. A degradação observada em níveis superiores, como o nível 4, decorre da necessidade de processar múltiplos blocos e metadados para uma resposta de um instante temporal, o que aumenta o tempo de resposta em até $19,7\%$ em dados não comprimidos. Conclui-se que a otimização global do sistema exige uma calibração estratégica entre o limiar elevado e o nível de particionamento, priorizando o nível de particionamento espaço-temporal para séries temporais e limiares baixos para consultas de instante temporal, reduzindo a transferência de dados via `zstd`.

4.4.2 Resultados para experimentos sintéticos

Esta seção apresenta os resultados obtidos a partir de testes conduzidos em um contexto sintético, o qual permite avaliar o sistema sob condições controladas e estabelecer comparações com os dados do Cenário Real. Dessa forma, busca-se evidenciar a eficiência da metodologia proposta de maneira abrangente. São analisados os mesmos parâmetros do Cenário Real, abrangendo aspectos de infraestrutura, persistência e particionamento, além de uma investigação detalhada das consultas de janela deslizante, janela aleatória e instante temporal. Esta análise permite expandir a compreensão sobre o Dympas para

além dos limites permitidos pelos resultados do cenário real.

4.4.2.1 Persistência e Particionamento

Nesta etapa, é detalhado as métricas de desempenho do Dympas relativas ao consumo de memória secundária, ao tempo médio de ingestão de dados matriciais e à latência do particionamento espaço-temporal. Os resultados consolidados para o cenário sintético encontram-se no Apêndice (Seção A.6).

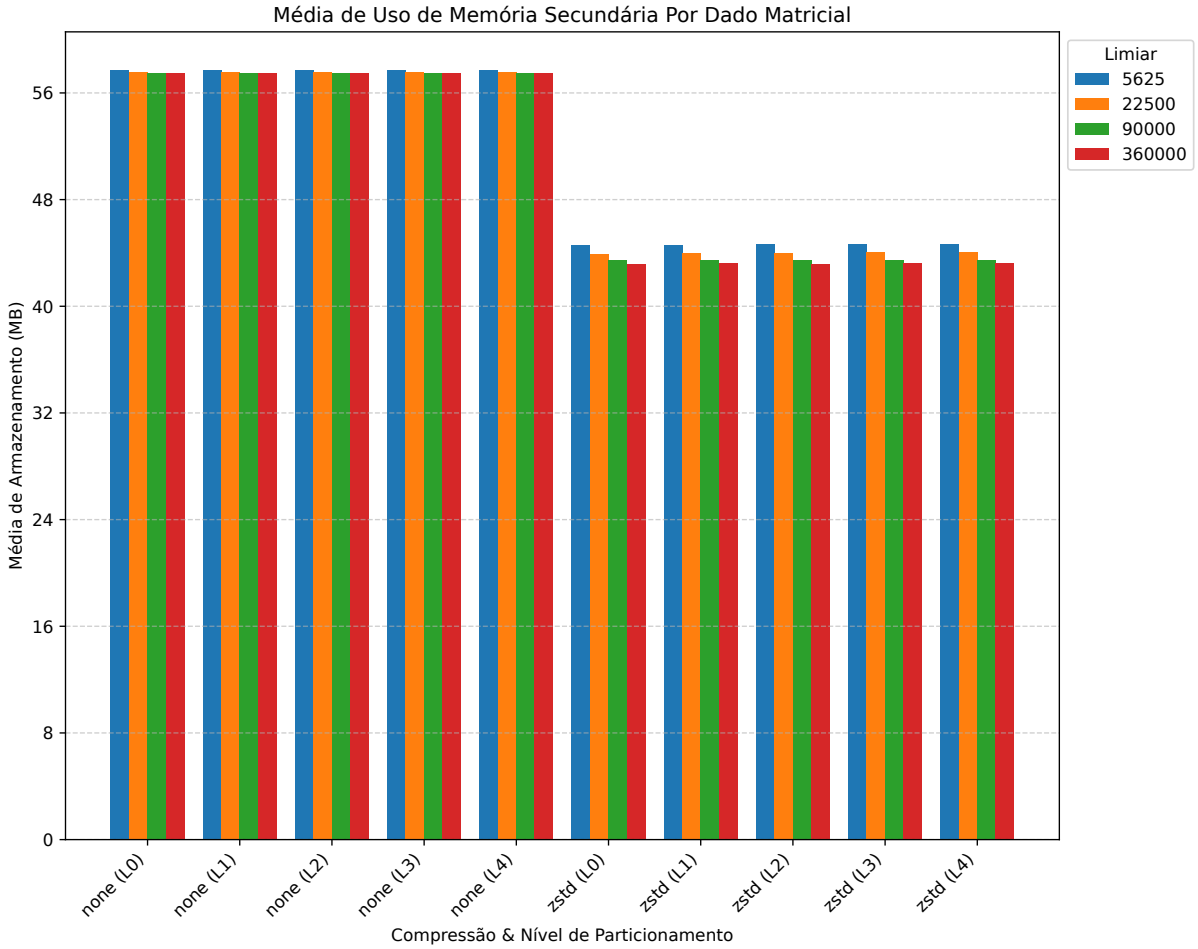

Figura 58 – Tamanho médio de uso da memória secundário por dado matricial inserido no Dympas no cenário sintético.

A análise do consumo médio de memória secundária por ingestão de dado matricial no Dympas é apresentada na Figura 58, que ilustra o custo de armazenamento para diferentes configurações de compressão, limiar e nível de particionamento. As siglas $L0$ a $L4$ representam, respectivamente, os níveis de particionamento de 0 a 4. O emprego do algoritmo `zstd` proporcionou redução no consumo de memória secundária. O volume médio para dados sem compressão (`none`) foi de aproximadamente $60,36$ MB, enquanto os dados comprimidos (`zstd`) ocuparam $46,01$ MB. Tal valor representa uma economia média

de armazenamento de $23,77\%$ ao utilizar compactação, independentemente do limiar ou nível de particionamento adotado. Os resultados detalhados encontram-se na Tabela 29 do Apêndice A.

A configuração de maior ocupação foi observada no cenário sem compressão, com limiar 5625 e nível 4 ($60,50$ MB). O cenário de maior eficiência de armazenamento ocorreu com limiar 360000 no nível 0 sob compressão `zstd` ($45,29$ MB), representando uma redução de $25,14\%$ em comparação ao cenário de maior ocupação. A variação do limiar de 5625 para 360000 nos dados sem compressão resultou em uma redução de $0,31\%$ no tamanho do arquivo. Os resultados indicam que o fator determinante para a ocupação em memória secundária é a aplicação da compressão, visto que o limiar demonstrou influência irrelevante e os níveis de particionamento não apresentaram relevância significativa na persistência dos dados.

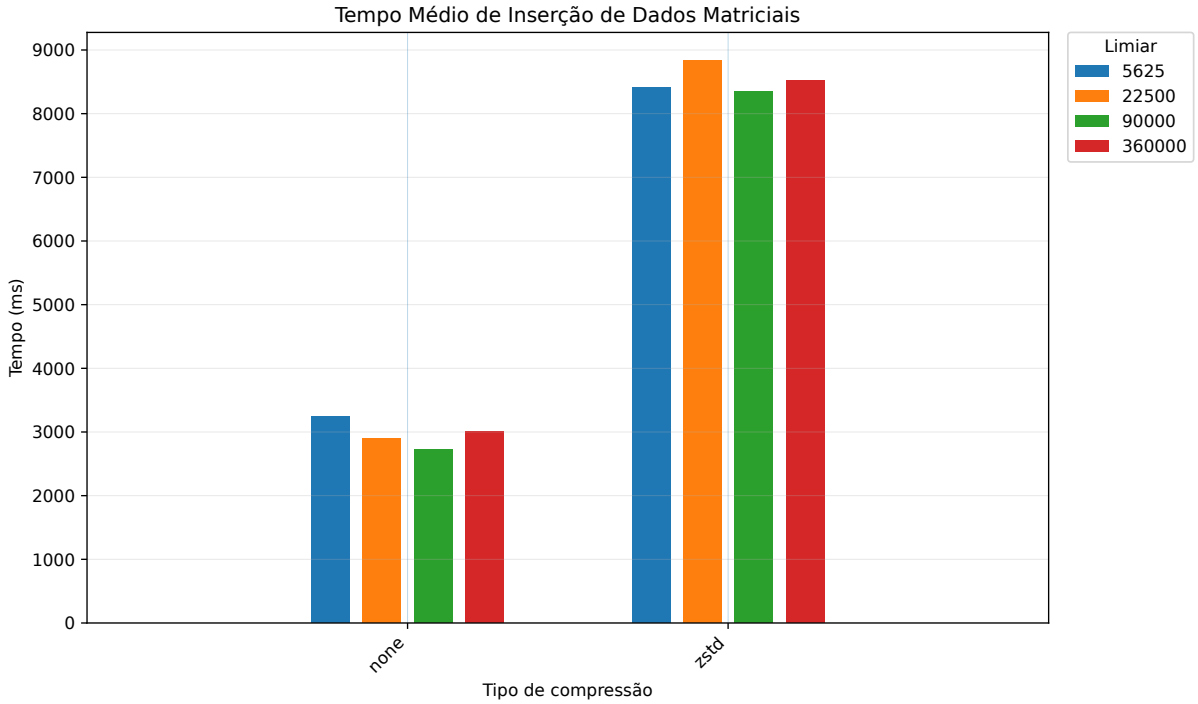

Figura 59 – Tempo médio de inserção de dados matriciais no Dympas por tipo de compressão e limiar no caso sintético.

A Figura 59 exibe os tempos médios de ingestão de dados matriciais no Dympas, avaliados sob diferentes parâmetros de limiar e compressão. Os dados sem compressão (`none`) apresentaram um desempenho até $69,07\%$ superior em relação aos dados comprimidos (`zstd`). O maior tempo de ingestão foi registrado na configuração com limiar 22500 sob `zstd` (8834 ms), enquanto o menor tempo ocorreu com o limiar 90000 sem compressão (2732 ms). Os resultados detalhados constam na Tabela 30 do Apêndice A.

Pode se observar o impacto do *overhead* de compactação, em que a ausência

de compressão resultou em uma redução média de 65, 16% no tempo de ingestão em comparação ao uso de `zstd`. Quanto à sensibilidade ao limiar, no cenário sem compressão, o aumento do limiar de 5625 para 90000 proporcionou uma redução de 15, 73% no tempo, embora tenha ocorrido uma elevação na marca de 360000. Portanto, para otimizar a latência de ingestão dos dados matriciais, é indicado a persistência dos dados sem compressão.

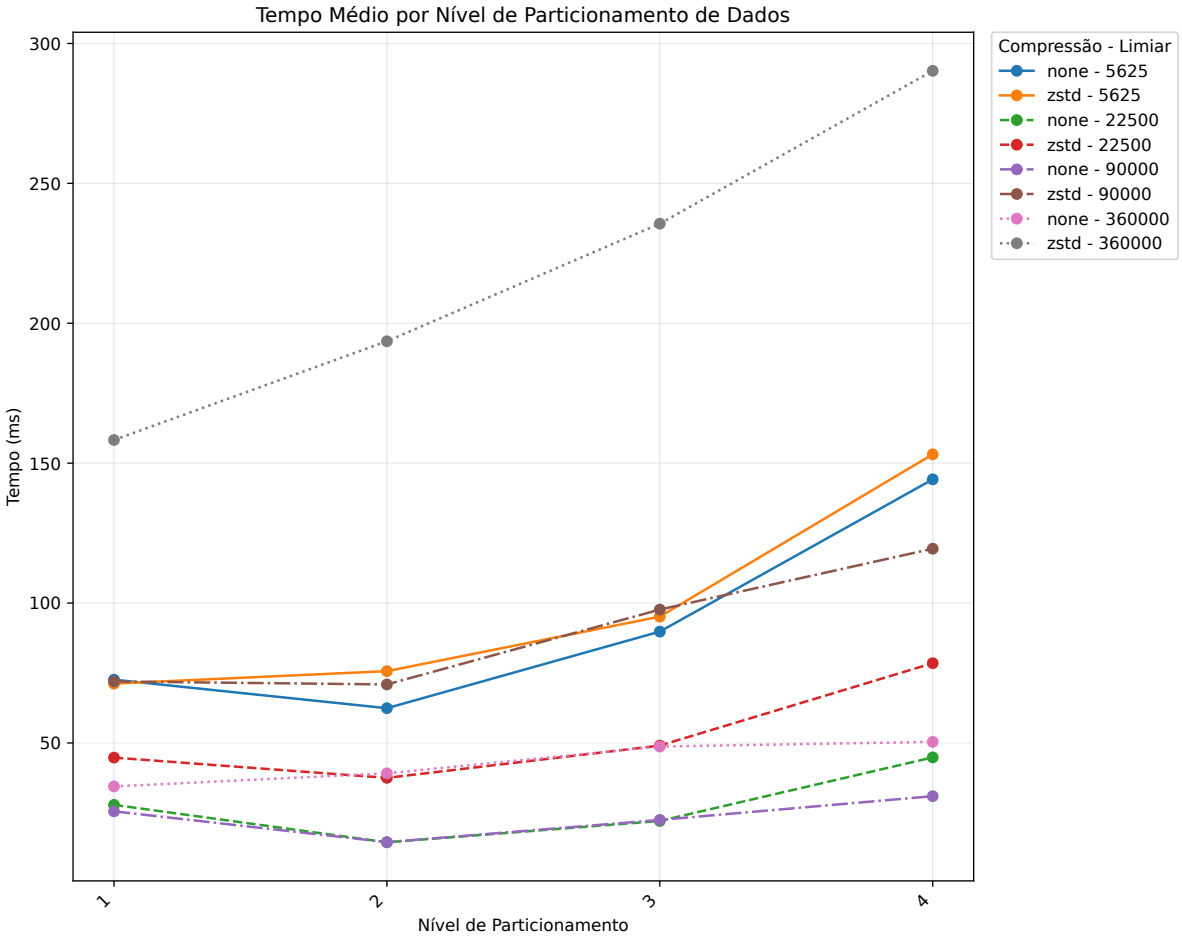

Figura 60 – Tempo médio de particionamento dos dados matriciais no Dympas para o caso sintético por nível de particionamento, tipo de compressão é limiar.

A Figura 60 exibe a análise do tempo de execução da operação de *split* (particionamento espaço-temporal) sob diferentes configurações de compressão, limiar e nível de particionamento. O particionamento ocorre de forma sequencial e incremental, do nível 0 ao nível 4. Os dados indicam que o limiar e o método de compressão são os fatores determinantes no custo da operação. A maior latência registrada foi de 290 ms(limiar 360000, sob `zstd`, na transição para o nível 4), enquanto a menor latência foi de 14 ms (limiar 22500, sem compressão, na transição para o nível 2). Os resultados detalhados constam na Tabela 31 do Apêndice A.

Em média, houve uma redução de 59.59% no tempo de transição entre níveis para os

dados sem compressão em comparação ao algoritmo `zstd`. Além disso, o nível 4 apresentou o maior *overhead* em todos os cenários, indicando que o custo de particionamento aumenta conforme a expansão do nível de particionamento espaço-temporal e o incremento do limiar.

A correlação dos dados de infraestrutura para o cenário sintético revela o comportamento estrutural do Dympas, onde a eficiência de armazenamento apresenta uma relação inversamente proporcional à agilidade de processamento de ingestão e particionamento. A economia foi de $23,77\%$ no consumo de memória secundária viabilizada pelo `zstd` (Figura 58), resultando em um acréscimo de $65,16\%$ no tempo médio de ingestão (Figura 59) e de $59,59\%$ no custo operacional de particionamento (Figura 60). Estes resultados indicam que o ganho de armazenamento é dependente, no qual a redução de tráfego em memória secundária é sobreposta pelo custo computacional exigido para a descompressão e recompressão de blocos de dados matriciais durante a manipulação de particionamento.

Existe uma neutralidade no armazenamento em relação ao nível de particionamento espaço-temporal. A estabilidade no tamanho dos arquivos entre o nível 0 e o nível 4, com oscilações inferiores a $0,31\%$ no cenário sem compressão, demonstra que o custo de persistência dos metadados de particionamento é irrelevante perante o volume dos dados matriciais. Entretanto, no desempenho temporal, referente ao armazenamento, a latência da operação de *split* é sensível tanto ao nível quanto ao limiar. Enquanto o aumento do limiar para 90000 otimizar a ingestão inicial reduzindo o tempo em $15,73\%$ (Figura 59), limiares elevados como 360000 maximizam o *overhead* de gerência no nível 4, atingindo 290 ms sob compressão (Figura 60).

Essa relação aponta que a configuração ideal para o Dympas depende da prioridade operacional. O uso de limiares reduzidos (5625) associado à Níveis de particionamento mais altos (nível 4) maximiza a taxa de recuperação de dados da consulta sem prejudicar o uso de memória secundária, mas impõe a maior penalidade de escrita e particionamento espaço-temporal. Ademais, o limiar de 90000 sem compressão se consolidou como o ponto de equilíbrio, minimizando a latência de inserção (2732 ms) e os custos de particionamento (14 ms), ainda que ceda na eficiência de armazenamento de $25,14\%$ observada nas configurações comprimidas. Portanto, o particionamento espaço-temporal é uma estratégia de otimização de busca que não gera custos de persistência, mas cujo custo de manutenção estrutural escala conforme a profundidade espaço-temporal e a densidade de compactação dos blocos.

4.4.2.2 Análise de Desempenho da Recuperação Espaço-Temporal

Esta seção descreve as análises de consultas espaço-temporais no cenário sintético, decompondo os resultados em diferentes regimes experimentais. São avaliadas as latências de consultas agrupadas das consultas de janelas deslizantes, janelas aleatórias e instantes

temporais em função do nível de particionamento, limiar físico e método de compactação. Os resultados detalhados encontram-se no Apêndice (Seção A.7).

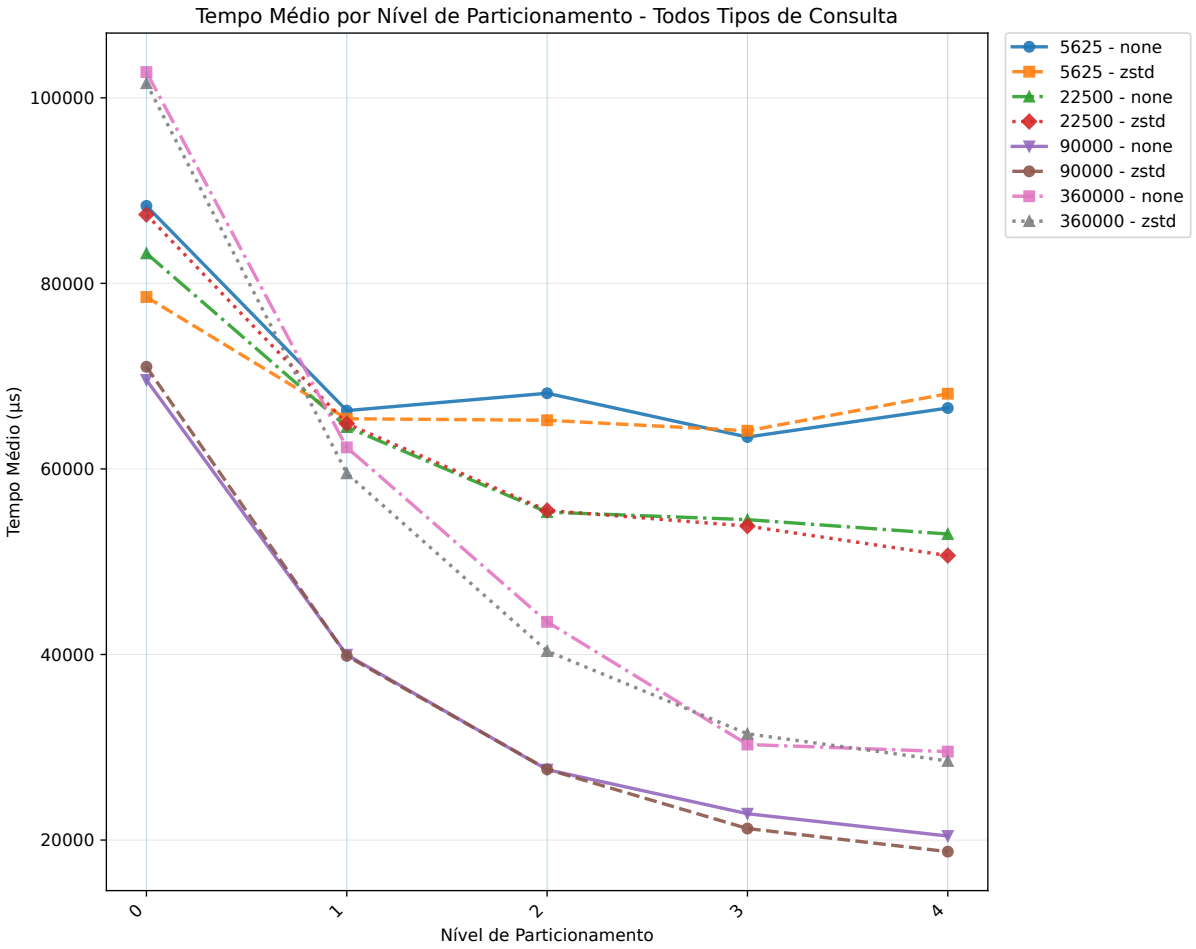

Figura 61 – Tempo médio das consultas (janela aleatória, janela deslizante e instante temporal) divididas por nível de particionamento, limiar e tipo de compressão.

A Figura 61 exibe o comportamento temporal do sistema Dympas integrando os diferentes regimes de busca. Com a latência mínima de $18,75$ ms foi registrada no nível 4 com o limiar 90000 com dados comprimidos (`zstd`), enquanto o cenário de menor eficiência ocorreu no nível 0 com limiar 360000 sem compressão, atingindo $102,77$ ms. Esta disparidade representa uma redução global de latência de $81,75\%$, evidenciando que a otimização do sistema é dependente da relação entre o nível do particionamento e o limiar. Deste modo, a transição do nível 0 para o nível 4 sob os melhores limiares reduz o tempo médio de $69,59$ ms para $18,75$ ms (73%), consolidando a estrutura espaço-temporal como relevante método de redução de tempo de consulta.

A análise de sensibilidade ao limiar indica que a escolha de limiares maiores, como 90000 e 360000, favorece a redução de latência em níveis superiores de particionamento em comparação a limiares reduzidos como 5625. No nível 4, o uso do limiar 90000 em

comparação do limiar 5625 resultou em uma redução de latências de 72%. Isso sugere que limiares elevados minimizam a quantidade inicial de blocos para recuperação do ROI, permitindo que os particionamentos espaço-temporais subsequentes sejam mais direcionados ao ROI. Enquanto limiares menores apresentam curvas de tendência que estabilizam em latências mais altas, limiares maiores demonstram maior potencial de otimização contínua até o nível 4. A compressão `zstd` atua como um ganho com pouca relevância, proporcionando uma redução adicional de $8,2\%$ no tempo de resposta no ponto de operação ideal ($20,43$ ms para $18,75$ ms), sem introduzir custos computacionais que anulem o benefício da redução recuperação de dados da memória secundária. Por fim, a tendência de estabilização das curvas no nível 4 indica que o incremento do particionamento espaço-temporal além deste nível apresenta rendimentos decrescentes de consulta avaliadas neste contexto.

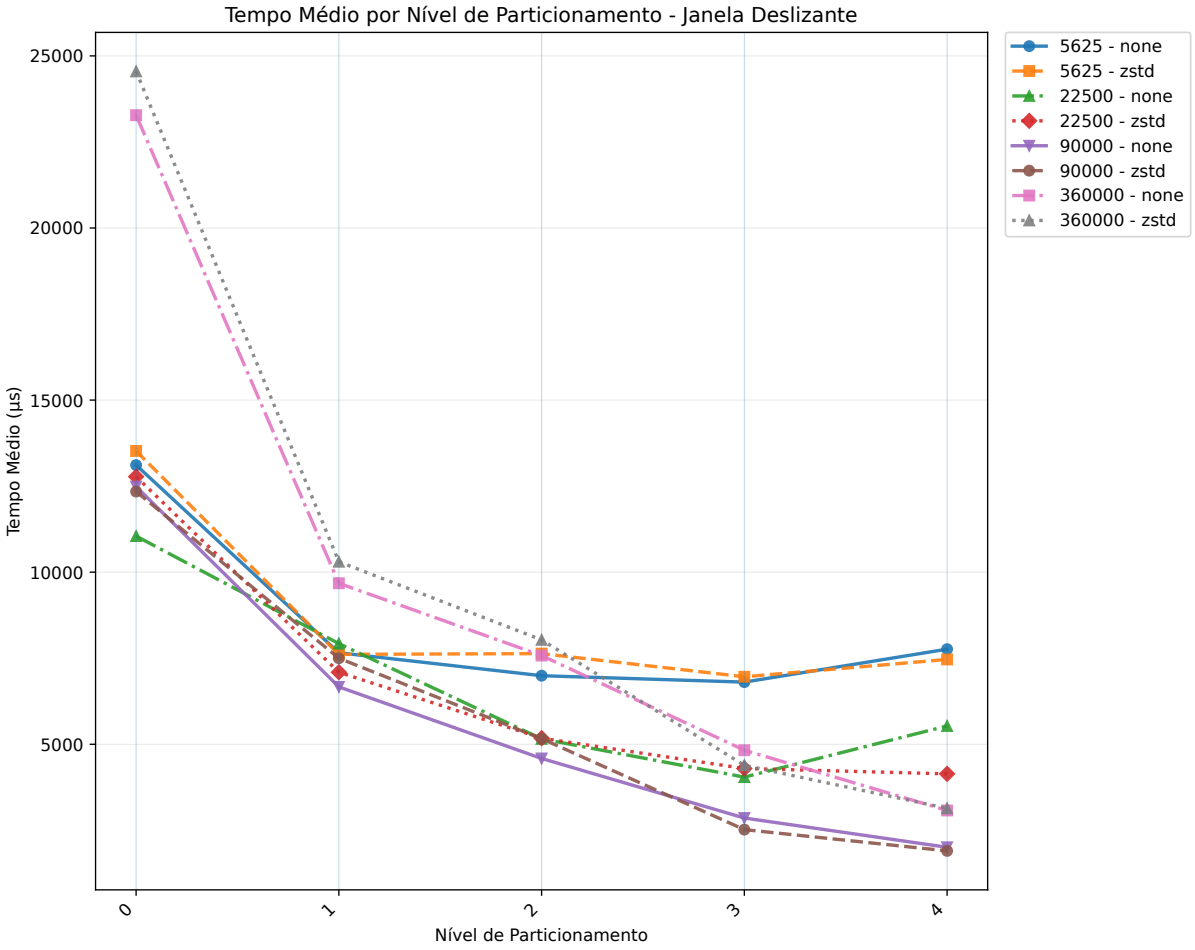

Figura 62 – Tempo médio das consultas de janela deslizante divididas por nível de particionamento, limiar e tipo de compressão para o cenário sintético.

A análise do desempenho para o regime de janela deslizante no cenário sintético, conforme ilustrada na Figura 62, demonstra a eficácia do particionamento espaço-temporal

na redução da latência de consulta. Este padrão de consulta favorece a utilização do *cache*, resultando em latências reduzidas, nas quais os valores registrados representam o tempo médio de consulta para o processamento de uma janela integral. O ponto de operação ideal foi registrado no nível 4, com o limiar 90000 sob compressão `zstd`, atingindo 1.90 ms. Em contraste, a maior latência ocorreu no nível 0, com limiar 360000 e compressão `zstd`, registrando 24.55 ms. Essa amplitude reflete uma redução de latência global de 92.3%, evidenciando a capacidade de otimização do Dympas em série de dados matriciais. Os resultados detalhados constam na Tabela 38 do Apêndice A.

A transição entre os níveis de particionamento revela que a mudança do nível 0 para o nível 4, considerando as melhores configurações de cada estágio (limiar 22500 sem compressão no nível 0 e limiar 90000 com `zstd` no nível 4), proporcionou uma redução temporal de 11.05 ms para 1.90 ms (82.8%). No nível 4, a escolha do limiar 90000 em detrimento do limiar 5625 resultou em uma redução de 75.5% na latência (7.76 ms para 1.90 ms). O impacto da compactação, embora pequeno em consultas já otimizadas, proporcionou uma melhora adicional de 5.2% no nível 4 com limiar 90000, reduzindo o tempo de 2.01 ms para 1.90 ms ao utilizar compreensão `zstd`.

Os dados indicam que o limiar 90000 se mostrou a configuração mais eficiente, pois permite ao sistema descartar volumes significativos de dados irrelevantes enquanto a compressão `zstd` otimiza a transferência da memória secundária. Também pode ser observado uma recuperação de desempenho notável para o limiar 360000. Embora apresente latências elevadas no nível 0, a progressão do particionamento permitiu que seus resultados convergissem para os patamares de melhor desempenho no nível 4. Ademais, pode verificar que um ponto de inflexão na eficácia do particionamento conforme a escala do limiar. O limiar 5625 apresentou estabilização de latência a partir do nível 1, enquanto o limiar 22500 estabilizou a partir do nível 2. Esse comportamento reforça que, para limiares reduzidos, o incremento do nível de particionamento excessivo não resulta em ganhos de performance proporcionais, devido ao limite de granularidade da ROI.

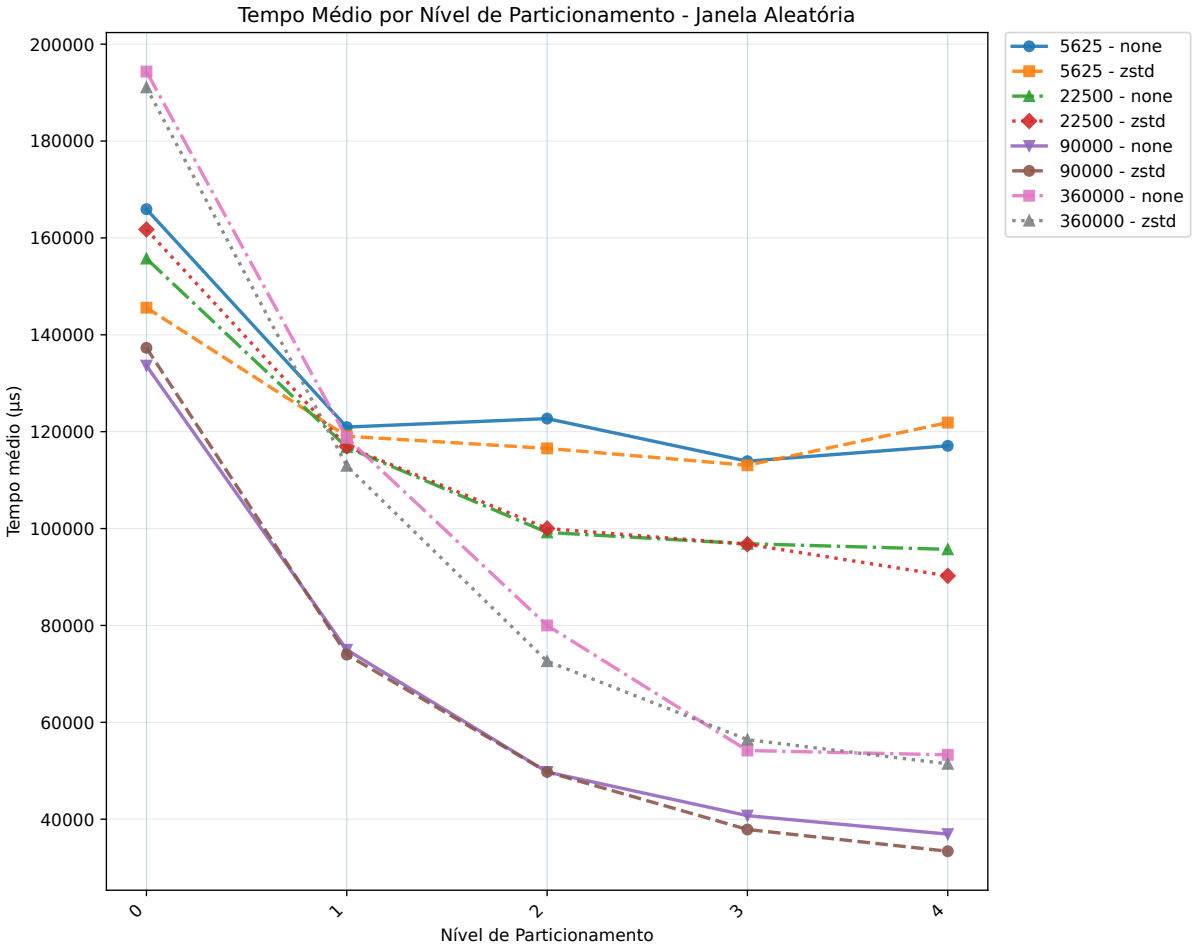

Figura 63 – Tempo médio das consultas de janela aleatórias divididas por nível de particionamento, limiar e tipo de compressão.

A Figura 63 exibe a análise de tempo de resposta das consultas de janelas aleatórias por limiar, nível de particionamento e método de compactação. Neste regime, os valores registrados representam o tempo médio para a recuperação integral dos dados de uma janela, existe uma redução no reaproveitamento de *cache* em comparação ao regime de janela deslizante. O ponto de operação ideal ocorreu no nível 4, com o limiar 90000 sob compressão `zstd`, registrando $33,39$ ms. O cenário de menor eficiência foi registrado no nível 0, com limiar 360000 sem compressão, atingindo $194,35$ ms. Essa diferença reflete uma redução de latência global de $82,8\%$. Os dados detalhados constam na Tabela 45 do Apêndice A.

A transição do nível 0 para o nível 4, considerando as configurações mais eficientes de cada estágio (limiar 90000 sem compressão no nível 0 e limiar 90000 com `zstd` no nível 4), proporcionou uma redução temporal de $133,64$ ms para $33,39$ ms (75%). No nível 4, a escolha do limiar 90000 em relação o limiar 5625 resultou em uma redução de $71,5\%$ no tempo de consulta ($117,08$ ms para $33,39$ ms). O impacto da compactação via `zstd` no

nível 4 com limiar 90000 gerou um ganho adicional de $9,6\%$ ($36,92$ ms para $33,39$ ms). Pode se observar que a compressão apresenta benefícios estatisticamente significativos apenas em níveis superiores de particionamento, consolidando-se como uma estratégia eficaz no nível 4 neste contexto de consultas.

Os resultados demostram o limiar 90000 como a configuração mais eficiente para este regime. Quando associado ao nível 4, este parâmetro permite que o sistema descarte volumes expressivos de dados irrelevantes, enquanto a compressão `zstd` otimiza a transferência da memória secundária sem gerar grande custo computacional. Ademais, o comportamento de estabilização precoce na latência para blocos de menor limiar é repetido neste regime. No caso do limiar 5625, o ganho de desempenho cessa a partir do nível 1. Esse comportamento reforça a premissa de que quanto menor o limiar inicial, mais rapidamente o sistema atinge seu ponto de inflexão, no qual incrementos no particionamento espaço-temporal não resultam em melhorias adicionais na taxa de recuperação da consulta e redução de tempo.

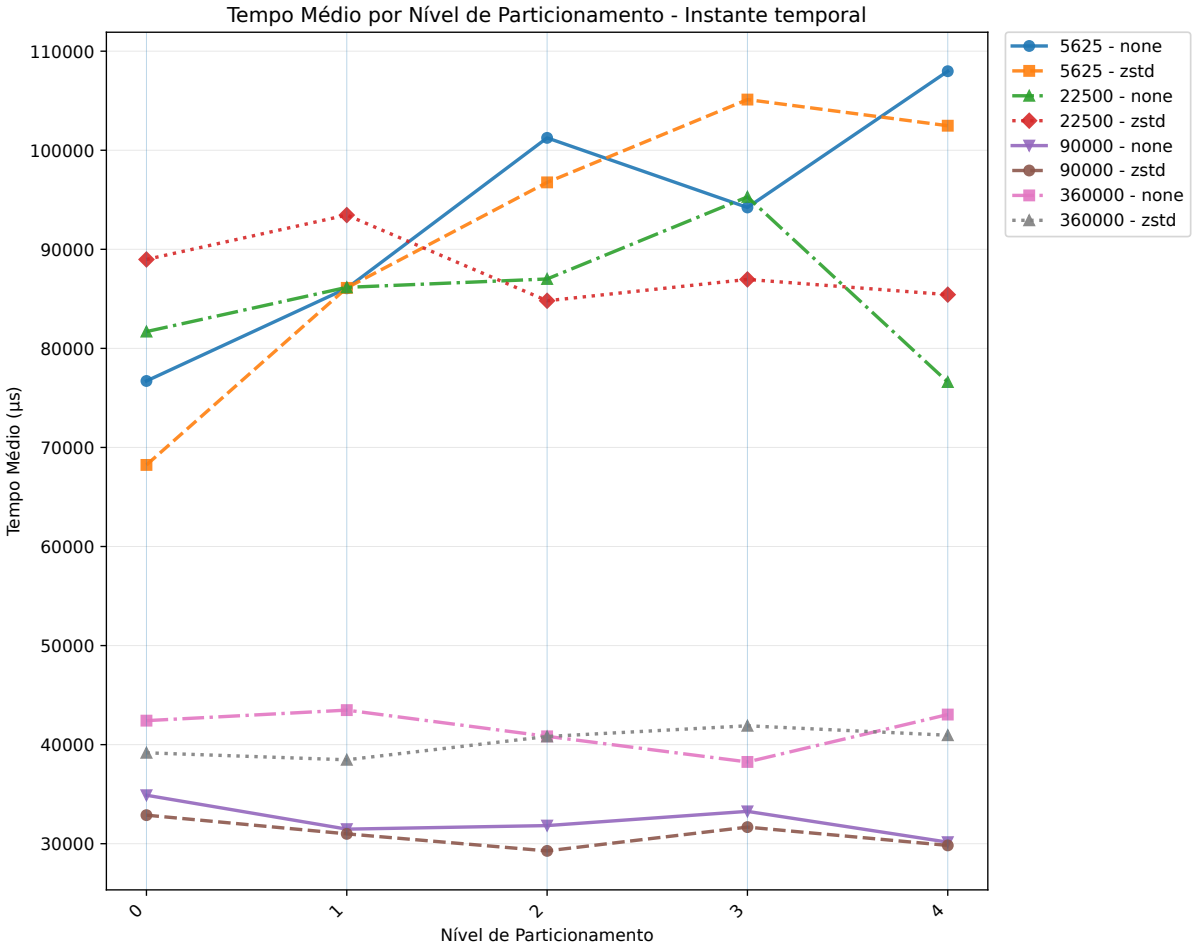

Figura 64 – Tempo médio das consultas de instante temporal divididas por nível de particionamento, limiar e tipo de compressão para o cenário sintético.

A análise do desempenho para o regime de instante temporal no cenário sintético,

conforme exibido na Figura 64, mostra uma polarização nos resultados baseada na magnitude do limiar, estabelecendo dois grupos distintos de eficiência. O ponto de operação ideal foi registrado no nível 2, com limiar 90000 sob compressão `zstd`, atingindo 29, 27 ms. Em contrapartida, o cenário de menor eficiência ocorreu no nível 4, com limiar 5625 sem compressão, registrando 107, 98 ms. Essa amplitude reflete uma redução de latência global de 72, 9%, em que consultas pontuais, a configuração do limiar é mais determinante que o nível do particionamento isoladamente neste contexto de consultas. Os dados detalhados constam na Tabela 52 do Apêndice A.

Os resultados mostram uma variação estrutural, enquanto os limiares reduzidos (5625 e 22500) estabilizam em latências elevadas (acima de 90 ms), os limiares superiores (90000 e 360000) operam com consistência abaixo de 35 ms. No nível 2, a escolha do limiar 90000 em referência ao limiar de 5625 reduz a latência de 96, 76 ms para 29, 27 ms (69, 7%). Esse comportamento indica que limiares maiores favorecem a localidade espacial do ROI, garantindo que o instante temporal esteja contido em um número reduzido de blocos, o que minimiza o custo de gerência de múltiplos blocos.

Diferente dos regimes de janela, o particionamento para instantes temporais atinge seu benefício máximo precocemente. A transição do nível 0 para o nível 2 (limiar 90000, `zstd`) proporcionou um ganho de 11% (32, 88 ms para 29, 27 ms). Entretanto, a progressão para o nível 4 introduziu um *overhead* que elevou a latência para 29, 83 ms. A compressão `zstd` contribuiu com uma melhoria de 8%, mostrando que a redução da transferência de dados compensa o tempo de descompressão. Assim, a estratégia ótima para este regime reside na utilização de limiares elevados (90000) associados a níveis intermediários de particionamento (nível 2), onde a taxa de recuperação é maximizada.

A integração dos resultados evidencia que a eficiência do Dympas é maximizada pela relação entre o particionamento espaço-temporal e o limiar. Enquanto a análise unificada (Figura 61) aponta uma redução global de latência de 81, 75%, as análises isoladas por regimes revela que o sistema atinge sua performance máxima em consultas de janela deslizante (Figura 62), com tempos de resposta de 1, 90 ms no nível 4. Esse desempenho superior no regime deslizante, comparado aos 33, 39 ms das janelas aleatórias (Figura 63), demonstra que a estrutura de particionamento não apenas reduz o tráfego de dados irrelevantes, mas também potencializa o reaproveitamento de *cache* em acessos de séries temporais, pode ser destacado o limiar 90000 como o ponto de equilíbrio ideal em todos os cenários de janela.

No regime de instante temporal (Figura 64), observa-se o cenário de maior desafio para a estrutura, uma vez que a busca por um único ponto no tempo limita o benefício do particionamento espaço-temporal em comparação aos regimes de janela. Embora o instante temporal represente tecnicamente o pior caso de consulta devido à menor taxa de recuperação, os resultados mostram que não ocorre uma degradação acentuada de

performance. Assim, a latência é estabiliza em patamares próximos a 29 ms nos melhores limiares. Diferente dos regimes de janela, em que o nível 4 é o que mais se destaca, o instante temporal encontra seu ponto máximo no nível 2, indicando que o custo de gerência de múltiplos blocos no nível 4 gera um *overhead* residual que, contudo, não compromete a viabilidade do sistema para buscas de instante temporais.

A correlação entre esses regimes valida o Dympas como uma alternativa robusta para a gerência de dados matriciais. A ligeira perda de eficiência observada em instantes temporais sob níveis elevados de particionamento é amplamente compensada pelos ganhos nos regimes de janela, que variam de 73% a mais de 92% de redução no tempo de resposta (Figura 62). A estabilidade do limiar 90000 em todos os testes reforça que a arquitetura consegue manter uma localidade de dados eficiente, garantindo que o custo de manutenção da estrutura espaço-temporal seja é viável e se traduz em alta escalabilidade para as operações de recuperação mais complexas do sistema.

4.4.2.3 Análise de Consultas de Janela Deslizante

Esta seção apresenta a análise das consultas de janela deslizante, decompondo os resultados em distintos cenários experimentais. São avaliadas as taxas de recuperação entre a memória secundária e a resposta, bem como o desempenho temporal das consultas. Todas as métricas são analisadas em função do limiar, do tamanho da janela e do tipo de ROI. Os resultados detalhados encontram-se no Apêndice (Seção A.8).

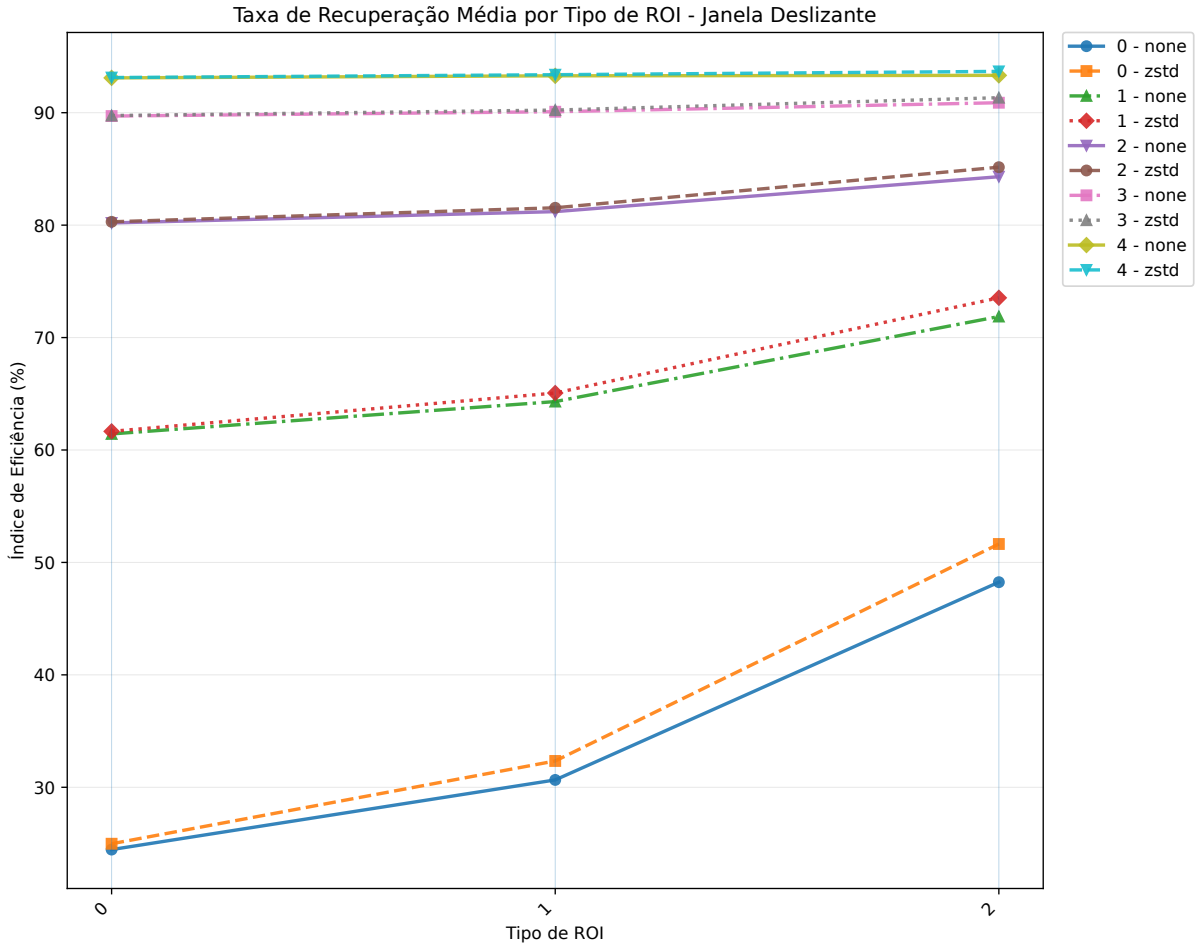

Figura 65 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tipo de ROI, nível de particionamento e tipo de compressão no cenário sintético.

A Figura 65 exibe a eficiência de recuperação, definida pela razão entre os dados recuperados da memória secundária e o tamanho da resposta. Esta métrica quantifica o aproveitamento dos dados recuperados, em que índices elevados indicam menor processamento de dados sem valor semântico para a consulta. Pode ser observado que a eficiência das consultas de janela deslizante é diretamente proporcional à profundidade do particionamento espaço-temporal. No nível 0, a taxa de recuperação oscilou entre $24,46\%$ e $48,24\%$, evidenciando um baixo aproveitamento na recuperação de dados da memória secundária. Em contrapartida, a progressão para o nível 4 resultou na convergência da eficiência para patamares superiores a 93% em todas as categorias de ROI. Os resultados detalhados constam na Tabela 36 do Apêndice A.

Pode ser observado que o impacto dos tipos de ROI reduz conforme o incremento do nível de particionamento. Além disso, existe uma redução gradual no ganho da taxa de recuperação de cada nível de particionamento, indicando uma tendência de estabilização.

Nesse sentido, particionamentos além do nível 4 tendem a apresentar ganhos irrelevantes. Adicionalmente, o emprego da compressão `zstd` exerceu pouca relevância na taxa de recuperação em alto níveis de particionamento (Níveis 3 e 4), sugerindo que a taxa de recuperação em particionamento espaço-temporal é principal fator para a eficiência de recuperação de dados matriciais nesses cenários.

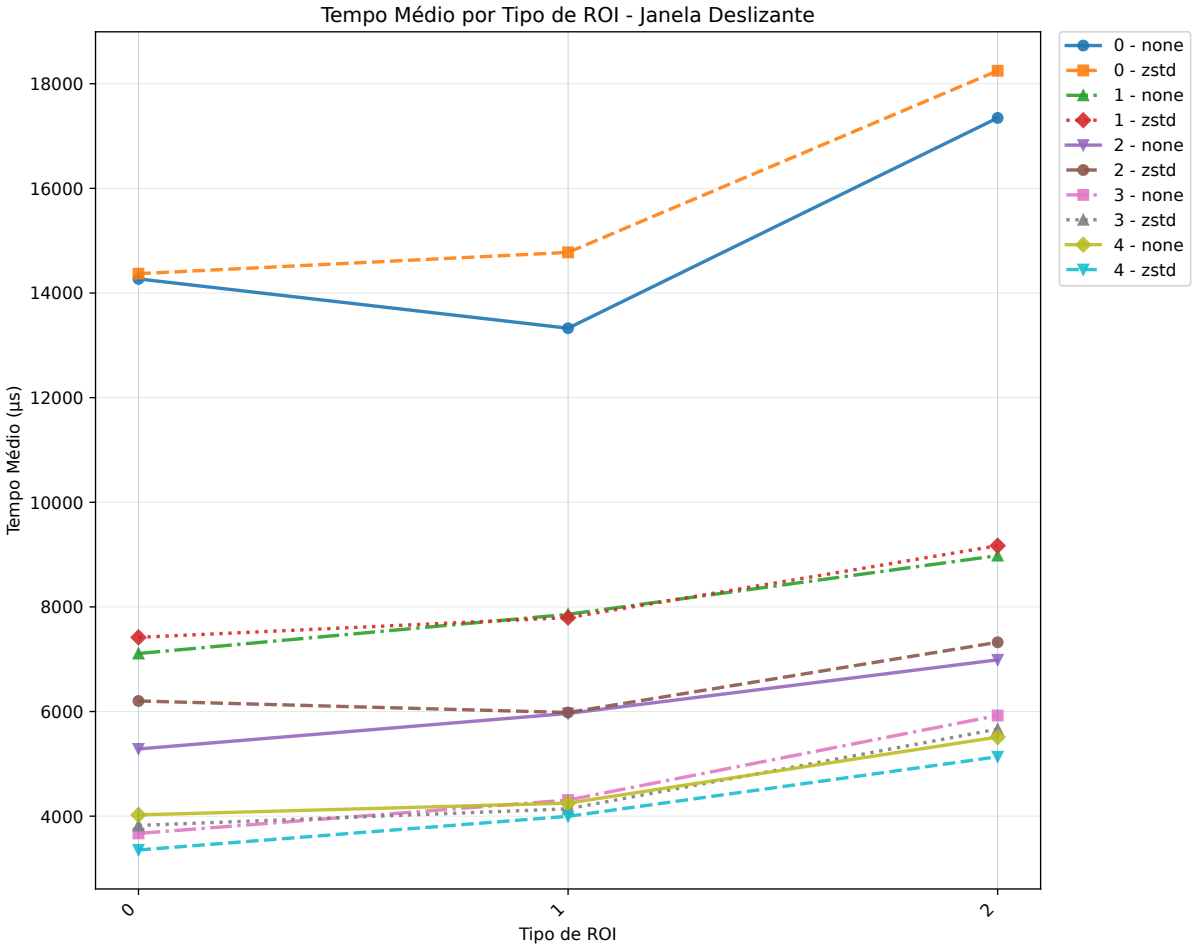

Figura 66 – Média de tempo das consultas agrupadas por tipo de ROI por nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 66 exibe o tempo médio das consultas por janela deslizante por tipo de ROI, evidenciando a eficácia do particionamento espaço-temporal na redução do custo temporal de recuperação de dados de interesse. Pode ser observado que o aumento do nível 0 para o nível 4 promoveu uma redução de latência superior a 70% em todas as categorias de ROI. No ROI 0 sob compressão `zstd`, o tempo médio reduziu de $14,37$ ms no nível 0 para $33,53$ ms no nível 4, uma redução de $76,6\%$. Estes resultados demonstram que o particionamento espaço-temporal é eficaz para diferentes dimensões de ROI, o que, aliado ao uso de *cache*, potencializa a redução da latência de consulta, com uma tendência de estabilização observada nos níveis de particionamento superiores (Níveis 3 e 4). Os dados

detalhados constam na Tabela 37 do Apêndice A.

A aplicação da compressão não apresenta vantagens generalizadas no tempo de resposta das consultas, visto que seu benefício é pontual e não uniforme em todos os cenários. Em níveis elevados de particionamento (nível 4), os dados comprimidos apresentaram latências inferiores aos dados não comprimidos em todas as categorias de ROI. O ROI de tipo 2 tem o tempo médio reduziu de $5,51$ ms (`none`, dado sem compressão) para $5,14$ ms (`zstd`). Contudo, tal ganho é pontual, uma vez que nos níveis iniciais de particionamento a ausência de compressão apresenta melhor desempenho. Isso indica que a redução na carga de leitura proporcionada pela compressão compensa o custo de decodificação em memória primária apenas quando a taxa de recuperação em particionamento espaço-temporal já é elevada, não sendo uma vantagem real de modo geral para todo o processo de consulta.

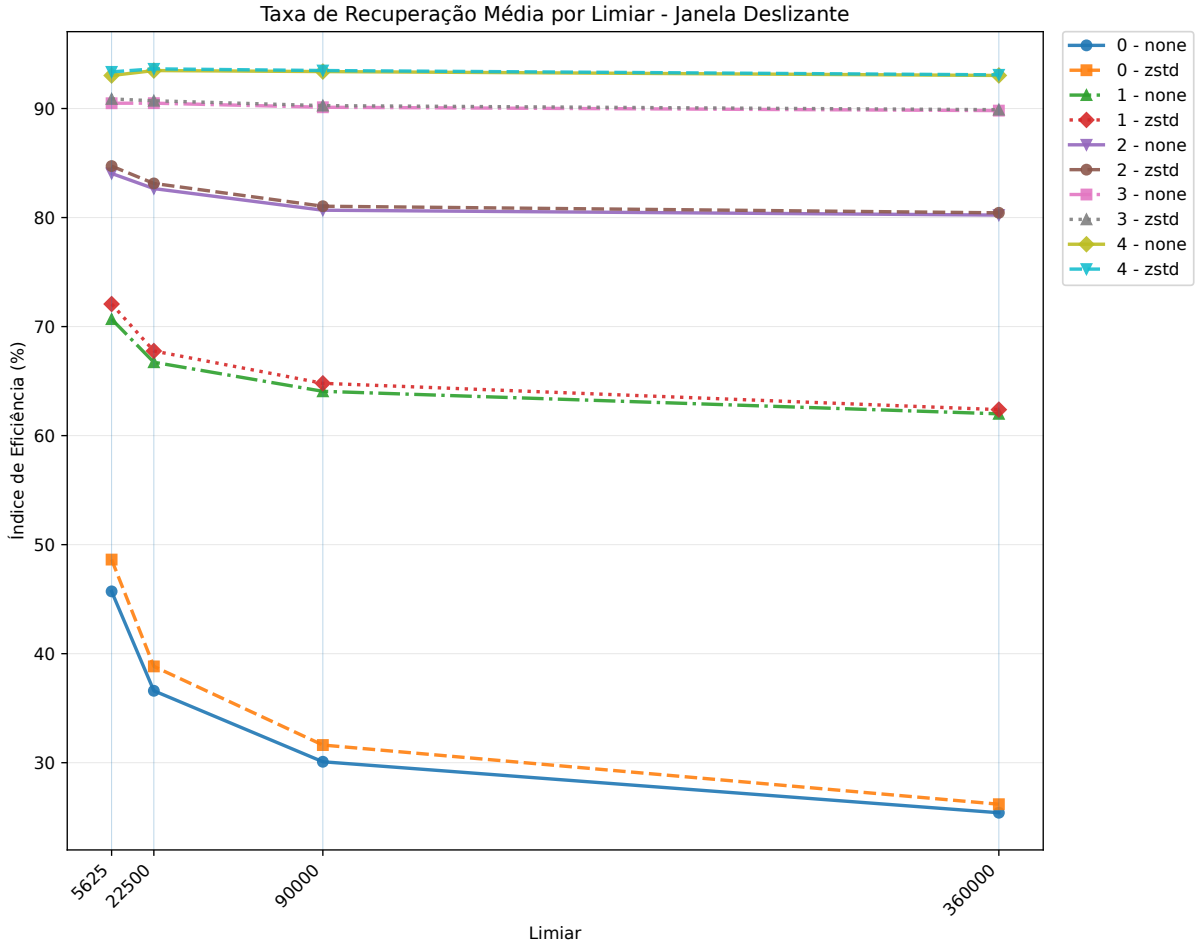

Figura 67 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por limiar, nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 67 exibe a taxa de recuperação média sob diferentes limiares de particiona-

mento, evidenciando que a eficiência inicial do sistema é sensível ao valor do limiar aplicado apenas nos níveis iniciais de particionamento (nível 0, 1 e 2). No nível 0, o limiar 5625 sem compressão apresentou uma taxa de $45,71\%$, enquanto o limiar 360000 demonstrou uma taxa de $25,41\%$. Este comportamento indica que limiares menores favorecem um melhor aproveitamento de recuperação de dados da memória secundária no nível 0, reduzindo o volume de dados sem valor semântico para a consulta antes do incremento do nível de particionamento. Os dados detalhados podem ser observados na Tabela 39 do Apêndice A.

Contudo, o impacto do limiar é gradualmente menos relevante à medida que o nível de particionamento espaço-temporal aumenta. No nível 4, ocorre uma convergência de desempenho, a taxa de recuperação fica estabilizada em patamares superiores a 93% para todos os limiares avaliados. O limiar 360000 sem compressão, que iniciou com a menor eficiência, atinge $93,03\%$ no nível 4, valor equivalente ao do limiar 5625 (93,03%). Esse resultado demonstra que o particionamento espaço-temporal em níveis maiores é o fator determinante para a taxa de recuperação, sendo capaz de equalizar a eficiência de recuperação independentemente do limiar de criação adotado. Ademais, a compressão `zstd` manteve uma pequena contribuição na melhoria da taxa de recuperação em todos os níveis, elevando a taxa máxima para $93,64\%$ no limiar 22500 (nível 4).

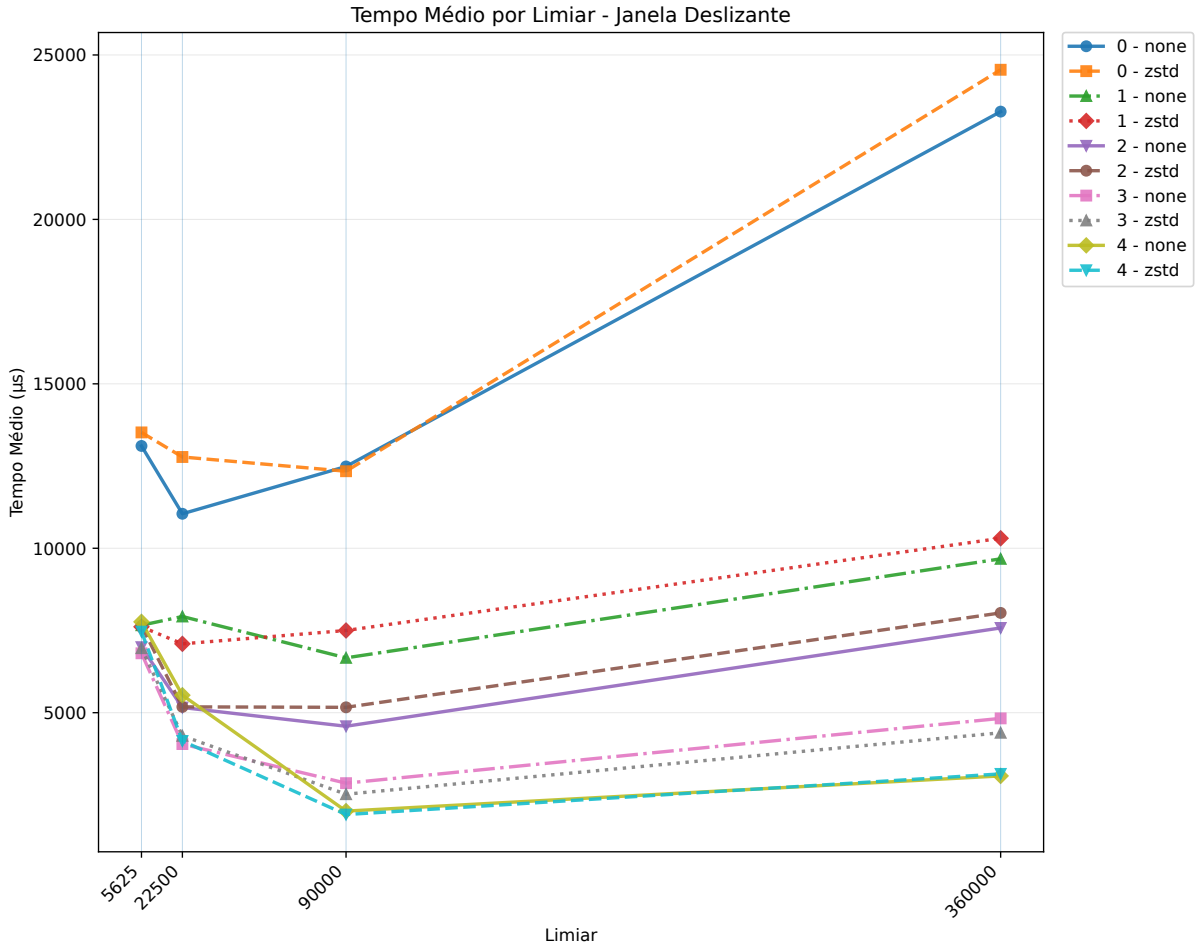

Figura 68 – Média de tempo das consultas agrupadas por limiar pelo nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 68 exibe o tempo médio das consultas por janela deslizante, demonstrando que a latência é condicionada pelo nível de particionamento e pelo limiar. O incremento do nível de particionamento resultou em uma redução de latência de até $87,2\%$ no limiar 360000 sob compressão `zstd`, reduzindo de $24,55$ ms (nível 0) para $3,14$ ms (nível 4). Esses resultados confirmam a eficácia do particionamento espaço-temporal na otimização de buscas em séries de dados matriciais, reduzindo o custo de recuperação conforme o nível de particionamento aumenta. Os dados detalhados constam na Tabela 40 do Apêndice A.

O limiar 90000 mostra como o ponto de máxima eficiência, com latência mínima de $1,90$ ms (nível 4, comprimido pelo `zstd`), enquanto limiares reduzidos (5625) apresentaram resultados de desempenho próximos, elevando o tempo de $6,81$ ms (nível 3) para $7,76$ ms (nível 4) sem compressão. A performance sob compressão `zstd` no nível 4 superou os dados sem compressão nos limiares 5625, 22500 e 90000. Contudo, no limiar 360000, a ausência de compressão foi superior ($3,08$ ms contra $3,14$ ms), indicando que o benefício da compactação é condicionado pelo equilíbrio entre o limiar e o custo de decodificação

em memória primária.

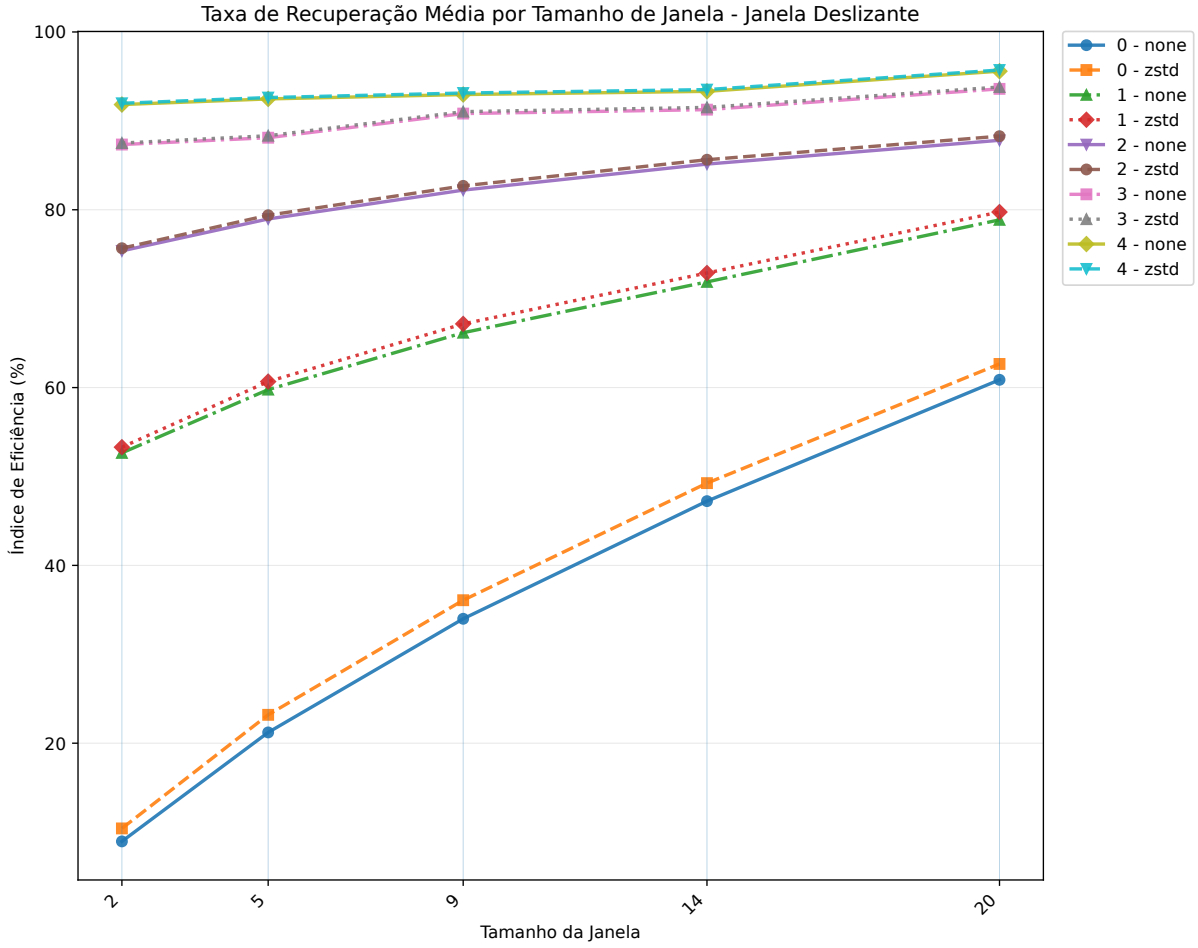

Figura 69 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tamanho de janela, nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 69 exibe a taxa de recuperação, definida pela razão entre dados recuperados da memória secundária e tamanho da resposta, sob diferentes extensões de janelas temporais. Pode se observar que a taxa de recuperação inicial é sensível ao tamanho da janela. No nível 0, a janela de tamanho 2 apresentou eficiência de apenas $8,95\%$, enquanto a janela de tamanho 20 atingiu $60,87\%$ de taxa de recuperação. Os ganhos de desempenho mais expressivos ocorrem nos estágios iniciais de particionamento espaço-temporal, onde a transição do nível 0 para o nível 1 elevou a taxa de recuperação em $43,7\%$ para a janela de tamanho 2 (de $8,95\%$ para $52,65\%$). Este comportamento demonstra que o particionamento espaço-temporal atua na redução do custo de leitura de dados sem valor semântico para a consulta a partir dos primeiros níveis. Os dados detalhados constam na Tabela 41 do Apêndice A.

À medida que o nível de particionamento espaço-temporal aumenta, uma tendência

de estabilização nos ganhos na taxa de recuperação. No nível 4, todas as extensões de janela convergiram para patamares de eficiência superiores a 91%, com a janela de tamanho 20 atingindo a taxa de recuperação máxima de 95, 72% sob compressão `zstd`. A redução nos incrementos de performance entre o nível 3 e o nível 4 indica que o sistema atinge um ponto de estabilização para as dimensões consultadas, reforçando que ao nível de particionamento espaço-temporal possui rendimentos decrescentes após atingir a precisão necessária recuperação de um ROI. Ademais, o emprego da compressão `zstd` manteve uma influência positiva, mas pequena, em todos os níveis, auxiliando no aproveitamento de recuperação de dados da memória secundária sem alterar a curva de convergência para este contexto de consultas.

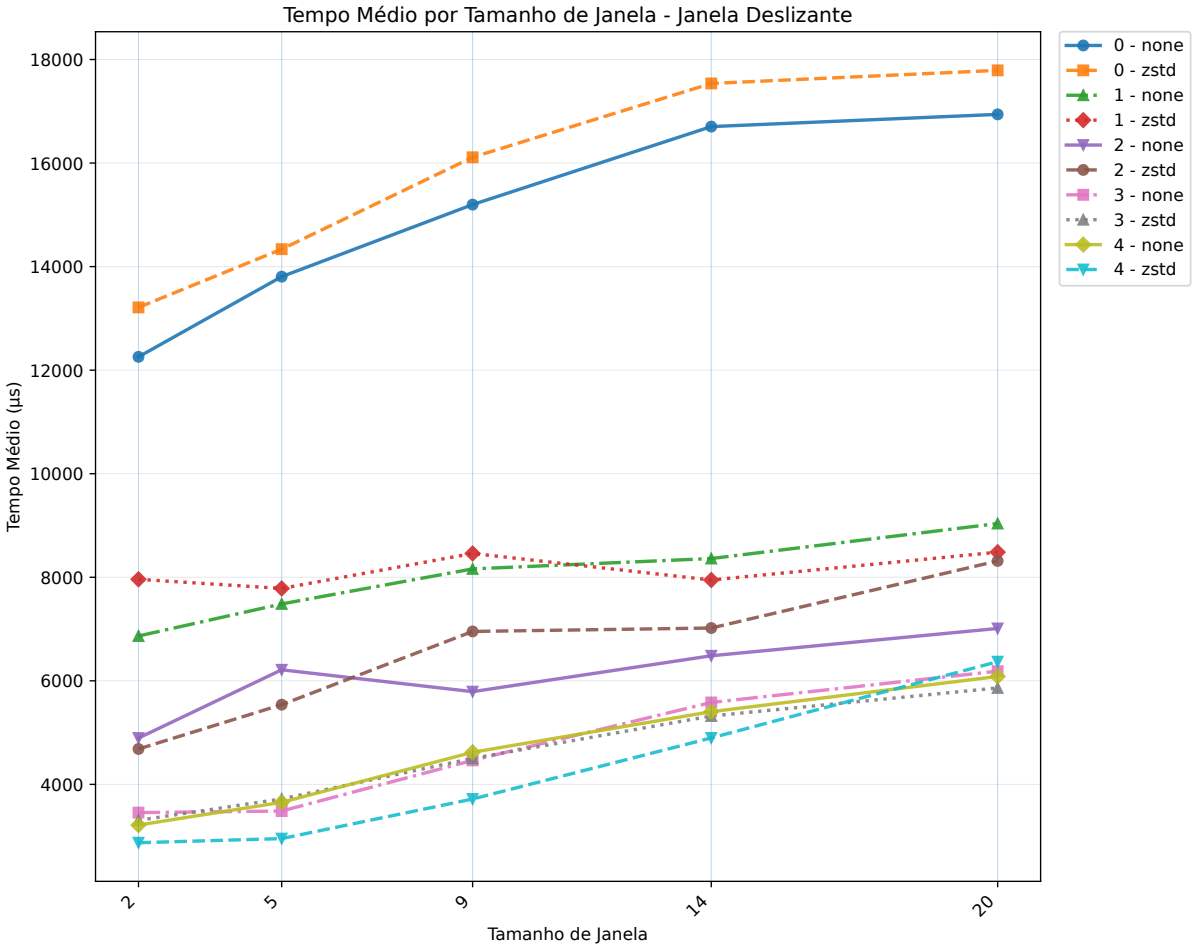

Figura 70 – Média de tempo das consultas agrupadas por tamanho de janela pelo nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 70 exibe a avaliação do tempo médio de resposta para consultas de janela aleatória reforça a tese de que o particionamento espaço-temporal é o principal motor de desempenho do sistema. O incremento do nível de particionamento (do nível 0 para o nível 4) promoveu uma redução média de latência superior a 70%, independentemente da

extensão da janela temporal consultada. Este resultado evidencia que a taxa de recuperação em particionamento espacial-temporal proporcionada pela estrutura em níveis reduz o custo de acesso à memória secundária. Os dados detalhados podem ser observados no Apêndice A na Tabela 42.

Observou-se, ainda, uma notável eficiência na escalabilidade temporal do sistema. No nível 4, o aumento de dez vezes na dimensão da janela ($T = 2$ para $T = 20$) resultou em um acréscimo de latência de apenas 110%, demonstrando que o custo de travessia estrutural é diluído em consultas de maior profundidade. Adicionalmente, a performance sob compressão `zstd` destacou-se nos níveis de particionamento mais elevados, superando os dados não comprimidos em termos de latência final. Tal fenômeno confirma que o processo de consulta no Dympas é limitado pela velocidade de leitura em memória secundária, onde o benefício da redução volumétrica dos dados comprimidos sobrepuja o custo computacional da descompressão em tempo real.

A Figura 70 exibe a análise do tempo médio de resposta para consultas de janela deslizante, reforçando que ao nível de particionamento espaço-temporal dos dados tem maior impacto na otimização de redução de tempo nas consultas. Pode se observar que o incremento do nível de particionamento promoveu uma redução média de latência superior a 60% entre o nível 0 e o nível 4. Na janela de tamanho 2 sob compressão `zstd`, o tempo médio decaiu de $13,21$ ms no nível 0 para $2,87$ ms no nível 4, representando uma redução de $78,3\%$. Os ganhos de desempenho mais acentuados ocorrem nos níveis mais baixos de particionamento (do nível 0 para o 1) na janela de tamanho 20 com dados sem compressão, a transição do nível 0 para o nível 1 reduziu a latência de $16,94$ ms para $9,04$ ms ($46,6\%$), evidenciando a eficácia da taxa de recuperação em particionamento espaço-temporal logo nos primeiros níveis. Os dados detalhados constam na Tabela 42 do Apêndice A.

À medida que o nível do particionamento espaço-temporal aumenta, uma tendência de estabilização no desempenho, indicando que o sistema atinge um ponto de estabilização em relação ao tamanho da janela. No nível 4, o aumento de dez vezes na dimensão da janela (tamanho 2 para tamanho 20) resultou em um acréscimo de latência de $89,4\%$ para os dados comprimidos ($3,21$ ms para $6,08$ ms). Contudo, a redução nos ganhos entre o nível 3 e o nível 4 sugere que particionamento espaço-temporal possui rendimentos decrescentes. Na janela de tamanho 20 sob compressão `zstd`, o tempo aumentou de $5,86$ ms para $6,37$ ms. A performance sob compressão `zstd` reafirma que apenas em cenários específicos de alta taxa de recuperação, onde a economia no recuperação de dados da memória secundária compensa o custo de descompressão em memória primária.

A análise integrada das consultas de janela deslizante evidencia que a eficiência de recuperação possui correlação direta com a profundidade do particionamento espaço-temporal. Independentemente do tipo de ROI, do limiar configurado ou da extensão da janela temporal, pode se observar uma taxa de recuperação superior a 93% ao atingir o

nível 4, conforme indicam a Figura 65, a Figura 67 e a Figura 69. A otimização do acesso é concentrada na transição do nível 0 para o nível 1, etapa na qual a estrutura minimiza a recuperação de dados sem valor semântico para a consulta. Esse ganho é isolado em cenários de janelas temporais reduzidas, nos quais o particionamento elevou a eficiência da recuperação de $8,95\%$ no nível 0 para $52,65\%$ no nível 1, conforme a Figura 69. Os resultados confirmam que a divisão espaço-temporal otimiza a taxa de recuperação da memória secundária e assegura a resiliência do sistema frente às variações de limiares dos blocos iniciais.

Em relação ao desempenho temporal, a redução da latência acompanha o ganho de taxa de recuperação em particionamento espaço-temporal, apresentando melhorias superiores a 70% em tipos de ROI no nível 4 (Figura 66). O limiar 90000 apresentou como o ponto de máxima eficiência, atingindo a marca de $1,90$ ms (Figura 68). Pode se observar a eficiência no uso de *cache*, em que o custo de é diluído em consultas com maior nível de particionamento, permitindo que um aumento de dez vezes no tamanho da janela resulte em um acréscimo de latência de apenas 110% (Figura 70). Entretanto, existe uma tendência de estabilização entre os Níveis 3 e 4, sugerindo que o incremento do particionamento atinge rendimentos decrescentes após a precisão necessária para a recuperação do ROI ser estabelecida.

O impacto da compressão `zstd`, nos níveis iniciais de particionamento, o custo de decodificação em memória primária frequentemente superior à economia de tempo para recuperação dos dados, tornando o modo sem compressão mais vantajoso. Contudo, em cenários de particionamento alto (nível 4), a compressão proporciona ganhos pontuais na latência final, como observado no ROI de tipo 2, onde o tempo reduziu de $5,51$ ms para $5,14$ ms (Figura 66). Esse comportamento indica que o algoritmo `zstd` atua como uma forma de otimização complementar, onde é necessário presar pelo menor volume de armazenamento em memória secundária.

4.4.2.4 Análise de Consultas de Janela Aleatória

Esta seção apresenta a análise das consultas de janela aleatória, decompondo os resultados em distintos cenários experimentais. São avaliadas as taxas de recuperação entre a memória secundária e a resposta, bem como o desempenho temporal das consultas. Todas as métricas são analisadas em função do limiar, do tamanho da janela e do tipo de ROI. Os resultados detalhados encontram-se no Apêndice (Seção A.9).

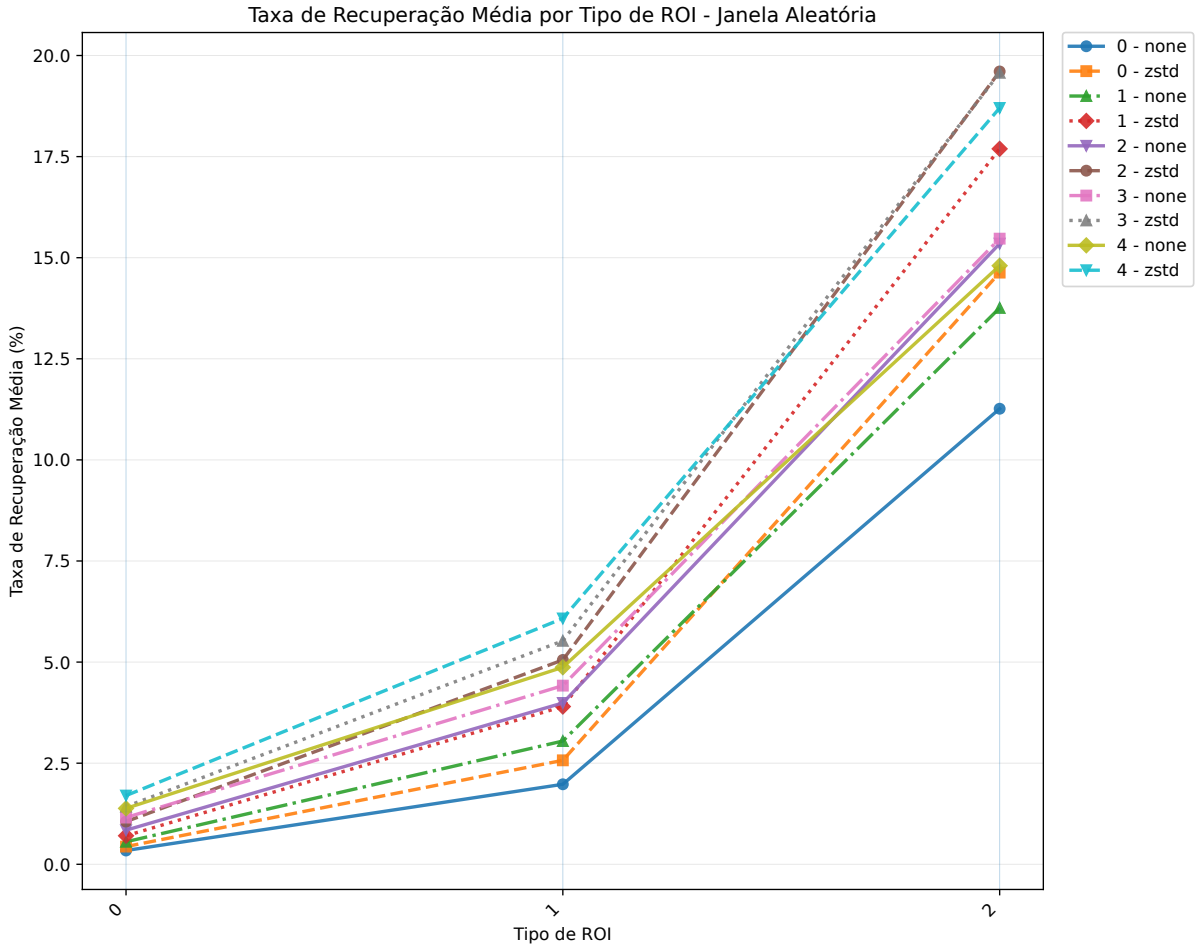

Figura 71 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tipo de ROI, nível de particionamento e tipo de compressão em janela aleatória para o cenário sintético.

A Figura 71 exibe a eficiência de recuperação para consultas de janela aleatória, definida pela razão entre os dados recuperados da memória secundária e o tamanho da resposta. Pode ser observado que a eficiência das consultas é influenciada pelo tipo de ROI e pelo nível de particionamento espaço-temporal. Pode ser verificado que a taxa de aproveitamento de recuperação está relacionada ao tamanho do ROI, com o ROI 0 apresentando uma taxa de apenas $0,34\%$ no nível 0 sem compressão, enquanto o ROI 2 atingiu $11,26\%$ para os mesmos parâmetros. Este comportamento indica que ROIs de maior dimensão favorecem um melhor aproveitamento na recuperação de dados da memória secundária, reduzindo o impacto de dados sem valor semântico para a consulta. Os dados detalhados constam na Tabela 43 do Apêndice A.

Pode ser observado que o impacto do nível de particionamento espaço-temporal e da compressão é condicionado pelo tipo de ROI consultado. No ROI de tipo 0, o ganho da taxa de recuperação é pequena, em que a taxa de recuperação oscilou de $0,34\%$ (nível

0, sem compressão) para apenas 1, 38% (nível 4, sem compressão). Isso mostra que as otimizações têm pouca relevância para ROIs pequenos (tipo 0). Esse impacto torna-se mais perceptível no ROI de tipo 1, no qual o incremento do nível de particionamento e o uso da compressão `zstd` elevaram a taxa de recuperação de 1, 97% para 6, 07%. Por fim, o ROI de tipo 2 apresenta a maior sensibilidade às otimizações, com a taxa de recuperação expandindo de 11, 26% para o patamar de 19, 60% de aproveitamento no nível 2 sob compressão `zstd`. Esses resultados demonstram que o particionamento espaço-temporal e a compressão potencializam a eficiência de recuperação de forma progressiva conforme a dimensão dos ROIs aumentam, permitindo que o sistema reduza a taxa volume de dados recuperados sem valor semântico em ROIs de maior escala.

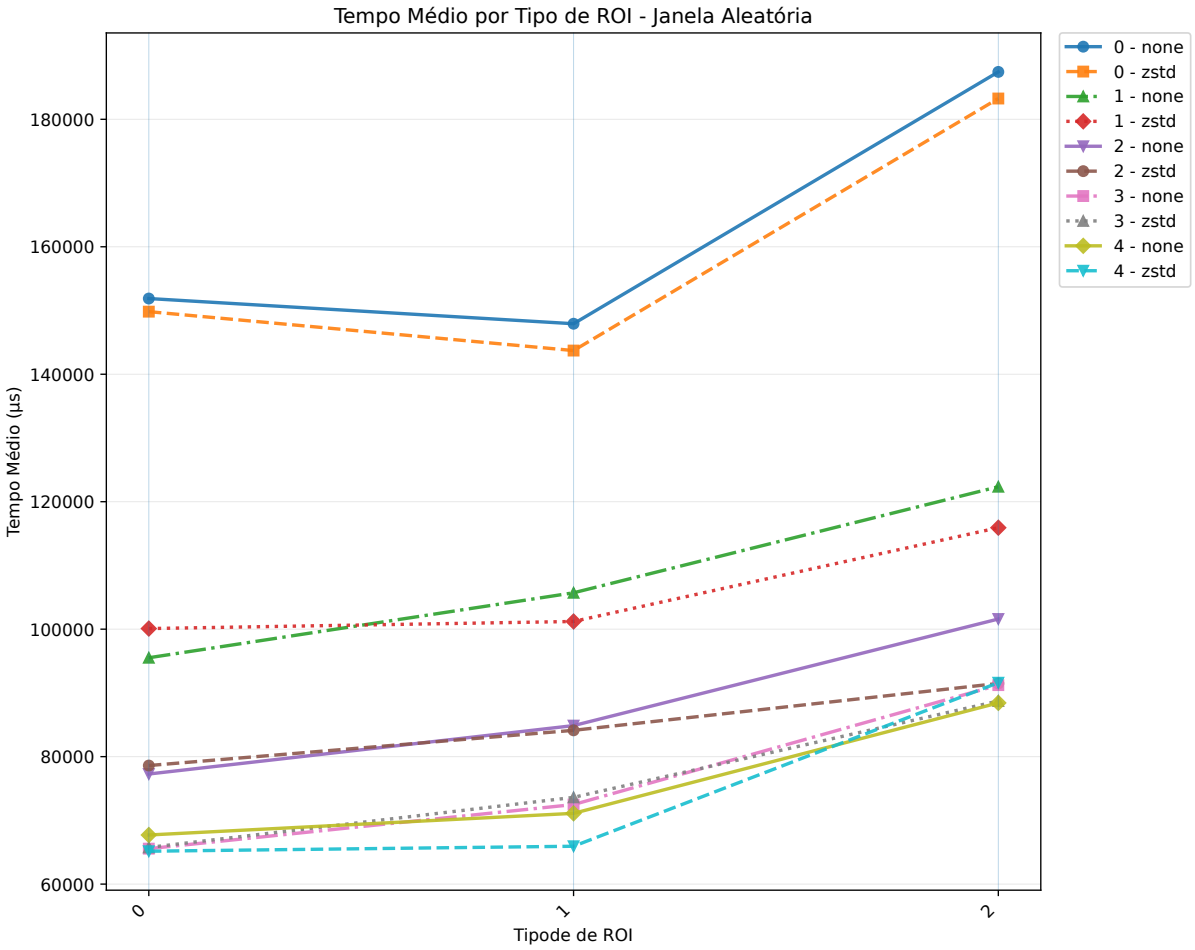

Figura 72 – Média de tempo das consultas de janela aleatória agrupadas por tipo de ROI por nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 72 exibe o tempo médio das consultas por janela aleatória, mostrando a eficácia do particionamento espaço-temporal na redução do tempo das consultas. Pode ser observado que o incremento do nível 0 para o nível 4 promoveu uma redução de latência superior a 50% em todas as categorias de ROI para mesmo conjunto de parâmetros. No

ROI 0 sob compressão `zstd`, o tempo médio reduziu de 149, 82 ms no nível 0 para 65, 15 ms no nível 4, representando uma redução de 56, 5%. Estes resultados demonstram que a taxa de recuperação em particionamento espaço-temporal é eficaz para diferentes dimensões de busca. Os dados detalhados constam na Tabela 44 do Apêndice A.

A aplicação da compressão não apresenta vantagens em todos os casos de consultas. No ROI de tipo 1 (nível 4), o tempo médio reduziu de 71, 10 ms (sem compressão) para 65, 94 ms (`zstd`), indicando que a redução na carga de recuperação de dados na memória secundária compensa o custo de decodificação em memória primária conforme a taxa de recuperação aumenta. Contudo, em níveis intermediários de particionamento, como no ROI 0 (nível 1), a ausência de compressão apresentou melhor desempenho (95, 51 ms contra 100, 09 ms). Esse comportamento reforça que o particionamento espaço-temporal é o principal fator de otimização temporal, enquanto a compressão `zstd` atua como uma melhoria complementar que depende do equilíbrio entre do contexto dos dados e o processamento necessário para a descompressão.

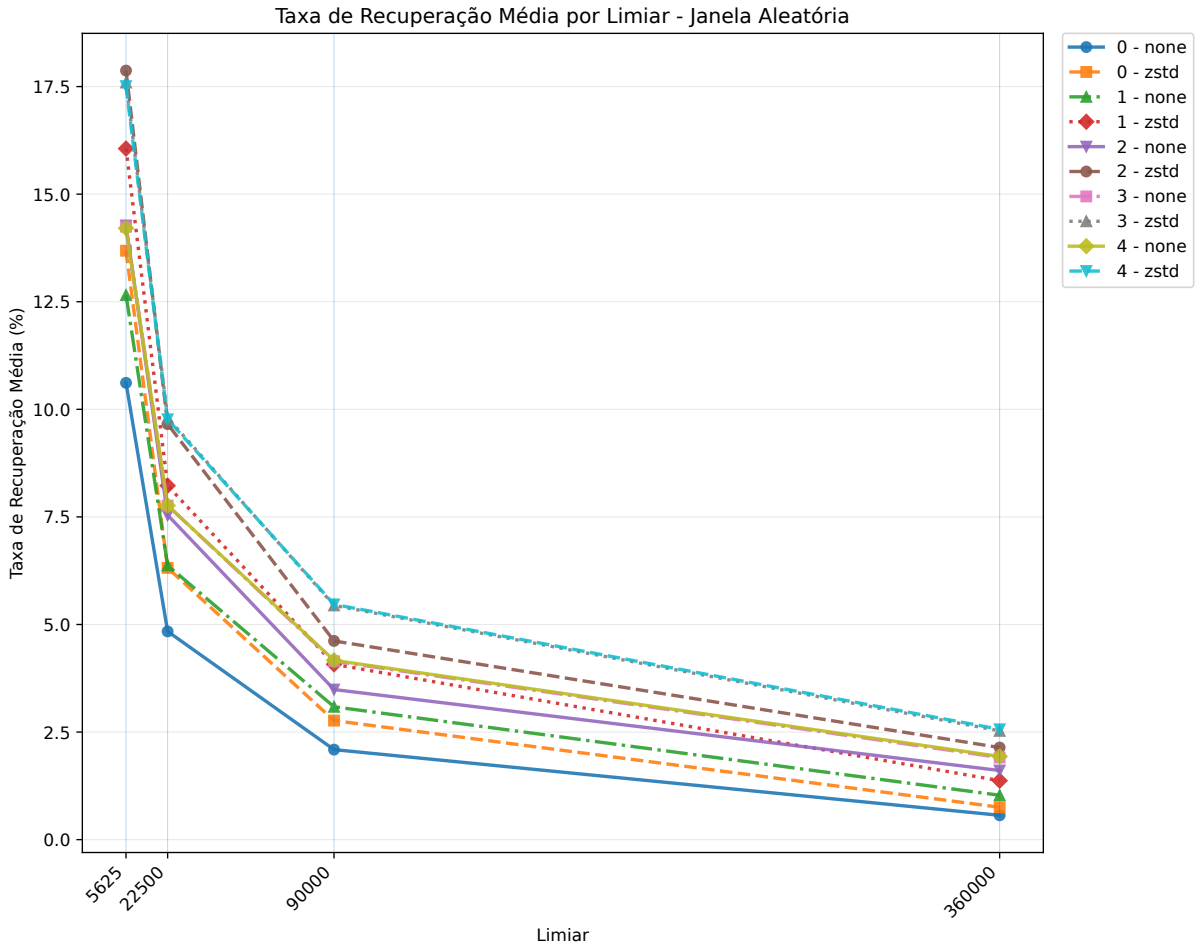

Figura 73 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta de janela aleatória por limiar, nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 73 ilustra a análise da taxa de recuperação média sob diferentes limiares de particionamento, evidenciando que a eficiência inicial do sistema é sensível ao valor do limiar aplicado. Pode ser observado que limiares menores favorecem um melhor aproveitamento de recuperação de dados da memória secundária, reduzindo o volume de dados sem valor semântico para a consulta. No nível 0, o limiar 5625 sem compressão apresentou uma taxa de $10,61\%$, enquanto o limiar 360000 demonstrou uma taxa de recuperação de apenas$0,56\%$. Estes resultados indicam que o uso de limiares menores no nível 0 minimiza a recuperação de dados adjacentes que não pertencem à região de interesse consultada. Os dados detalhados constam na Tabela 46 do Apêndice A.

Pode ser observado que o incremento do nível de particionamento espaço-temporal atua como uma otimização da taxa de recuperação, sendo eficaz em cenários de limiares elevados. No limiar 360000 sob compressão `zstd`, a transição do nível 0 para o nível 4 elevou a taxa de recuperação de $0,75\%$ para $2,56\%$, representando um incremento superior a 240%. Adicionalmente, o emprego da compressão `zstd` manteve melhora em todos os níveis e limiares, atingindo a eficiência máxima de $17,87\%$ no limiar 5625 (nível 2). Assim, a compressão auxilia no aproveitamento de recuperação de dados matriciais, permitindo que cada operação de recuperação de dados em memória secundária contenha uma proporção maior de dados úteis para a resposta da consulta.

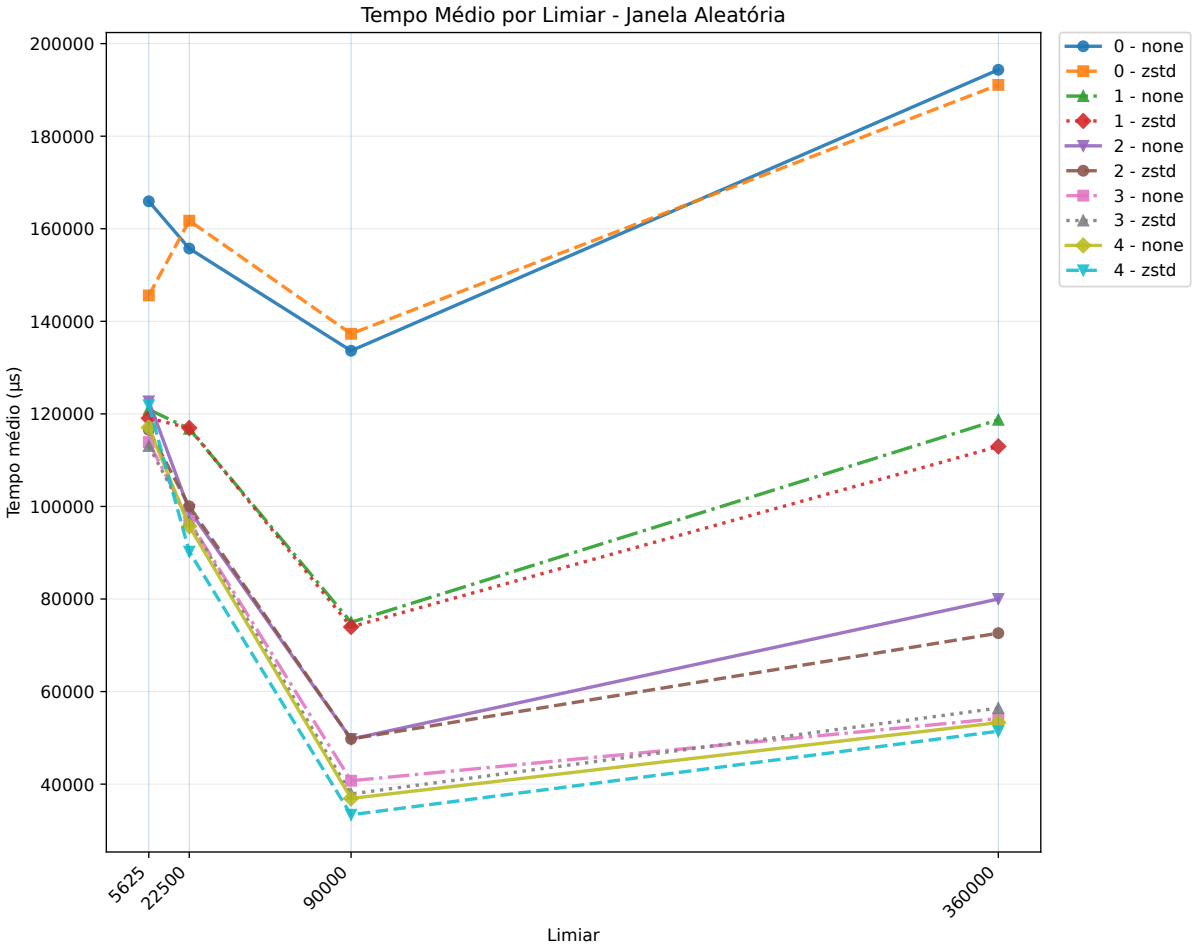

Figura 74 – Média de tempo das consultas de janela aleatória agrupadas por limiar pelo nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 74 exibe o tempo médio das consultas por janela aleatória pelo limiar. É apresentado que o tempo de consulta tem relação entre o limiar e o nível de particionamento espaço-temporal. Pode ser verificado que o limiar de 90000 sob compressão `zstd` representa o ponto de máxima eficiência, atingindo a latência mínima de $33,38$ ms no nível 4. Em contrapartida, limiares reduzidos (5625) apresentaram uma tendência de estabilização precoce no desempenho, evidenciando que o particionamento em blocos muito reduzidos introduz custos de gerência estrutural que limitam os ganhos de performance temporal. Os dados detalhados constam na Tabela 47 do Apêndice A.

O aumento do nível de particionamento espaço-temporal viabiliza a operação com limiares elevados. Para o limiar de 360000 sob compressão `zstd`, a transição do nível 0 para o nível 4 reduziu a latência média em $73,08\%$ (de $191,07$ ms para $51,43$ ms). O menor tempo de resposta dessa configuração nos maiores níveis de particionamento indica que o gargalo do sistema é o I/O. Assim, a redução do volume de dados recuperados devido à compressão supera o custo computacional da descompressão em memória principal,

minimizando o tempo total de consulta.

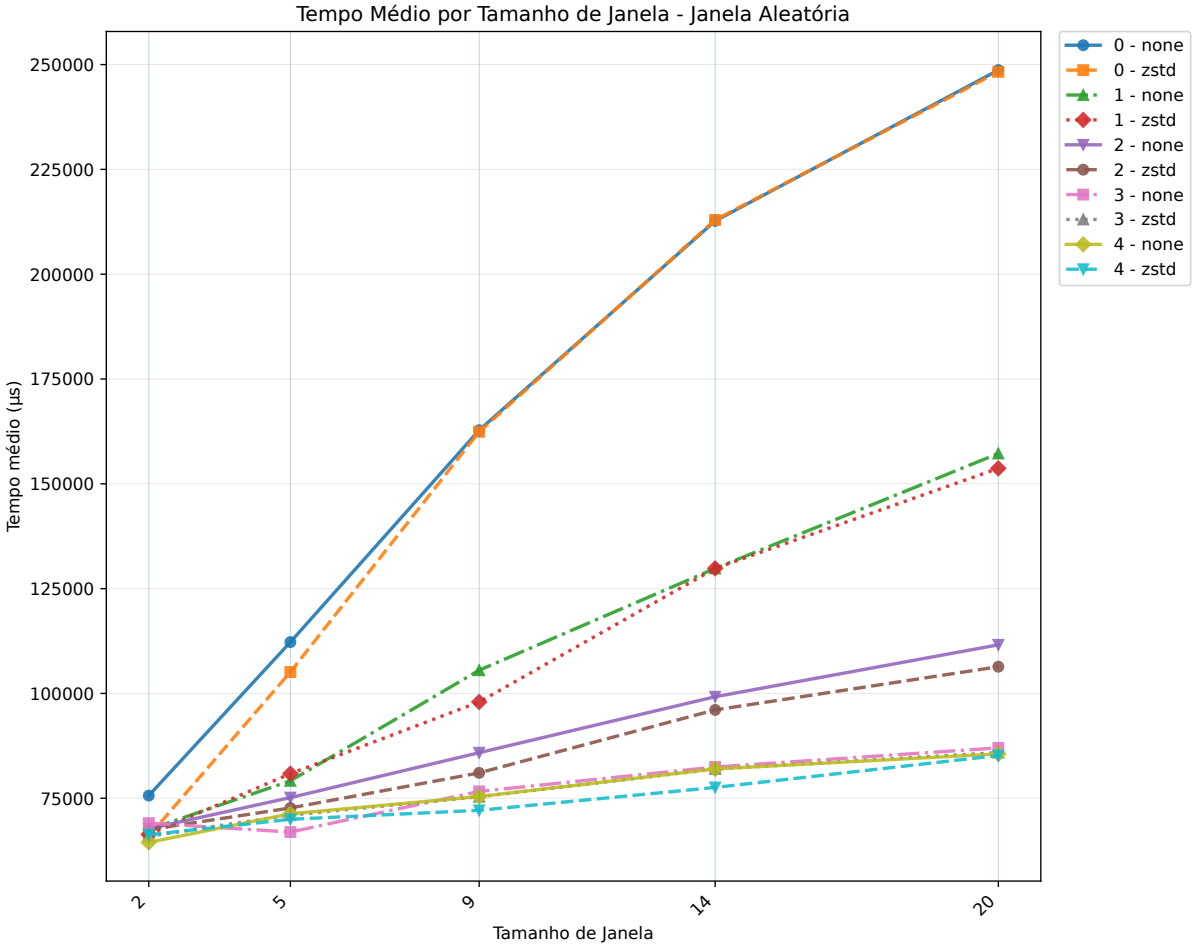

Figura 75 – Média de tempo das consultas de janela aleatória agrupadas por tamanho de janela pelo nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 75 exibe a análise do tempo médio de resposta para consultas de janela aleatória em relação aos tamanhos de janelas, reforçando que o nível de particionamento espaço-temporal é o principal fator de otimização para reduzir o custo de acesso à memória secundária. Pode ser observado que o sistema apresenta um comportamento de escalabilidade temporal eficiente conforme o tamanho da janela aumenta. No nível 4, o incremento de dez vezes no tamanho da janela, resultou em um acréscimo de latência de $32,7\%$, elevando o tempo de $64,45$ ms para $85,57$ ms (sem compressão). Este resultado contrasta com o observado no nível 0, no qual o mesmo aumento na dimensão temporal elevou o custo em aproximadamente $228,7\%$. Os dados detalhados podem ser observados na Tabela 49 do Apêndice A.

O incremento do nível de particionamento espaço-temporal proporciona a estabilidade do desempenho em consultas de maior abrangência temporal. Para janelas extensas (20

instantes temporais), a progressão do nível 0 para o nível 4 promoveu uma redução de $65,6\%$ na latência média, demonstrando a eficácia da taxa de recuperação na redução do volume de dados processados. Além disso, a paridade temporal observada entre as configurações sem compressão e `zstd` em níveis elevados de particionamento confirma que o processo de consulta é limitado pela velocidade de leitura em memória secundária. Nesses cenários, o benefício da redução volumétrica entregue pela compressão compensa o custo computacional da decodificação em memória primária, resultando em latências finais equivalentes ou superiores, mas não se aplica a todas as consultas.

A análise integrada das consultas de janela aleatória demonstra que a eficiência do sistema é a relação entre o tipo de ROI (classe de tamanho), o limiar de particionamento e o nível de particionamento espaço-temporal. É percebido que a taxa de recuperação em memória secundária é diretamente proporcional ao tipo do ROI, em que o incremento do nível de particionamento e o uso da compressão `zstd` otimizam o aproveitamento de recuperação de forma progressiva (Figura 71). Enquanto o ROI de tipo 0 apresentou baixa sensibilidade às otimizações, o ROI de tipo 2 atingiu $19,60\%$ da taxa de recuperação no nível 2. Esse comportamento mostra que a redução do volume de dados recuperados sem valor semântico com maior eficácia em consultas de maior limiar.

Em relação ao desempenho de tempo de consulta, a redução da latência acompanha o ganho da taxa de recuperação em particionamento espaço-temporal, com melhorias superiores a 50% em todas as categorias de ROI ao atingir o nível 4 (Figura 72). A análise baseada no limiar de particionamento inicial (Figura 73 e Figura 74) revelou que, embora limiares menores (5625) ofereçam melhor taxa de recuperação inicial. O aumento do nível de particionamento espaço-temporal de limiares elevados (360000) promove reduções de latência de até $73,08\%$. A configuração de maior eficiência foi estabelecido no limiar 90000 sob compressão `zstd`, atingindo $33,38$ ms, o que demonstra o equilíbrio ideal.

O custo de recuperação de dados é amortizado em níveis elevados de particionamento. Existe um aumento de dez vezes no tamanho da janela resultou em um acréscimo de latência de apenas $32,7\%$ no nível 4, comparado aos $228,7\%$ observados no nível 0. Ademais, o impacto da compressão `zstd` mostra que o processo de consulta é limitado pela recuperação de dados em memória secundária. Em cenários de alta taxa de recuperação, a redução do limar compensa o custo de decodificação em memória primária, tornando a compressão uma otimização complementar, embora seu benefício não seja uniforme em todos os cenários de particionamento intermediário.

4.4.2.5 Análise de Consultas de Instante Temporal

Esta seção descreve as análises de consultas de instante temporal, decompondo os resultados em diferentes cenários experimentais. São avaliadas as taxas de recuperação relacionando a memória secundária à resposta e do desempenho temporal das consultas.

Todas as métricas são analisadas em função do limiar e do tipo de ROI. Os resultados detalhados encontram-se no Apêndice (Seção A.10).

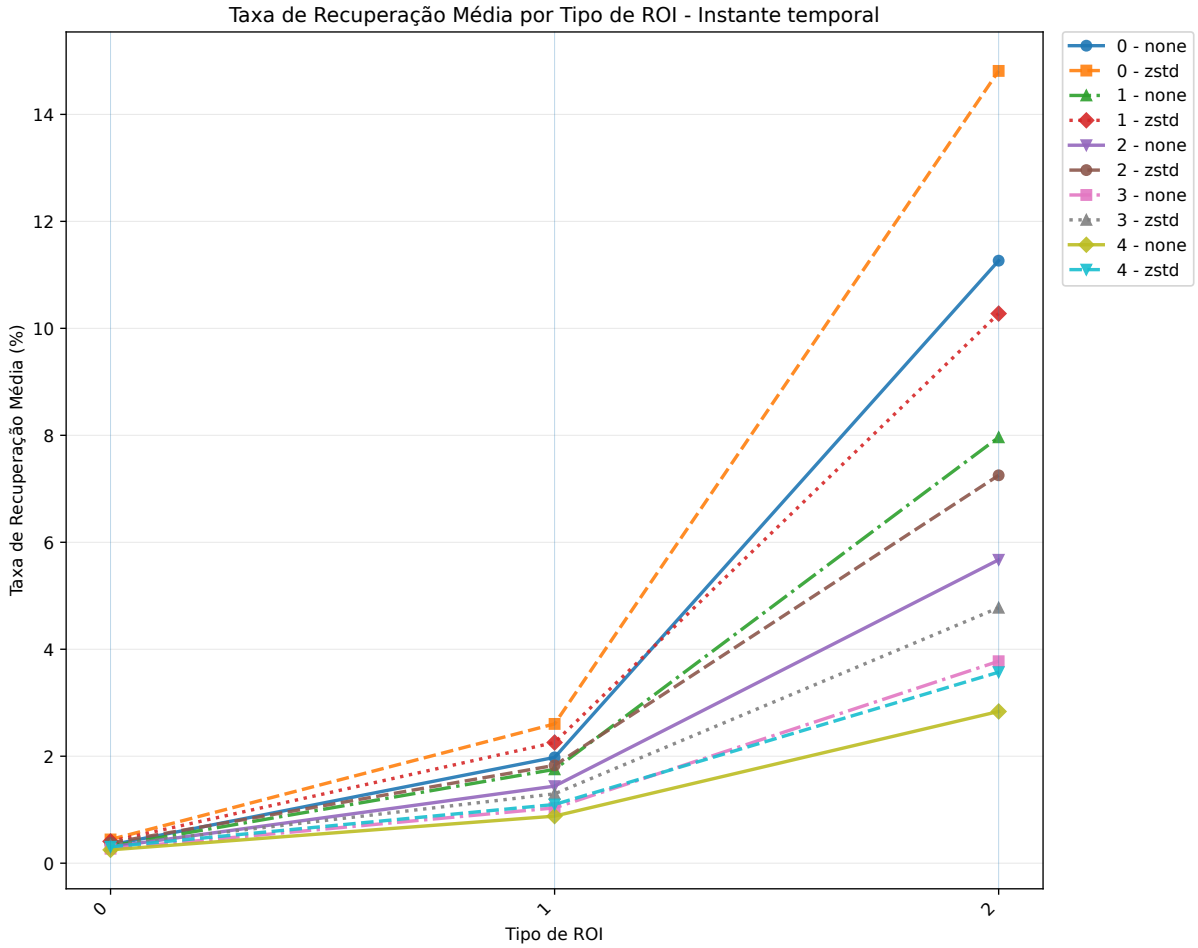

Figura 76 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta por tipo de ROI, nível de particionamento e tipo de compressão em instante temporal para o cenário sintético.

A Figura 76 exibe a taxa de recuperação em memória secundária pelo tamanho da resposta para consultas de instante temporal. A análise revela uma inversão no comportamento de eficiência em função do nível de particionamento. Diferente dos padrões observados em janelas temporais extensas, o aumento do nível de particionamento provoca uma redução progressiva na taxa de aproveitamento, com os ROIs de tipo 2 apresentando um decréscimo de $11,26\%$ no nível 0 para $2,83\%$ no nível sem compressão. Esse resultado mostra que o particionamento espaço-temporal excessiva tende a um *overhead* de recuperação de múltiplos blocos da memória secundária. Os dados detalhados podem ser observados na Tabela 50 do Apêndice A.

O uso de compressão (`zstd`) que reduz a perda da taxa de recuperação, elevando a densidade de informação recuperada em aproximadamente $31,5\%$ nos ROIs de tipo 2

no nível 0 se comprado com blocos sem compressão (14, 81% contra 11, 26%). O uso do `zstd` manteve a superioridade em todos os resultados sem compressão, compensando o impacto do volume de dados adjacentes recuperados sem valor semântico para a consulta. Os resultados mostram que consultas de instante temporal, com configurações de níveis de particionamento reduzidos (níveis ≤ 1) contribui para um acesso mais eficiente à memória secundária, maximizando a razão entre dados úteis e recuperação de dados matriciais.

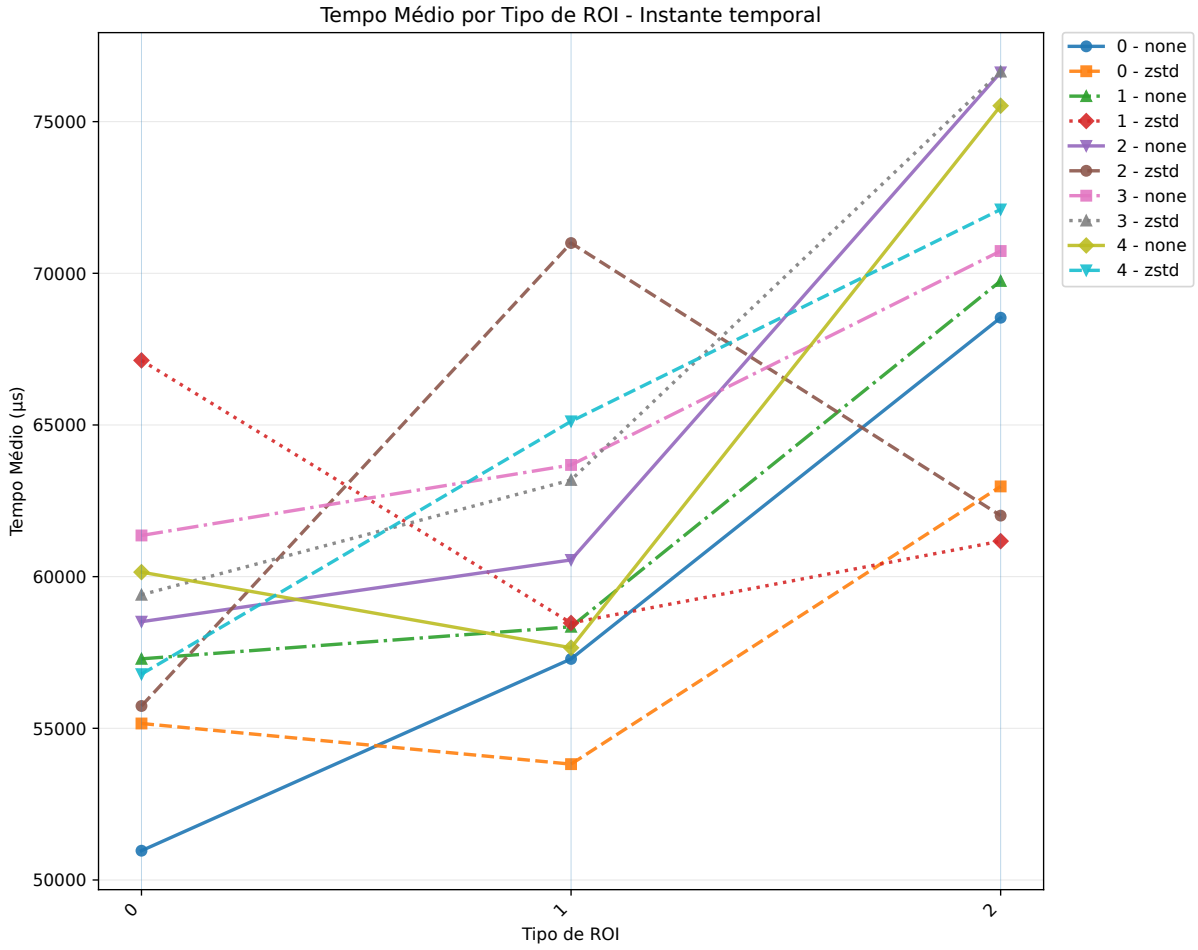

Figura 77 – Média de tempo das consultas de instante temporal agrupadas por tipo de ROI por nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 77 exibe a avaliação do desempenho temporal para consultas de instante temporal. A análise demonstra que a latência média tende a aumentar com o incremento do nível de particionamento, embora este não tenha sido um comportamento unânime. Existe uma degradação geral de desempenho entre o nível 0 e o nível 4 em todas as categorias de ROI, motivada pela necessidade de processar múltiplos blocos espaço-temporais. A recuperação de dados da memória secundária permanece como o principal gargalo de desempenho do sistema. Nos ROIs de tipo 2 sem compressão, o tempo médio aumentou de 68, 54 ms para 75, 52 ms, representando um aumento de 10, 19% na latência. Nos ROIs de

tipo 0 sob a mesma configuração, a latência subiu de $50,96$ ms para $60,15$ ms ($18,02\%$), enquanto os ROIs de tipo 1 com compressão `zstd` registraram o maior aumento, de $53,82$ ms para $65,12$ ms (21%). Os dados detalhados podem ser observados na Tabela 51 do Apêndice A.

Embora a tendência global indique aumento da latência conforme a escala do nível de particionamento avança, existem reduções pontuais em transições específicas. Nos ROIs tipo 0 sob compressão `zstd`, é apresentada uma redução de $16,97\%$ ao variar do nível 1 ($67,13$ ms) para o nível 2 ($55,74$ ms), além de um decréscimo de $4,41\%$ entre o nível 3 ($59,40$ ms) e o nível 4 ($56,79$ ms). Nos ROIs de tipo 1 sem compressão, a redução foi de $9,46\%$ na transição do nível 3 ($63,68$ ms) para o nível 4 ($57,66$ ms). Já os ROIs de tipo 2 comprimidos apresentaram uma queda de $2,86\%$ do nível 0 ($62,97$ ms) para o nível 1 ($61,17$ ms) e de $5,93\%$ entre o nível 3 ($76,65$ ms) e o nível 4 ($72,10$ ms). Estes dados sugerem que, mesmo em consultas de instante temporal, o refinamento do particionamento espacial pode mitigar o tempo de resposta em cenários específicos, embora o ponto de operação ideal permaneça, normalmente, nos níveis de menor profundidade.

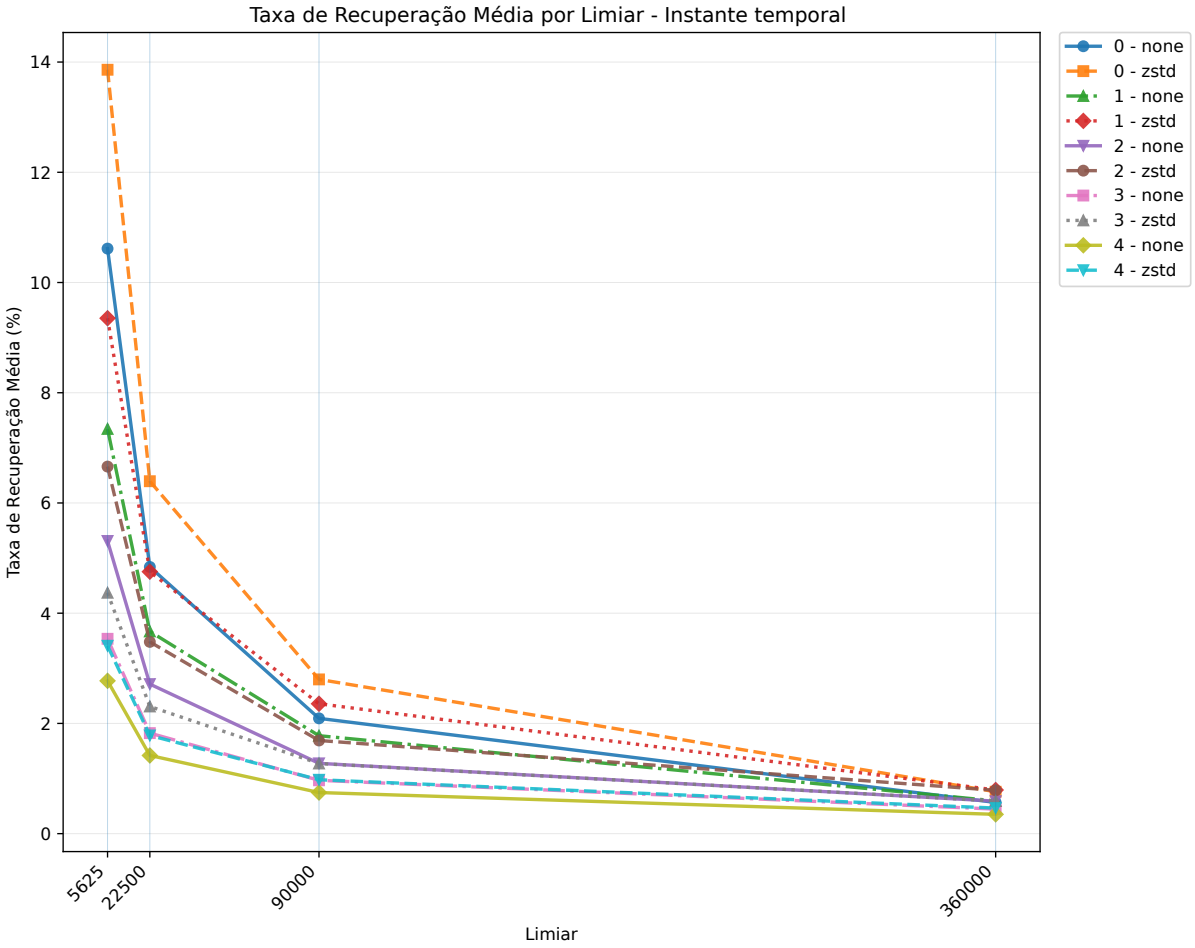

Figura 78 – Taxa média de recuperação do dado da memória secundária em relação à resposta da consulta de instante temporal por limiar, nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 78 apresenta a avaliação da eficiência de recuperação em memória secundária em relação ao tamanho da resposta para consultas de instante temporal, revelando que a taxa de recuperação é derivada da relação entre o limiar inicial e o nível de particionamento. Os limiares reduzidos (5625) proporcionam uma taxa de aproveitamento superior no nível 0, atingindo $10,61\%$ sem compressão, enquanto limiares elevados (360000) iniciam com apenas $0,56\%$. O incremento do nível de particionamento provoca uma degradação da taxa de recuperação na maioria dos cenários, visto que o particionamento excessivo para o instante temporal eleva a complexidade e pode aumentar a quantidade de blocos recuperados da memória secundária. No limiar de 360000, uma inversão pontual de tendência entre o nível 0 e o nível 1, onde a taxa subiu de $0,56\%$ para $0,59\%$, representando um ganho irrelevante da taxa de recuperação de $4,95\%$ antes do declínio subsequente nos níveis maiores. Os dados detalhados podem ser observados na Tabela 53 do Apêndice A.

A redução de eficiência entre o nível 0 e o nível 4 demonstra que os limiares menores

sofrem as perdas mais acentuadas devido ao alto nívep de particionamento. No limiar de 5625 sem compressão, a eficiência regrediu de $10,61\%$ para $2,77\%$, o que representa uma redução de $73,88\%$, enquanto na configuração `zstd` a queda foi de $75,40\%$ (de $13,86\%$ para $3,41\%$). No limiar de 22500 sem compressão tem um declínio de $4,84\%$ para $1,42\%$, com uma redução de 70,68%. O limiar de 90000 sem compressão a taxa reduziu de $2,09\%$ para $0,75\%$, uma perda de $64,34\%$ na taxa de recuperação. Por fim, o limiar de 360000 (sem compressão) apresentou a menor redução no intervalo, reduzindo de $0,57\%$ para $0,35\%$, redução de $38,16\%$. Estes resultados demostram a recuperação de instantes temporais, configurações com menor profundidade de particionamento e limiares reduzidos maximizam a taxa de recuperação dos dados.

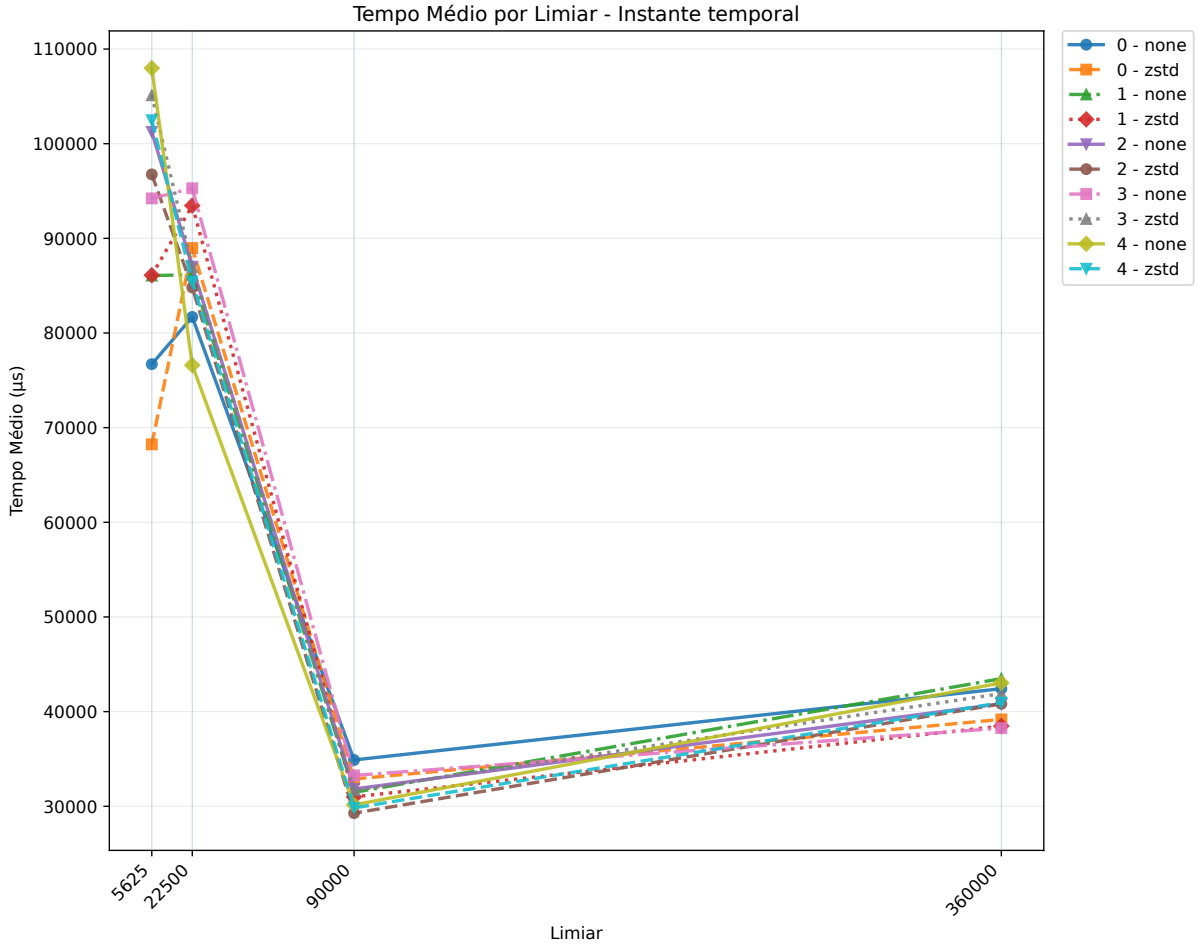

Figura 79 – Média de tempo das consultas de instante temporal agrupadas por limiar pelo nível de particionamento e tipo de compressão para o cenário sintético.

A Figura 79 exibe a análise do tempo médio para consultas de instante temporal, que mostra o limiar de 90000 como o ponto de operação de maior eficiência, atingindo a latência mínima de $29,27$ ms no nível 2 sob compressão `zstd`. Os limiares intermediários e elevados apresentam tempos de resposta significativamente inferiores ao limiar reduzido de

5625, o qual mostra uma maior intervalo de variação, devido à necessidade de gerenciar mais blocos. Embora a tendência em consultas de instantes temperais seja o aumento do tempo conforme aumente o nível de particionamento, não se aplicou a todas as configurações testadas. Os dados detalhados podem ser observados na Tabela 54 do Apêndice A.

O limiar de 22500 sem compressão registrou a redução mais acentuada do nível 3 (95, 29 ms) para o nível 4 (76, 60 ms), resultando em um ganho de performance de 19, 61%. No limiar de 90000, as reduções para os dados sem compressão entre o nível 0 (34, 89 ms) e o nível 1 (31, 47 ms), com queda de 9, 81%. Os dados comprimidos com `zstd` entre o nível 1 (30, 99 ms) e o nível 2 (29, 27 ms), apresentam uma redução de 5, 55%. Ademais, o limiar de 5625 sob a configuração sem compressão apresentou uma redução de 6, 94% entre o nível 2 (101, 25 ms) e o nível 3 (94, 22 ms), enquanto o limiar de 360000 obteve uma queda de 6, 33% na transição entre o nível 2 (40, 84 ms) e o nível 3 (38, 25 ms). Estes resultados demonstram que o ajuste do nível de particionamento pode mitigar a latência mesmo em regimes de instante temporal, desde que a distribuição espacial otimize o acesso aos blocos físicos na memória secundária.

A análise integrada das consultas de instante temporal demonstra que, diferentemente do observado nas janelas aleatória e deslizantes, a eficiência do sistema é inversamente proporcional ao incremento do nível de particionamento. A taxa de recuperação se mostrou o ponto de maior aproveitamento ocorre em níveis reduzidos de particionamento (nível 0), onde os ROIs do tipo 2 atingiram 11, 26% de eficiência (Figura 76). A progressão para o nível 4 reduziu essa taxa para 2, 83% sem compressão, indicando que a dispersão dos dados em múltiplos blocos de dados prejudica a razão entre dados úteis e o volume total recuperado. O emprego da compressão `zstd` otimizou a taxa de recuperação, aumentando em até 31, 50% nos ROIs de tipo 2 no nível 0, embora não tenha revertido a tendência de queda imposta pelo aumento do nível de particionamento.

A latência média teve degradação com a tendência de aumento com a progressão do nível de particionamento (Figura 77). Nos ROIs de tipo 2 sem compressão, o tempo médio aumentou de 68, 54 ms para 75, 52 ms. Entretanto, a análise baseada no limiar de particionamento revelou que o limiar de 90000 sob compressão `zstd` estabeleceu o ponto de operação ideal, atingindo a latência mínima de 29, 27 ms no nível 2 (Figura 79). Esse comportamento indica que, para consultas de instante temporal, o equilíbrio entre o nível de particionamento e o limiar determina a minimização do custo de acesso à memória secundária.

Os resultados demonstram que, para o regime de instante temporal, configurações com menor profundidade de particionamento (Níveis ≤ 1) e limiares intermediários (9000) proporcionam o melhor balanço entre a taxa de recuperação e o tempo de resposta. Embora o particionamento espacial possa oferecer reduções pontuais de latência em transições específicas de nível, o custo de gerência de blocos com particionados excessivos sobrepuja

os benefícios da taxa de recuperação em particionamento espacial na maioria dos cenários de consulta de instante temporal.

4.4.2.6 Discussões do Cenário Sintético

A análise consolidada dos resultados experimentais no cenário sintético permite identificar as principais relações entre eficiência de armazenamento, custo de ingestão e desempenho de recuperação para o Dympas. A aplicação do algoritmo `zstd` promove uma economia consistente de $23,77\%$ na ocupação de memória secundária, conforme ilustrado na Figura 58, sem que o nível de particionamento ou o limiar exerçam influência significativa na persistência física dos dados. Contudo, essa eficiência de armazenamento impõe um *overhead* computacional severo durante a ingestão, resultando em tempos de inserção até $69,07\%$ superiores aos observados em dados sem compressão (Figura 59). Este cenário configura um regime limitado por processamento durante a persistência dos dados matriciais, em que a redução do I/O é superada pelo custo de compactação, enquanto a operação de particionamento espaço-temporal (*split*) apresenta latências sensíveis tanto à compressão quanto ao nível, atingindo seu custo máximo na transição para o nível 4 sob o limiar de 360000 (Figura 60).

O particionamento espaço-temporal demonstrou, para os tempos de resposta de consultas, e como o principal meio de otimização, superando o impacto do método de compressão dos blocos de forma isolada. A transição do nível 0 para o nível 4 proporcionou uma redução global de latência de 73% na análise unificada (Figura 61), evidenciando que a taxa de recuperação resultante do particionamento é determinante para minimizar o acesso a dados irrelevantes. Este resultado é maximizado no regime de janela deslizante, onde o reaproveitamento de *cache* associado à alta da taxa de recuperação do nível 4 permitiu atingir latências médias mínimas de $1,90$ ms (Figura 66), representando um ganho de eficiência superior a 92% em relação ao estado inicial da estrutura (nível 0). Entretanto, as consultas de janela aleatória (Figura 72) e de instante temporal (Figura 77) demonstraram maior dependência do limar de dados, com a compressão `zstd` apresentando uma otimização complementar apenas em cenários de alta da taxa de recuperação, onde a redução do volume de blocos recuperados da memória secundária compensa o custo de decodificação.

Outro ponto de relevância da análise refere-se à inversão de comportamento observada nas consultas de instante temporal em comparação aos regimes de janela. Enquanto para as janelas (deslizantes ou aleatórias) o incremento do nível de particionamento eleva invariavelmente a taxa de recuperação para patamares superiores a 93% (Figura 67), o regime de instante temporal apresenta uma degradação da taxa de recuperação conforme aumenta o nível de particionamento espaço-temporal, reduzindo a eficiência de $11,26\%$ para $2,83\%$ nos ROIs de tipo 2 (Figura 76). A particionamento excessivo para consultas

de instante temporal gera um *overhead* de gerência de múltiplos blocos espaço-temporais que degrada os ganhos de taxa de recuperação espacial. Assim, enquanto o nível 4 é o ponto de operação ideal para séries temporais (conforme a configuração), os regimes de consulta de instante temporal tem seu equilíbrio em níveis intermediários (nível 2), onde a latência é minimizada pela redução da complexidade estrutural (Figura 79).

O limiar de 90000 é a configuração de maior versatilidade para o sistema Dympas no cenário sintético. Ele tem o equilíbrio entre a agilidade de ingestão com a escalabilidade das consultas em todos os regimes avaliados. A estabilidade demonstrada frente ao aumento das janelas temporais, em que um incremento de dez vezes na dimensão da consulta resultou em acréscimos de latência proporcionalmente reduzidos no nível 4 (Figura 70 e Figura 75), valida a robustez da arquitetura para grandes volumes de dados matriciais. Portanto, a estratégia de particionamento espaço-temporal em múltiplos níveis não apenas reduz o custo de acesso à memória secundária, mas também contribui para a previsibilidade do desempenho do sistema, desde que a profundidade da estrutura seja calibrada conforme a taxa de recuperação temporal exigida pela aplicação.

5 CONCLUSÃO

Esta dissertação apresentou o Dympas (**Dy**namic **M**atrix **P**artitioning **S**ystem), uma abordagem para o particionamento dinâmico de séries de dados matriciais proposta para suprir a rigidez identificada nas soluções de armazenamento e consulta na literatura. A motivação central do trabalho baseou-se na análise das limitações do estado da arte, que opera predominantemente de forma estática e exclusiva, otimizando ou instantes temporais isolados, ou intervalos espaciais fixas (hipercubo). Estas abordagens impõem restrições ao desempenho de consultas que exigem alta granularidade espacial em Regiões de Interesse (ROIs) e uma análise profunda ao longo de séries temporais.

A metodologia proposta introduziu operações coordenadas de *split* (particionamento espaço-temporal) e *unsplit* (operação inversa do *split*). Diferente dos modelos convencionais, o Dympas permite particionar os dados no espaço e, de forma compensatória, reagrupá-los no eixo temporal, assegurando que a informação seja organizada em blocos espaço-temporais de dimensionalidade equivalente. Este procedimento mantém o volume de dados por bloco constante enquanto a geometria do particionamento é ajustada à demanda de operações específicas. Além disso, as operações válidas do Dympas podem atuar em uma área espaço-temporal pontual, sem afetar dados adjacentes, realizando otimizações precisas. A infraestrutura de consulta é sustentada por uma hierarquia de índices composta por uma B^+-tree para o domínio temporal, KD-tree para o domínio espacial de cada instante e um Hash Index *in loco* para a localização temporal interna aos blocos, visando minimizar o acesso redundante à memória secundária.

Para alcançar os objetivos pretendidos, foi realizada uma análise quantitativa comparando o nível 0 (particionamento exclusivamente espacial), que reflete o particionamento tradicional, com níveis superiores de particionamento espaço-temporal. O Dympas demonstrou eficácia nos particionamentos espaço-temporais, provendo uma infraestrutura dinâmica capaz de se adaptar à disposição dos dados em memória secundária para otimizar as demandas de acesso, visando consultas mais eficientes em termos de latência. Os resultados experimentais validaram a abordagem com reduções no tempo de consulta de até 81% em cenários reais e 87% em cenários sintéticos para uma mesma configuração, atingindo 92% de redução global em testes específicos. Além disso, a taxa de recuperação converge para patamares superiores a 93% no nível 4 devido à estratégia de uso de *cache*, reduzindo significativamente a recuperação de dados da memória secundária sem valor semântico para a resposta. Assim, a pesquisa cumpriu o propósito de estabelecer uma ponte técnica eficiente para a análise de grandes séries matriciais, facilitando o monitoramento de séries temporais com alto desempenho.

As contribuições técnicas do Dympas são apresentadas na entrega de uma aná-

lise quantitativa, a qual demonstrou a viabilidade e a superioridade da abordagem de particionamento espaço-temporal dinâmico sobre os modelos estáticos convencionais. A principal contribuição reside na superação das limitações de desempenho impostas pelo particionamento de nível 0, demonstrando que a reorganização lógica dos dados em blocos de dimensionalidade equivalente constitui o fator relevante para a escalabilidade de sistemas de bancos de dados matriciais. Ao manter o volume de dados por bloco constante enquanto a geometria do particionamento se ajusta à demanda, o método proposto assegura uma gestão eficiente da memória secundária e otimiza a recuperação de informações em cenários de consultas espaço-temporais sem degradação acentuada de consultas de instante temporal.

A referida melhoria de latência é sustentada pela taxa de recuperação da metodologia, que demonstrou a capacidade de reduzir dados irrelevantes para o processo de consulta. Enquanto o modelo de particionamento tradicional (nível 0) apresentou taxas de recuperação útil de apenas $57,98\%$ para determinadas regiões de interesse, a progressão para o nível 4 elevou essa eficiência para patamares superiores a 94%, demonstrou que o refinamento estrutural proposto reduz a recuperação de dados adjacentes sem valor semântico.

Especificamente em consultas de janela deslizante, aplicáveis ao monitoramento agrícola e ambiental, a latência de consulta foi reduzida em até $87,2\%$ no cenário sintético. A estrutura proposta potencializa o reaproveitamento de dados em memória primária através do uso de *cache*, permitindo que o custo de acesso à memória secundária seja diluído à medida que a profundidade da série temporal aumenta conforme o contexto da consulta. A robustez do sistema foi demonstrada por sua estabilidade temporal. No nível 4, um aumento de dez vezes na dimensão da janela de consulta resultou em um acréscimo de latência de apenas $32,7\%$, em contraste com a elevação de $228,7\%$ observada no particionamento exclusivamente espacial.

Uma contribuição técnica relevante desta pesquisa foi a identificação de que a eficiência do sistema além de um particionamento espaço-temporal precisa de uma calibração estratégica baseada na carga de trabalho. Os dados demonstraram que, enquanto consultas de janela deslizante escalam de forma quase linear até o nível máximo de particionamento avaliado, regimes de janela aleatória e instante temporal encontram seu ponto de equilíbrio e eficiência ótima em níveis intermediários, como os Níveis 1 e 2. Essa distinção é necessária para evitar o *overhead* de gerenciamento de múltiplos particionamentos em consultas de instante temporal, onde o particionamento excessivo pode degradar o desempenho global.

Pode ser destacado o uso estratégico de algoritmos de compressão integrados ao particionamento, com a adoção do algoritmo `zstd` proporcionando uma economia média de $40,07\%$ no consumo de memória secundária. Embora a compactação aumente o tempo de ingestão inicial dos dados matriciais, os resultados mostram que esse custo computacional

é compensado durante a recuperação de séries temporais extensas, onde a redução no volume de dados recuperados da memória secundária é um fator para a performance. Por fim, a manutenção da lógica de particionamento representa menos de 7% do tempo total de persistência dos dados matriciais, validando o Dympas como uma camada de otimização viável para o processamento de grandes volumes de dados matriciais.

A contribuição do Dympas vai além do domínio estritamente acadêmico, ele pode ser visto como um facilitador para demandas da indústria. A análise dos resultados demonstrou que a elevada eficiência na recuperação de dados permite que o sistema de gerência de banco de dados evolua de um repositório passivo para um componente ativo em sistemas de apoio à decisão em larga escala. No contexto do monitoramento geoespacial, como o provido pelo programa Sentinel, a redução de latência em consultas de janela deslizante reflete-se em uma capacidade de resposta analítica superior. Para a gestão ambiental, tal desempenho possibilita que análises históricas sobre desmatamento ilegal ou a evolução de focos de queimadas sejam concluídas em frações do tempo anteriormente exigido, viabilizando a formulação de políticas públicas baseadas em evidências processadas de modo eficiente.

Embora o Dympas tenha demonstrado avanços na gestão de séries temporais matriciais, a complexidade presente nos dados geoespaciais e astronômicos abre perspectivas para evoluções que visam ampliar a autonomia e a precisão do sistema. Os trabalhos futuros visam a transição da estrutura atual para um ambiente auto gerenciável, com mecanismos capazes de processar fenômenos físicos de elevada dinamicidade. O objetivo é converter o modelo de particionamento reativo em uma infraestrutura proativa, capaz de antecipar demandas e adaptar a padrões de acesso variáveis sem a necessidade de intervenção manual direta.

No estágio atual, as operações de particionamento no Dympas são disparadas por consultas manuais conforme a demanda analítica dos usuários. Uma evolução natural do sistema reside na automação integral dessas tarefas de manutenção estrutural por meio de particionamento preditivo e escalonamento inteligente. Por isso, é pretendido o desenvolvimento de um motor de agendamento automático apto a identificar tendências de acesso e disparar operações de *split* e *unsplit* de forma proativa baseado em estatísticas. Para assegurar que a reorganização física não comprometa o desempenho em períodos de alta exigência, o sistema deve ser capacitado a realizar o particionamento em horários de baixa carga, baseado em dados estatísticos. Essa abordagem de computação autônoma permitiria ajustes transparentes na infraestrutura, otimizando o equilíbrio entre o custo computacional da reestruturação e a latência na entrega de respostas de consultas.

Um dos desafios para as etapas subsequentes da pesquisa é a adaptação do método para o tratamento de regiões de interesse que variam temporalmente. Em domínios como a astronomia, os objetos mapeados não ocupam coordenadas estáticas devido às posições

relativas dos astros e ao movimento próprio dos corpos celestes. Nesse sentido, um trabalho futuro consiste na implementação de trajetórias de particionamento. Em substituição ao modelo de volume estático que atravessa a série temporal, o Dympas poderá gerar blocos cujas coordenadas espaciais evoluem cronologicamente. Essa capacidade de rastrear o dado de interesse através das camadas espaço-temporais permitiria uma taxa de recuperação ainda superior para objetos em movimento, reduzindo o volume de dados adjacentes recuperados e potencializando a descoberta científica em *datasets* astronômicos e de satélites de órbita baixa.

Adicionalmente, a integração direta com *frameworks* de aprendizado de máquina possibilitaria que os blocos particionados pelo Dympas servissem como entrada direta para *pipelines* de treinamento. Nesse cenário, o sistema identificaria automaticamente as ROIs mais relevantes para o aprendizado de padrões, consolidando o Dympas como uma infraestrutura de dados inteligente voltada à ciência de dados geoespaciais e à análise de grandes volumes de dados matriciais.

TRABALHOS PUBLICADOS PELO AUTOR

Trabalhos publicados pelo autor durante o programa.

Publicações principais do trabalho.

O trabalho com título "**Particionamento Dinâmico Espaço-Temporal para Séries de Dados Matriciais**" recebeu "**menção honrosa**" de "*Best papers*" (artigos completos) no 40º Simpósio Brasileiro de Banco de Dados (SBBD), realizado em Fortaleza-CE no período de 29 de setembro a 02 de outubro de 2025.

1. Geovani Pereira dos Santos, Daniel dos Santos Kaster, **Particionamento Dinâmico Espaço-Temporal para Séries de Dados Matriciais**, 40º Simpósio Brasileiro de Banco de Dados (SBBD 2025), 2025, Sociedade Brasileira de Computação - SBC. (Qualis Eventos 2025, A4)

Publicações complementares.

O trabalho com título "**Avaliação Comparativa de Estratégias de Particionamento para Dados Raster em Bancos de Dados Multidimensionais**" ganhou o prêmio "***Best Paper* - Categoria Pesquisa**" dentre os artigos apresentados na XX Escola Regional de Banco de Dados (ERBD), evento que ocorreu nos dias 23, 24 e 25 de abril de 2025 promovido pela Sociedade Brasileira de Computação (SBC) e organizado pela Universidade Federal de Santa Catarina (UFSC).

1. Marco Túlio Alves de Barros, Geovani Pereira dos Santos, Daniel dos Santos Kaster, **Avaliação Comparativa de Estratégias de Particionamento para Dados Raster em Bancos de Dados Multidimensionais**, Anais da XX Escola Regional de Banco de Dados (ERBD 2025), 04/2025, Sociedade Brasileira de Computação - SBC, p. 1-10, DOI: 10.5753/erbd.2025.7340.

2. Ester de Carvalho Pereira, Geovani Pereira dos Santos, Daniel dos Santos Kaster, Gabriela Cristina Salgado, Cynthia Junqueira, Adilson Chinatto, Ana Claudia dos Santos Luciano, **COMPARAÇÃO DE ÍNDICES DE VEGETAÇÃO DERIVADOS DO SENTINEL-2 PARA ESTIMATIVA DE PRODUTIVIDADE DE SOJA**, Anais do XXI Simpósio Brasileiro de Sensoriamento Remoto, 04/2025, Galoá, Acesso em: 16 jul. 2025, ISBN: 978-65-80968-29-9. (Qualis 2017, B3)

3. Ester de Carvalho Pereira, Geovani Pereira dos Santos, Michel Eduardo Dias Chaves, Gabriela Cristina Salgado, Raúl Roberto Poppiel, Daniel dos Santos Kaster, Nícolas

Augusto Rosin, José Alexandre Melo Demattê, Ana Claudia dos Santos Luciano, **Soybean yield estimation in the Brazilian Midwest using Sentinel-2 imagery**, *Big Earth Data*, 2026, v. 1, p. 1–25, DOI: 10.1080/20964471.2026.2631900. (Qualis Periódicos 2021-2024, A2)

REFERÊNCIAS

[1] ZHAO, Q. et al. An overview of the applications of earth observation satellite data: Impacts and future trends. *Remote Sensing*, v. 14, n. 8, 2022. ISSN 2072-4292. Disponível em: <https://www.mdpi.com/2072-4292/14/8/1863>.

[2] LUO, T. yun et al. High stability of opto-mechanical structure design and validation of Beijing No.3 camera. In: JIANG, Y. et al. (Ed.). *AOPC 2023: Optical Sensing, Imaging, and Display Technology and Applications; and Biomedical Optics*. SPIE, 2023. v. 12963, p. 129630V. Disponível em: <https://doi.org/10.1117/12.3007544>.

[3] SIMONGINI, A. et al. Core-collapse supernova parameter estimation with the upcoming vera c. rubin observatory. *Astronomy & Astrophysics*, EDP Sciences, jun. 2025. ISSN 1432-0746. Disponível em: <http://dx.doi.org/10.1051/0004-6361/202554740>.

[4] GONçALVES, R. et al. Relationship between the spectral response of sugar cane, based on avhrr/noaa satellite images, and the climate condition, in the state of sao paulo (brazil), from 2001 to 2008. In: *International Workshop on the Analysis of Multi-temporal Remote Sensing images (MultiTemp)*. [S.l.: s.n.], 2009. v. 5, p. 315–322.

[5] GONCALVES, R. R. do V. et al. Agricultural monitoring using clustering techniques on satellite image time series of low spatial resolution. In: *2017 9th International Workshop on the Analysis of Multitemporal Remote Sensing Images (MultiTemp)*. IEEE, 2017. p. 1–4. ISBN 978-1-5386-3327-4. Disponível em: <https://ieeexplore.ieee.org/document/8035234/>.

[6] KHAKI, S.; PHAM, H.; WANG, L. Simultaneous corn and soybean yield prediction from remote sensing data using deep transfer learning. *Scientific Reports*, Nature Research, v. 11, p. 11132, 12 2021. ISSN 2045-2322. Disponível em: <http://www.nature.com/articles/s41598-021-89779-z>.

[7] SAKAMOTO, T. Incorporating environmental variables into a modis-based crop yield estimation method for united states corn and soybeans through the use of a random forest regression algorithm. *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier B.V., v. 160, p. 208–228, 2 2020. ISSN 09242716. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0924271619303065>.

[8] TEAM, P. D. *PostGIS*. [S.l.], 2024. PostGIS is a spatial database extender for PostgreSQL. Disponível em: <https://postgis.net/>.

[9] ZALIPYNIS, R. A. R. Chronosdb. *Proceedings of the VLDB Endowment*, Association for Computing Machinery, v. 11, p. 1247–1261, 6 2018. ISSN 2150-8097. Disponível em: <https://dl.acm.org/doi/10.14778/3231751.3231754>.

[10] BAUMANN, P. et al. Array databases: Concepts, standards, implementations. *Journal of Big Data*, Springer, v. 8, p. 1–61, 2021.

[11] VU, T.; ELDAWY, A. R*-grove: Balanced spatial partitioning for large-scale datasets. *Frontiers in Big Data*, Frontiers Media S.A., v. 3, 8 2020. ISSN 2624-909X. Disponível em: <https://www.frontiersin.org/article/10.3389/fdata.2020.00028/full>.

[12] VU, T. et al. Incremental partitioning for efficient spatial data analytics. In: *Proceedings of the VLDB Endowment*. [S.l.]: VLDB Endowment, 2021. v. 15, p. 713–726. ISSN 21508097.

[13] HOJATI, M.; ROBERTS, S.; ROBERTSON, C. Dstree: A spatio-temporal indexing data structure for distributed networks. *Mathematical and Computational Applications*, MDPI AG, v. 29, p. 42, 5 2024. ISSN 2297-8747. Search about DHT. Disponível em: <https://www.mdpi.com/2297-8747/29/3/42>.

[14] HU, F. et al. A hierarchical indexing strategy for optimizing apache spark with hdfs to efficiently query big geospatial raster data. *International Journal of Digital Earth*, Taylor and Francis Ltd., v. 13, p. 410–428, 3 2020. ISSN 17538955.

[15] GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*. [S.l.], 2024. Disponível em: <https://gdal.org>.

[16] VILLARROYA, S.; BAUMANN, P. A survey on machine learning in array databases. *Applied Intelligence*, Springer, v. 53, p. 9799–9822, 5 2023. ISSN 0924-669X. Disponível em: <https://link.springer.com/10.1007/s10489-022-03979-2>.

[17] BAUMANN, P. Array databases. In: _____. *Encyclopedia of Database Systems*. New York, NY: Springer New York, 2016. p. 1–12. ISBN 978-1-4899-7993-3. Disponível em: <https://doi.org/10.1007/978-1-4899-7993-3_2061-2>.

[18] LI, J. et al. Study on spatio-temporal indexing model of geohazard monitoring data based on data stream clustering algorithm. *ISPRS International Journal of Geo-Information*, Multidisciplinary Digital Publishing Institute (MDPI), v. 13, p. 93, 3 2024. ISSN 2220-9964. Disponível em: <https://www.mdpi.com/2220-9964/13/3/93>.

[19] BAUMANN, P. Management of multidimensional discrete data. *The VLDB Journal*, Springer, v. 3, n. 4, p. 401–444, 1994. ISSN 0949-877X. Disponível em: <https://doi.org/10.1007/BF01231603>.

[20] BAUMANN, P. et al. The multidimensional database system rasdaman. In: *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. ACM, 1998. p. 575–577. ISBN 0897919955. Disponível em: <https://dl.acm.org/doi/10.1145/276304.276386>.

[21] BAUMANN, P. On the analysis-readiness of spatio-temporal earth data and suggestions for its enhancement. *Environmental Modelling & Software*, Elsevier Ltd, v. 176, p. 106017, 5 2024. ISSN 13648152. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1364815224000781>.

[22] TIAN, R. et al. A survey of spatio-temporal big data indexing methods in distributed environment. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Institute of Electrical and Electronics Engineers Inc., v. 15, p. 4132–4155, 2022. ISSN 1939-1404. Disponível em: <https://ieeexplore.ieee.org/document/9779567/>.

[23] ROMANI, L. A. S. et al. A new time series mining approach applied to multitemporal remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, v. 51, p. 140–150, 1 2013. ISSN 0196-2892. Disponível em: <http://ieeexplore.ieee.org/document/6215038/>.

[24] GONçALVES, R. R. et al. Analysis of ndvi time series using cross-correlation and forecasting methods for monitoring sugarcane fields in brazil. *International Journal of Remote Sensing*, Taylor and Francis Ltd., v. 33, p. 4653–4672, 2012. ISSN 13665901.

[25] SINGLA, S. Raptor: Large scale processing of big raster + vector data. In: *Proceedings of the 2021 International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2021. (SIGMOD '21), p. 2905–2907. ISBN 9781450383431. Disponível em: <https://doi.org/10.1145/3448016.3450585>.

[26] MORAGA, P. *Spatial statistics for data science*. Philadelphia, PA: Chapman & Hall/CRC, 2023.

[27] IBGE. *Introdução ao ambiente SIG QGIS*. [S.l.]: Instituto Brasileiro de Geografia e Estatística, 2018.

[28] DOMINGUES, C. V.; SIMõES, L. L. O sig na gestão pública: análise crítica de um caso bem-sucedido – desafios e perspectivas. *Exacta*, v. 5, p. 353–360, 8 2008. ISSN 1983-9308. Disponível em: <http://periodicos.uninove.br/index.php?journal=exacta&page=article&op=view&path[]=1185>.

[29] ESA. *SENTINEL-2 User Handbook*. 2015. MultiSpectral Instrument (MSI). Disponível em: <https://sentinel.esa.int/documents/247904/685211/Sentinel-2_User_Handbook.pdf/8869acdf-fd84-43ec-ae8c-3e80a436a16c?t=1438278087000>. Acesso em: 28.8.2022.

[30] NIXON, M.; AGUADO, A. *Feature Extraction and Image Processing*. 1ª. ed. [S.l.]: Newnes, 2002.

[31] GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens*. 3ª. ed. [S.l.]: Pearson Prentice Hall, 2010. ISBN 978-85-8143-586-2.

[32] TOMLIN, C. D. et al. *Geographic information systems and cartographic modeling*. Prentice Hall Englewood Cliffs, NJ, 1990. v. 249. Disponível em: <https://library.wur.nl/WebQuery/titel/564493>.

[33] BERRY, J. K. A mathematical structure for analyzing maps. *Environmental Management*, Springer, v. 11, p. 317–325, 1987. Disponível em: <https://doi.org/10.1007/BF01867159>.

[34] TOMLIN, C. Map algebra: one perspective. *Landscape and Urban Planning*, v. 30, n. 1, p. 3–12, 1994. ISSN 0169-2046. Special Issue Landscape Planning: Expanding the Tool Kit. Disponível em: <https://www.sciencedirect.com/science/article/pii/0169204694900639>.

[35] FRANK, A. U. Map algebra extended with functors for temporal data. In: SPRINGER. *International Conference on Conceptual Modeling*. 2005. p. 194–207. Disponível em: <https://link-springer-com.ez78.periodicos.capes.gov.br/chapter/10.1007/11568346_22>.

[36] SILVA-COIRA, F.; PARAMá, J. R.; LADRA, S. Map algebra on raster datasets represented by compact data structures. *Software: Practice and Experience*, v. 53, n. 6, p. 1362–1390, 2023. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3191>.

[37] GüTING, R. H. An introduction to spatial database systems. *The VLDB Journal*, Springer Science and Business Media LLC, v. 3, n. 4, p. 357–399, out. 1994. ISSN 0949-877X. Disponível em: <http://dx.doi.org/10.1007/BF01231602>.

[38] BAYER, R.; MCCREIGHT, E. Organization and maintenance of large ordered indices. In: *Proceedings of the 1970 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control - SIGFIDET '70*. ACM Press, 1970. p. 107. Disponível em: <http://portal.acm.org/citation.cfm?doid=1734663.1734671>.

[39] KNUTH, D. E. *The Art of Computer Programming, Volume 3: Sorting and Searching*. 2nd. ed. Reading, Massachusetts: Addison-Wesley, 1998. ISBN 0-201-89685-0.

[40] CORMEN, T. H. et al. *Introduction to Algorithms*. 3. ed. [S.l.]: MIT Press, 2009. ISBN 978-0-262-53305-8.

[41] CHI, L.; ZHU, X. Hashing techniques: A survey and taxonomy. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 50, n. 1, abr. 2017. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3047307>.

[42] GUTTMAN, A. R-trees: a dynamic index structure for spatial searching. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 1984. (SIGMOD '84), p. 47–57. ISBN 0897911288. Disponível em: <https://doi.org/10.1145/602259.602266>.

[43] BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, v. 18, p. 509–517, 9 1975. ISSN 0001-0782. Disponível em: <https://dl.acm.org/doi/10.1145/361002.361007>.

[44] SEDGEWICK, R.; WAYNE, K. *Algorithms*. 4th. ed. [S.l.]: Addison-Wesley, 2011. 992 p. ISBN 978-0-321-57351-3.

[45] BERG, M. de et al. *Computational Geometry: Algorithms and Applications*. 3rd. ed. [S.l.]: Springer, 2008. 386 p. ISBN 978-3-540-77974-2.

[46] SAMET, H. An overview of quadtrees, octrees, and related hierarchical data structures. *Theoretical Foundations of Computer Graphics and CAD*, Springer, p. 51–68, 1988.

[47] LI, Z. et al. A spatiotemporal indexing approach for efficient processing of big array-based climate data with mapreduce. *International Journal of Geographical Information Science*, v. 31, p. 17–35, 1 2017. ISSN 1365-8816. Disponível em: <https://www.tandfonline.com/doi/full/10.1080/13658816.2015.1131830>.

[48] MAHMOOD, A. R.; PUNNI, S.; AREF, W. G. Spatio-temporal access methods: a survey (2010 - 2017). *GeoInformatica*, Springer Science and Business Media, LLC, v. 23, p. 1–36, 1 2019. ISSN 1384-6175. Disponível em: <http://link.springer.com/10.1007/s10707-018-0329-2>.

[49] ZäSCHKE, T.; ZIMMERLI, C.; NORRIE, M. C. The ph-tree: a space-efficient storage structure and multi-dimensional index. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2014. (SIGMOD '14), p. 397–408. ISBN 9781450323765. Disponível em: <https://doi.org/10.1145/2588555.2588564>.

[50] CAI, R. et al. Ditir: distributed index for high throughput trajectory insertion and real-time temporal range query. *Proc. VLDB Endow.*, VLDB Endowment, v. 10, n. 12, p. 1865–1868, ago. 2017. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/3137765.3137795>.

[51] SKOVSGAARD, A.; SIDLAUSKAS, D.; JENSEN, C. S. Scalable top-k spatio-temporal term querying. In: *2014 IEEE 30th International Conference on Data Engineering*. [S.l.: s.n.], 2014. p. 148–159.

[52] WANG, H.; BELHASSENA, A. Parallel trajectory search based on distributed index. *Information Sciences*, v. 388-389, p. 62–83, 2017. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025517300178>.

[53] KEMMER, C. et al. Proposta de uma plataforma para o desenvolvimento e análise visual de modelos preditivos para doenças da soja. In: *Anais do XIV Congresso Brasileiro de Agroinformática*. Porto Alegre, RS, Brasil: SBC, 2023. p. 326–333. ISSN 2177-9724. Disponível em: <https://sol.sbc.org.br/index.php/sbiagro/article/view/26575>.

[54] BERG, M. de et al. *Computational Geometry: Algorithms and Applications*. 3. ed. Berlin: Springer-Verlag, 2008. ISBN 978-3540779735.

[55] DRUSCH, M. et al. Sentinel-2: Esa's optical high-resolution mission for gmes operational services. *Remote Sensing of Environment*, v. 120, p. 25–36, 2012. ISSN 0034-4257. The Sentinel Missions - New Opportunities for Science. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0034425712000636>.

Apêndices

APÊNDICE A – RESULTADOS DOS TESTES EXPERIMENTAIS PARA O CENÁRIO REAL

A.1 Uso de memória secundária e Operações de Inserção para o Cenário Real

Tabela 3 – Média de Uso de Memória Secundária Por Dado Matricial

Limiar	Tipo de Compressão	Nível de Particionamento	Tamanho Médio (MB)
5625	none	0	230.70
5625	none	1	230.71
5625	none	2	230.72
5625	none	4	230.74
5625	zstd	0	142.30
5625	zstd	1	142.32
5625	zstd	2	142.32
5625	zstd	4	142.34
22500	none	0	230.14
22500	none	1	230.14
22500	none	2	230.15
22500	none	4	230.16
22500	zstd	0	139.51
22500	zstd	1	139.53
22500	zstd	2	139.52
22500	zstd	4	139.53
90000	none	0	230.00
90000	none	1	230.00
90000	none	2	230.00
90000	none	4	230.01
90000	zstd	0	136.82
90000	zstd	1	136.84
90000	zstd	2	136.82
90000	zstd	4	136.82
360000	none	0	229.96

Continua na próxima página...

Tabela 3 – Continuação

Limiar	Tipo de Compressão	Nível de Particionamento	Tamanho Médio (MB)
360000	none	1	229.96
360000	none	2	229.96
360000	none	4	229.97
360000	zstd	0	134.10
360000	zstd	1	134.13
360000	zstd	2	134.08
360000	zstd	4	134.07

Tabela 4 – Tempo Médio de Inserção de Dados Matriciais

Limiar	Tipo de Compressão	Tempo Médio (ms)
5625	none	6835
5625	zstd	35126
22500	none	5987
22500	zstd	37918
90000	none	5600
90000	zstd	35844
360000	none	5734
360000	zstd	37265

Tabela 5 – Tempo médio de Particionamento

Limiar	Tipo de Compressão	Nível de Particionamento	Tempo Médio (μs)
5625	none	1	410341
5625	none	2	492221
5625	none	4	1035298
5625	zstd	1	405444
5625	zstd	2	492050
5625	zstd	4	1123847
22500	none	1	43281
22500	none	2	47990
22500	none	4	170488

Continua na próxima página...

Tabela 5 – Continuação

Limiar	Tipo de Compressão	Nível de Particionamento	Tempo Médio (μs)
22500	zstd	1	65006
22500	zstd	2	81086
22500	zstd	4	309899
90000	none	1	20270
90000	none	2	15841
90000	none	4	70828
90000	zstd	1	66696
90000	zstd	2	73210
90000	zstd	4	336019
360000	none	1	23351
360000	none	2	26555
360000	none	4	75360
360000	zstd	1	121465
360000	zstd	2	167703
360000	zstd	4	642960

A.2 Desempenho da Recuperação Espaço-Temporal para o Cenário Real

Tabela 6 – Tempo de Consultas por Nível de Particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	104158
0	5625	zstd	86734
0	22500	none	104602
0	22500	zstd	96217
0	90000	none	89830
0	90000	zstd	90179
0	360000	none	97623
0	360000	zstd	79995
1	5625	none	105374
1	5625	zstd	84781

Continua na próxima página...

Tabela 6 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
1	22500	none	99171
1	22500	zstd	89343
1	90000	none	84592
1	90000	zstd	85214
1	360000	none	65698
1	360000	zstd	64536
2	5625	none	104745
2	5625	zstd	90959
2	22500	none	95680
2	22500	zstd	80895
2	90000	none	79689
2	90000	zstd	79046
2	360000	none	67049
2	360000	zstd	55276
4	5625	none	117008
4	5625	zstd	97034
4	22500	none	99680
4	22500	zstd	88213
4	90000	none	78892
4	90000	zstd	82111
4	360000	none	68739
4	360000	zstd	62823

Tabela 7 – Tempo médio das consultas de JD por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	16148
0	5625	zstd	16545
0	22500	none	10652
0	22500	zstd	10640
0	90000	none	10565
0	90000	zstd	10629

Continua na próxima página...

Tabela 7 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	360000	none	15044
0	360000	zstd	12939
1	5625	none	14848
1	5625	zstd	14275
1	22500	none	8575
1	22500	zstd	8187
1	90000	none	6562
1	90000	zstd	6704
1	360000	none	8689
1	360000	zstd	8298
2	5625	none	14043
2	5625	zstd	14190
2	22500	none	6986
2	22500	zstd	6150
2	90000	none	4871
2	90000	zstd	4402
2	360000	none	6870
2	360000	zstd	4767
4	5625	none	14662
4	5625	zstd	14079
4	22500	none	6219
4	22500	zstd	6171
4	90000	none	3493
4	90000	zstd	3317
4	360000	none	3867
4	360000	zstd	3082

Tabela 8 – Tempo médio das consultas de JA por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	176111
0	5625	zstd	149358

Continua na próxima página...

Tabela 8 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	22500	none	179574
0	22500	zstd	162305
0	90000	none	162626
0	90000	zstd	168436
0	360000	none	196634
0	360000	zstd	162317
1	5625	none	173025
1	5625	zstd	138817
1	22500	none	162688
1	22500	zstd	145959
1	90000	none	149658
1	90000	zstd	146370
1	360000	none	133503
1	360000	zstd	128970
2	5625	none	170209
2	5625	zstd	150636
2	22500	none	154175
2	22500	zstd	135401
2	90000	none	127158
2	90000	zstd	133772
2	360000	none	112928
2	360000	zstd	97294
4	5625	none	188223
4	5625	zstd	157744
4	22500	none	158856
4	22500	zstd	141754
4	90000	none	123648
4	90000	zstd	131609
4	360000	none	114597
4	360000	zstd	102176

Tabela 9 – Tempo médio das consultas de IT por nível de particionamento

Nível de Particionamento	**Limiar**	**Tipo de Compressão**	**Tempo Médio Médio (μs)**
0	5625	none	124937
0	5625	zstd	96524
0	22500	none	129162
0	22500	zstd	121437
0	90000	none	98201
0	90000	zstd	91850
0	360000	none	76357
0	360000	zstd	60238
1	5625	none	134975
1	5625	zstd	106095
1	22500	none	134214
1	22500	zstd	121100
1	90000	none	101370
1	90000	zstd	107674
1	360000	none	51725
1	360000	zstd	53928
2	5625	none	137408
2	5625	zstd	113079
2	22500	none	134760
2	22500	zstd	107087
2	90000	none	115083
2	90000	zstd	104823
2	360000	none	85557
2	360000	zstd	66266
4	5625	none	157296
4	5625	zstd	125820
4	22500	none	144050
4	22500	zstd	125097
4	90000	none	118548
4	90000	zstd	120024
4	360000	none	93347
4	360000	zstd	89208

A.3 Consultas de Janela Deslizante para o Cenário Real

Tabela 10 – Taxa (memória secundária x resposta) por tipo de ROI em JD

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	none	57.983
0	0	zstd	63.571
0	1	none	74.549
0	1	zstd	77.937
0	2	none	84.529
0	2	zstd	86.634
0	4	none	92.995
0	4	zstd	94.064

Tabela 11 – Média de tempo das consultas por tamanho de janela em JD

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
0	0	none	13102
0	0	zstd	12688
0	1	none	9669
0	1	zstd	9366
0	2	none	8192
0	2	zstd	7377
0	4	none	7060
0	4	zstd	6663

Tabela 12 – Tempo médio das consultas de JD por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	16148
0	5625	zstd	16545
0	22500	none	10652

Continua na próxima página...

Tabela 12 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	22500	zstd	10640
0	90000	none	10565
0	90000	zstd	10629
0	360000	none	15044
0	360000	zstd	12939
1	5625	none	14848
1	5625	zstd	14275
1	22500	none	8575
1	22500	zstd	8187
1	90000	none	6562
1	90000	zstd	6704
1	360000	none	8689
1	360000	zstd	8298
2	5625	none	14043
2	5625	zstd	14190
2	22500	none	6986
2	22500	zstd	6150
2	90000	none	4871
2	90000	zstd	4402
2	360000	none	6870
2	360000	zstd	4767
4	5625	none	14662
4	5625	zstd	14079
4	22500	none	6219
4	22500	zstd	6171
4	90000	none	3493
4	90000	zstd	3317
4	360000	none	3867
4	360000	zstd	3082

Tabela 13 – Taxa (memória secundária x resposta) por limiar em JD

Limiar	**Nível de Particionamento**	**Tipo de Compressão**	**Taxa Média (%)**
5625	0	none	73.273
5625	0	zstd	80.164
5625	1	none	81.926
5625	1	zstd	86.310
5625	2	none	86.998
5625	2	zstd	90.013
5625	4	none	91.462
5625	4	zstd	93.546
22500	0	none	62.979
22500	0	zstd	70.029
22500	1	none	76.477
22500	1	zstd	80.794
22500	2	none	85.051
22500	2	zstd	87.625
22500	4	none	92.384
22500	4	zstd	93.680
90000	0	none	51.639
90000	0	zstd	57.234
90000	1	none	71.813
90000	1	zstd	75.072
90000	2	none	83.489
90000	2	zstd	85.412
90000	4	none	94.041
90000	4	zstd	94.613
360000	0	none	44.041
360000	0	zstd	46.858
360000	1	none	67.979
360000	1	zstd	69.571
360000	2	none	82.579
360000	2	zstd	83.485
360000	4	none	94.093
360000	4	zstd	94.416

Tabela 14 – Média de tempo das consultas por Limiar em JD

Nível de Particionamento	**Tipo de Compressão**	**Tempo Médio (μs)**	
5625	0	none	16148
5625	0	zstd	16545
5625	1	none	14848
5625	1	zstd	14275
5625	2	none	14043
5625	2	zstd	14190
5625	4	none	14662
5625	4	zstd	14079
22500	0	none	10652
22500	0	zstd	10640
22500	1	none	8575
22500	1	zstd	8187
22500	2	none	6986
22500	2	zstd	6150
22500	4	none	6219
22500	4	zstd	6171
90000	0	none	10565
90000	0	zstd	10629
90000	1	none	6562
90000	1	zstd	6704
90000	2	none	4871
90000	2	zstd	4402
90000	4	none	3493
90000	4	zstd	3317
360000	0	none	15044
360000	0	zstd	12939
360000	1	none	8689
360000	1	zstd	8298
360000	2	none	6870
360000	2	zstd	4767
360000	4	none	3867
360000	4	zstd	3082

Tabela 15 – Taxa (memória secundária x resposta) por tamanho de janela em JD

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
7.000	0	none	57.983
7.000	0	zstd	63.571
7.000	1	none	74.549
7.000	1	zstd	77.937
7.000	2	none	84.529
7.000	2	zstd	86.634
7.000	4	none	92.995
7.000	4	zstd	94.064

Tabela 16 – Média de tempo das consultas por tamanho de janela em JD

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
7	0	none	13102
7	0	zstd	12688
7	1	none	9669
7	1	zstd	9366
7	2	none	8192
7	2	zstd	7377
7	4	none	7060
7	4	zstd	6663

A.4 Consultas de Janela Aleatória para o Cenário Real

Tabela 17 – Taxa (memória secundária x resposta) por tipo de ROI em JA

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	none	12.669
0	0	zstd	21.462

Continua na próxima página...

Tabela 17 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	1	none	15.550
0	1	zstd	26.178
0	2	none	16.435
0	2	zstd	27.733
0	4	none	12.879
0	4	zstd	21.663

Tabela 18 – Média de tempo das consultas por tipo de ROI em JA

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
0	0	none	178736
0	0	zstd	160604
0	1	none	154719
0	1	zstd	140029
0	2	none	141117
0	2	zstd	129276
0	4	none	146331
0	4	zstd	133321

Tabela 19 – Tempo médio das consultas de JA por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	176111
0	5625	zstd	149358
0	22500	none	179574
0	22500	zstd	162305
0	90000	none	162626
0	90000	zstd	168436
0	360000	none	196634
0	360000	zstd	162317

Continua na próxima página...

Tabela 19 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
1	5625	none	173025
1	5625	zstd	138817
1	22500	none	162688
1	22500	zstd	145959
1	90000	none	149658
1	90000	zstd	146370
1	360000	none	133503
1	360000	zstd	128970
2	5625	none	170209
2	5625	zstd	150636
2	22500	none	154175
2	22500	zstd	135401
2	90000	none	127158
2	90000	zstd	133772
2	360000	none	112928
2	360000	zstd	97294
4	5625	none	188223
4	5625	zstd	157744
4	22500	none	158856
4	22500	zstd	141754
4	90000	none	123648
4	90000	zstd	131609
4	360000	none	114597
4	360000	zstd	102176

Tabela 20 – Taxa (memória secundária x resposta) por limiar em JA

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	0	none	28.701
5625	0	zstd	48.113
5625	1	none	33.114
5625	1	zstd	55.032

Continua na próxima página...

Tabela 20 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	2	none	32.462
5625	2	zstd	53.857
5625	4	none	23.009
5625	4	zstd	37.789
22500	0	none	14.958
22500	0	zstd	25.429
22500	1	none	18.638
22500	1	zstd	31.541
22500	2	none	19.620
22500	2	zstd	33.245
22500	4	none	15.280
22500	4	zstd	25.757
90000	0	none	5.445
90000	0	zstd	9.501
90000	1	none	8.057
90000	1	zstd	13.891
90000	2	none	10.101
90000	2	zstd	17.515
90000	4	none	8.748
90000	4	zstd	15.186
360000	0	none	1.574
360000	0	zstd	2.804
360000	1	none	2.390
360000	1	zstd	4.249
360000	2	none	3.557
360000	2	zstd	6.316
360000	4	none	4.480
360000	4	zstd	7.918

Tabela 21 – Média de tempo das consultas por limiar em JA

Limiar	**Nível de Particionamento**	**Tipo de Compressão**	**Tempo Médio (μs)**
5625	0	none	176111
5625	0	zstd	149358
5625	1	none	173025
5625	1	zstd	138817
5625	2	none	170209
5625	2	zstd	150636
5625	4	none	188223
5625	4	zstd	157744
22500	0	none	179574
22500	0	zstd	162305
22500	1	none	162688
22500	1	zstd	145959
22500	2	none	154175
22500	2	zstd	135401
22500	4	none	158856
22500	4	zstd	141754
90000	0	none	162626
90000	0	zstd	168436
90000	1	none	149658
90000	1	zstd	146370
90000	2	none	127158
90000	2	zstd	133772
90000	4	none	123648
90000	4	zstd	131609
360000	0	none	196634
360000	0	zstd	162317
360000	1	none	133503
360000	1	zstd	128970
360000	2	none	112928
360000	2	zstd	97294
360000	4	none	114597
360000	4	zstd	102176

Tabela 22 – Taxa (memória secundária x resposta) por tamanho de janela em JA

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
7	0	none	12.669
7	0	zstd	21.462
7	1	none	15.550
7	1	zstd	26.178
7	2	none	16.435
7	2	zstd	27.733
7	4	none	12.879
7	4	zstd	21.663

Tabela 23 – Média de tempo das consultas por tamanho de janela em JA

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
7	0	none	178736
7	0	zstd	160604
7	1	none	154719
7	1	zstd	140029
7	2	none	141117
7	2	zstd	129276
7	4	none	146331
7	4	zstd	133321

A.5 Consultas de Instante Temporal para o Cenário Real

Tabela 24 – Taxa (memória secundária x resposta) por tipo de ROI em IT

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	none	12.669
0	0	zstd	21.533

Continua na próxima página...

Tabela 24 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	1	none	9.576
0	1	zstd	16.158
0	2	none	6.989
0	2	zstd	11.799
0	4	none	3.352
0	4	zstd	5.607

Tabela 25 – Média de tempo das consultas por tipo de ROI em IT

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
0	0	none	107164
0	0	zstd	92512
0	1	none	105571
0	1	zstd	97199
0	2	none	118202
0	2	zstd	97814
0	4	none	128310
0	4	zstd	115037

Tabela 26 – Tempo médio das consultas de IT por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	124937
0	5625	zstd	96524
0	22500	none	129162
0	22500	zstd	121437
0	90000	none	98201
0	90000	zstd	91850
0	360000	none	76357
0	360000	zstd	60238

Continua na próxima página...

Tabela 26 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
1	5625	none	134975
1	5625	zstd	106095
1	22500	none	134214
1	22500	zstd	121100
1	90000	none	101370
1	90000	zstd	107674
1	360000	none	51725
1	360000	zstd	53928
2	5625	none	137408
2	5625	zstd	113079
2	22500	none	134760
2	22500	zstd	107087
2	90000	none	115083
2	90000	zstd	104823
2	360000	none	85557
2	360000	zstd	66266
4	5625	none	157296
4	5625	zstd	125820
4	22500	none	144050
4	22500	zstd	125097
4	90000	none	118548
4	90000	zstd	120024
4	360000	none	93347
4	360000	zstd	89208

Tabela 27 – Taxa (memória secundária x resposta) por limiar em IT

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	0	none	28.701
5625	0	zstd	48.276
5625	1	none	20.583
5625	1	zstd	34.279

Continua na próxima página...

Tabela 27 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	2	none	14.277
5625	2	zstd	23.697
5625	4	none	6.449
5625	4	zstd	10.530
22500	0	none	14.958
22500	0	zstd	25.513
22500	1	none	11.389
22500	1	zstd	19.328
22500	2	none	8.131
22500	2	zstd	13.797
22500	4	none	3.834
22500	4	zstd	6.447
90000	0	none	5.445
90000	0	zstd	9.530
90000	1	none	4.891
90000	1	zstd	8.458
90000	2	none	4.132
90000	2	zstd	7.185
90000	4	none	2.081
90000	4	zstd	3.607
360000	0	none	1.574
360000	0	zstd	2.812
360000	1	none	1.440
360000	1	zstd	2.567
360000	2	none	1.416
360000	2	zstd	2.518
360000	4	none	1.044
360000	4	zstd	1.845

Tabela 28 – Média de tempo das consultas por limiar em IT

Limiar	**Nível de Particionamento**	**Tipo de Compressão**	**Tempo Médio (μs)**
5625	0	none	124937
5625	0	zstd	96524
5625	1	none	134975
5625	1	zstd	106095
5625	2	none	137408
5625	2	zstd	113079
5625	4	none	157296
5625	4	zstd	125820
22500	0	none	129162
22500	0	zstd	121437
22500	1	none	134214
22500	1	zstd	121100
22500	2	none	134760
22500	2	zstd	107087
22500	4	none	144050
22500	4	zstd	125097
90000	0	none	98201
90000	0	zstd	91850
90000	1	none	101370
90000	1	zstd	107674
90000	2	none	115083
90000	2	zstd	104823
90000	4	none	118548
90000	4	zstd	120024
360000	0	none	76357
360000	0	zstd	60238
360000	1	none	51725
360000	1	zstd	53928
360000	2	none	85557
360000	2	zstd	66266
360000	4	none	93347
360000	4	zstd	89208

A.6 Uso de memória secundária e Operações de Inserção para o Cenário Sintético

Tabela 29 – Média de Uso de Memória Secundária Por Dado Matricial

Limiar	Tipo de Compressão	Nível de Particionamento	Tamanho Médio (MB)
5625	none	0	60.48
5625	none	1	60.48
5625	none	2	60.49
5625	none	3	60.49
5625	none	4	60.50
5625	zstd	0	46.75
5625	zstd	1	46.78
5625	zstd	2	46.80
5625	zstd	3	46.82
5625	zstd	4	46.83
22500	none	0	60.33
22500	none	1	60.33
22500	none	2	60.33
22500	none	3	60.34
22500	none	4	60.34
22500	zstd	0	46.07
22500	zstd	1	46.13
22500	zstd	2	46.16
22500	zstd	3	46.18
22500	zstd	4	46.19
90000	none	0	60.29
90000	none	1	60.29
90000	none	2	60.30
90000	none	3	60.30
90000	none	4	60.30
90000	zstd	0	45.54
90000	zstd	1	45.57
90000	zstd	2	45.55
90000	zstd	3	45.58
90000	zstd	4	45.59
360000	none	0	60.28

Continua na próxima página...

Tabela 29 – Continuação

Limiar	Tipo de Compressão	Nível de Particionamento	Tamanho Médio (MB)
360000	none	1	60.28
360000	none	2	60.29
360000	none	3	60.29
360000	none	4	60.29
360000	zstd	0	45.29
360000	zstd	1	45.37
360000	zstd	2	45.29
360000	zstd	3	45.34
360000	zstd	4	45.34

Tabela 30 – Tempo Médio de Inserção de Dados Matriciais

Limiar	Tipo de Compressão	Tempo Médio (ms)
5625	none	3242
5625	zstd	8408
22500	none	2893
22500	zstd	8834
90000	none	2732
90000	zstd	8343
360000	none	3012
360000	zstd	8515

Tabela 31 – Tempo médio de Particionamento

Limiar	Tipo de Compressão	Nível de Particionamento	Tempo Médio (μs)
5625	none	1	72612
5625	none	2	62417
5625	none	3	89791
5625	none	4	144204
5625	zstd	1	71210
5625	zstd	2	75657

Continua na próxima página...

Tabela 31 – Continuação

Limiar	Tipo de Compressão	Nível de Particionamento	Tempo Médio (μs)
5625	zstd	3	95152
5625	zstd	4	153154
22500	none	1	27874
22500	none	2	14492
22500	none	3	22118
22500	none	4	44839
22500	zstd	1	44766
22500	zstd	2	37521
22500	zstd	3	49052
22500	zstd	4	78496
90000	none	1	25558
90000	none	2	14544
90000	none	3	22494
90000	none	4	30972
90000	zstd	1	71972
90000	zstd	2	70909
90000	zstd	3	97667
90000	zstd	4	119418
360000	none	1	34463
360000	none	2	39141
360000	none	3	48713
360000	none	4	50386
360000	zstd	1	158277
360000	zstd	2	193567
360000	zstd	3	235588
360000	zstd	4	290217

A.7 Desempenho da Recuperação Espaço-Temporal para o Cenário Sintético

Tabela 32 – Tempo de Consultas por Nível de Particionamento

Nível de Particionamento	**Limiar**	**Tipo de Compressão**	**Tempo Médio Médio (μs)**
0	5625	none	88355
0	5625	zstd	78528
0	22500	none	83238
0	22500	zstd	87417
0	90000	none	69591
0	90000	zstd	71011
0	360000	none	102776
0	360000	zstd	101575
1	5625	none	66284
1	5625	zstd	65409
1	22500	none	64516
1	22500	zstd	64883
1	90000	none	39969
1	90000	zstd	39851
1	360000	none	62319
1	360000	zstd	59528
2	5625	none	68155
2	5625	zstd	65245
2	22500	none	55345
2	22500	zstd	55527
2	90000	none	27591
2	90000	zstd	27625
2	360000	none	43520
2	360000	zstd	40372
3	5625	none	63436
3	5625	zstd	64112
3	22500	none	54533
3	22500	zstd	53839
3	90000	none	22837
3	90000	zstd	21242
3	360000	none	30298
3	360000	zstd	31452
4	5625	none	66560

Continua na próxima página...

Tabela 32 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
4	5625	zstd	68107
4	22500	none	52983
4	22500	zstd	50665
4	90000	none	20434
4	90000	zstd	18752
4	360000	none	29532
4	360000	zstd	28526

Tabela 33 – Tempo médio das consultas de JD por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	13111
0	5625	zstd	13522
0	22500	none	11048
0	22500	zstd	12775
0	90000	none	12486
0	90000	zstd	12342
0	360000	none	23277
0	360000	zstd	24549
1	5625	none	7659
1	5625	zstd	7612
1	22500	none	7924
1	22500	zstd	7093
1	90000	none	6667
1	90000	zstd	7499
1	360000	none	9678
1	360000	zstd	10303
2	5625	none	6992
2	5625	zstd	7634
2	22500	none	5157
2	22500	zstd	5177
2	90000	none	4587

Continua na próxima página...

Tabela 33 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
2	90000	zstd	5163
2	360000	none	7577
2	360000	zstd	8034
3	5625	none	6806
3	5625	zstd	6961
3	22500	none	4046
3	22500	zstd	4296
3	90000	none	2857
3	90000	zstd	2521
3	360000	none	4827
3	360000	zstd	4389
4	5625	none	7763
4	5625	zstd	7464
4	22500	none	5532
4	22500	zstd	4140
4	90000	none	2007
4	90000	zstd	1902
4	360000	none	3077
4	360000	zstd	3139

Tabela 34 – Tempo médio das consultas de JA por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	165930
0	5625	zstd	145593
0	22500	none	155737
0	22500	zstd	161748
0	90000	none	133636
0	90000	zstd	137306
0	360000	none	194346
0	360000	zstd	191079
1	5625	none	120952

Continua na próxima página...

Tabela 34 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
1	5625	zstd	119067
1	22500	none	116778
1	22500	zstd	116957
1	90000	none	74972
1	90000	zstd	73973
1	360000	none	118728
1	360000	zstd	112963
2	5625	none	122700
2	5625	zstd	116554
2	22500	none	99202
2	22500	zstd	100022
2	90000	none	49746
2	90000	zstd	49757
2	360000	none	80001
2	360000	zstd	72619
3	5625	none	113909
3	5625	zstd	113062
3	22500	none	96869
3	22500	zstd	96758
3	90000	none	40732
3	90000	zstd	37876
3	360000	none	54177
3	360000	zstd	56426
4	5625	none	117075
4	5625	zstd	121875
4	22500	none	95710
4	22500	zstd	90238
4	90000	none	36917
4	90000	zstd	33386
4	360000	none	53285
4	360000	zstd	51430

Tabela 35 – Tempo médio das consultas de IT por nível de particionamento

Nível de Particionamento	**Limiar**	**Tipo de Compressão**	**Tempo Médio Médio (μs)**
0	5625	none	76707
0	5625	zstd	68235
0	22500	none	81693
0	22500	zstd	88976
0	90000	none	34893
0	90000	zstd	32881
0	360000	none	42422
0	360000	zstd	39176
1	5625	none	86065
1	5625	zstd	86106
1	22500	none	86159
1	22500	zstd	93457
1	90000	none	31470
1	90000	zstd	30994
1	360000	none	43484
1	360000	zstd	38470
2	5625	none	101249
2	5625	zstd	96755
2	22500	none	87005
2	22500	zstd	84806
2	90000	none	31829
2	90000	zstd	29274
2	360000	none	40837
2	360000	zstd	40823
3	5625	none	94223
3	5625	zstd	105115
3	22500	none	95290
3	22500	zstd	86960
3	90000	none	33264
3	90000	zstd	31674
3	360000	none	38253
3	360000	zstd	41902
4	5625	none	107977

Continua na próxima página...

Tabela 35 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
4	5625	zstd	102476
4	22500	none	76602
4	22500	zstd	85424
4	90000	none	30154
4	90000	zstd	29829
4	360000	none	43037
4	360000	zstd	40949

A.8 Consultas de Janela Deslizante para o Cenário Sintético

Tabela 36 – Taxa (memória secundária x resposta) por tipo de ROI em JD

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	none	24.459
0	0	zstd	24.978
0	1	none	61.429
0	1	zstd	61.657
0	2	none	80.195
0	2	zstd	80.296
0	3	none	89.701
0	3	zstd	89.746
0	4	none	93.097
0	4	zstd	93.127
1	0	none	30.655
1	0	zstd	32.340
1	1	none	64.306
1	1	zstd	65.068
1	2	none	81.201
1	2	zstd	81.543
1	3	none	90.086
1	3	zstd	90.234
1	4	none	93.281

Continua na próxima página...

Tabela 36 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
1	4	zstd	93.380
2	0	none	48.243
2	0	zstd	51.639
2	1	none	71.870
2	1	zstd	73.536
2	2	none	84.307
2	2	zstd	85.155
2	3	none	90.887
2	3	zstd	91.335
2	4	none	93.322
2	4	zstd	93.675

Tabela 37 – Média de tempo das consultas por tipo de ROI em JD

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
0	0	none	14269
0	0	zstd	14369
0	1	none	7108
0	1	zstd	7417
0	2	none	5284
0	2	zstd	6201
0	3	none	3670
0	3	zstd	3822
0	4	none	4024
0	4	zstd	3353
1	0	none	13327
1	0	zstd	14775
1	1	none	7858
1	1	zstd	7794
1	2	none	5963
1	2	zstd	5981
1	3	none	4308

Contnua na próxima página...

Tabela 37 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
1	3	zstd	4139
1	4	none	4249
1	4	zstd	3997
2	0	none	17346
2	0	zstd	18247
2	1	none	8980
2	1	zstd	9169
2	2	none	6988
2	2	zstd	7323
2	3	none	5924
2	3	zstd	5663
2	4	none	5511
2	4	zstd	5135

Tabela 38 – Tempo médio das consultas de JD por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	13111
0	5625	zstd	13522
0	22500	none	11048
0	22500	zstd	12775
0	90000	none	12486
0	90000	zstd	12342
0	360000	none	23277
0	360000	zstd	24549
1	5625	none	7659
1	5625	zstd	7612
1	22500	none	7924
1	22500	zstd	7093
1	90000	none	6667
1	90000	zstd	7499
1	360000	none	9678

Continua na próxima página...

Tabela 38 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
1	360000	zstd	10303
2	5625	none	6992
2	5625	zstd	7634
2	22500	none	5157
2	22500	zstd	5177
2	90000	none	4587
2	90000	zstd	5163
2	360000	none	7577
2	360000	zstd	8034
3	5625	none	6806
3	5625	zstd	6961
3	22500	none	4046
3	22500	zstd	4296
3	90000	none	2857
3	90000	zstd	2521
3	360000	none	4827
3	360000	zstd	4389
4	5625	none	7763
4	5625	zstd	7464
4	22500	none	5532
4	22500	zstd	4140
4	90000	none	2007
4	90000	zstd	1902
4	360000	none	3077
4	360000	zstd	3139

Tabela 39 – Taxa (memória secundária x resposta) por limiar em JD

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	0	none	45.714
5625	0	zstd	48.631
5625	1	none	70.683

Continua na próxima página...

Tabela 39 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	1	zstd	72.070
5625	2	none	84.046
5625	2	zstd	84.722
5625	3	none	90.484
5625	3	zstd	90.880
5625	4	none	93.029
5625	4	zstd	93.361
22500	0	none	36.597
22500	0	zstd	38.833
22500	1	none	66.730
22500	1	zstd	67.763
22500	2	none	82.656
22500	2	zstd	83.129
22500	3	none	90.492
22500	3	zstd	90.713
22500	4	none	93.482
22500	4	zstd	93.639
90000	0	none	30.087
90000	0	zstd	31.620
90000	1	none	64.063
90000	1	zstd	64.797
90000	2	none	80.674
90000	2	zstd	81.039
90000	3	none	90.116
90000	3	zstd	90.267
90000	4	none	93.389
90000	4	zstd	93.485
360000	0	none	25.411
360000	0	zstd	26.192
360000	1	none	61.998
360000	1	zstd	62.384
360000	2	none	80.226
360000	2	zstd	80.436
360000	3	none	89.808

Continua na próxima página...

Tabela 39 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
360000	3	zstd	89.893
360000	4	none	93.034
360000	4	zstd	93.091

Tabela 40 – Média de tempo das consultas por Limiar em JD

Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)	
5625	0	none	13111
5625	0	zstd	13522
5625	1	none	7659
5625	1	zstd	7612
5625	2	none	6992
5625	2	zstd	7634
5625	3	none	6806
5625	3	zstd	6961
5625	4	none	7763
5625	4	zstd	7464
22500	0	none	11048
22500	0	zstd	12775
22500	1	none	7924
22500	1	zstd	7093
22500	2	none	5157
22500	2	zstd	5177
22500	3	none	4046
22500	3	zstd	4296
22500	4	none	5532
22500	4	zstd	4140
90000	0	none	12486
90000	0	zstd	12342
90000	1	none	6667
90000	1	zstd	7499
90000	2	none	4587

Continua na próxima página...

Tabela 40 – Continuação

Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)	
90000	2	zstd	5163
90000	3	none	2857
90000	3	zstd	2521
90000	4	none	2007
90000	4	zstd	1902
360000	0	none	23277
360000	0	zstd	24549
360000	1	none	9678
360000	1	zstd	10303
360000	2	none	7577
360000	2	zstd	8034
360000	3	none	4827
360000	3	zstd	4389
360000	4	none	3077
360000	4	zstd	3139

Tabela 41 – Taxa (memória secundária x resposta) por tamanho de janela em JD

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
2	0	none	8.953
2	0	zstd	10.414
2	1	none	52.651
2	1	zstd	53.296
2	2	none	75.375
2	2	zstd	75.675
2	3	none	87.337
2	3	zstd	87.505
2	4	none	91.833
2	4	zstd	91.979
5	0	none	21.212
5	0	zstd	23.191
5	1	none	59.769

Continua na próxima página...

Tabela 41 – Continuação

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5	1	zstd	60.675
5	2	none	78.964
5	2	zstd	79.386
5	3	none	88.106
5	3	zstd	88.327
5	4	none	92.468
5	4	zstd	92.622
9	0	none	33.994
9	0	zstd	36.082
9	1	none	66.171
9	1	zstd	67.164
9	2	none	82.211
9	2	zstd	82.690
9	3	none	90.821
9	3	zstd	91.037
9	4	none	92.960
9	4	zstd	93.132
14	0	none	47.226
14	0	zstd	49.252
14	1	none	71.885
14	1	zstd	72.887
14	2	none	85.139
14	2	zstd	85.629
14	3	none	91.261
14	3	zstd	91.515
14	4	none	93.320
14	4	zstd	93.517
20	0	none	60.874
20	0	zstd	62.657
20	1	none	78.865
20	1	zstd	79.745
20	2	none	87.814
20	2	zstd	88.277
20	3	none	93.600

Continua na próxima página...

Tabela 41 – Continuação

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
20	3	zstd	93.808
20	4	none	95.585
20	4	zstd	95.719

Tabela 42 – Média de tempo das consultas por tamanho de janela em JD

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
2	0	none	12257
2	0	zstd	13212
2	1	none	6865
2	1	zstd	7960
2	2	none	4894
2	2	zstd	4683
2	3	none	3455
2	3	zstd	3306
2	4	none	3214
2	4	zstd	2873
5	0	none	13806
5	0	zstd	14336
5	1	none	7485
5	1	zstd	7784
5	2	none	6213
5	2	zstd	5539
5	3	none	3484
5	3	zstd	3718
5	4	none	3653
5	4	zstd	2952
9	0	none	15197
9	0	zstd	16111
9	1	none	8161
9	1	zstd	8459
9	2	none	5790

Continua na próxima página...

Tabela 42 – Continuação

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
9	2	zstd	6953
9	3	none	4463
9	3	zstd	4502
9	4	none	4619
9	4	zstd	3717
14	0	none	16704
14	0	zstd	17537
14	1	none	8361
14	1	zstd	7947
14	2	none	6483
14	2	zstd	7019
14	3	none	5581
14	3	zstd	5320
14	4	none	5404
14	4	zstd	4896
20	0	none	16940
20	0	zstd	17790
20	1	none	9037
20	1	zstd	8484
20	2	none	7011
20	2	zstd	8315
20	3	none	6186
20	3	zstd	5862
20	4	none	6083
20	4	zstd	6369

A.9 Consultas de Janela Aleatória para o Cenário Sintético

Tabela 43 – Taxa (memória secundária x resposta) por tipo de ROI em JA

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	none	0.341

Continua na próxima página...

Tabela 43 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	zstd	0.438
0	1	none	0.556
0	1	zstd	0.705
0	2	none	0.844
0	2	zstd	1.056
0	3	none	1.157
0	3	zstd	1.427
0	4	none	1.383
0	4	zstd	1.699
1	0	none	1.978
1	0	zstd	2.571
1	1	none	3.048
1	1	zstd	3.901
1	2	none	3.990
1	2	zstd	5.053
1	3	none	4.418
1	3	zstd	5.526
1	4	none	4.872
1	4	zstd	6.079
2	0	none	11.265
2	0	zstd	14.632
2	1	none	13.758
2	1	zstd	17.692
2	2	none	15.348
2	2	zstd	19.605
2	3	none	15.471
2	3	zstd	19.572
2	4	none	14.802
2	4	zstd	18.703

Tabela 44 – Média de tempo das consultas por tipo de ROI em JA

Tipo de ROI	**Nível de Particionamento**	**Tipo de Compressão**	**Tempo Médio (μs)**
0	0	none	151878
0	0	zstd	149820
0	1	none	95515
0	1	zstd	100095
0	2	none	77275
0	2	zstd	78603
0	3	none	65563
0	3	zstd	65752
0	4	none	67706
0	4	zstd	65159
1	0	none	147923
1	0	zstd	143714
1	1	none	105708
1	1	zstd	101206
1	2	none	84870
1	2	zstd	84128
1	3	none	72482
1	3	zstd	73610
1	4	none	71103
1	4	zstd	65940
2	0	none	187436
2	0	zstd	183261
2	1	none	122349
2	1	zstd	115920
2	2	none	101592
2	2	zstd	91483
2	3	none	91220
2	3	zstd	88730
2	4	none	88431
2	4	zstd	91598

Tabela 45 – Tempo médio das consultas de JA por nível de particionamento

Nível de Particionamento	**Limiar**	**Tipo de Compressão**	**Tempo Médio Médio (μs)**
0	5625	none	165930
0	5625	zstd	145593
0	22500	none	155737
0	22500	zstd	161748
0	90000	none	133636
0	90000	zstd	137306
0	360000	none	194346
0	360000	zstd	191079
1	5625	none	120952
1	5625	zstd	119067
1	22500	none	116778
1	22500	zstd	116957
1	90000	none	74972
1	90000	zstd	73973
1	360000	none	118728
1	360000	zstd	112963
2	5625	none	122700
2	5625	zstd	116554
2	22500	none	99202
2	22500	zstd	100022
2	90000	none	49746
2	90000	zstd	49757
2	360000	none	80001
2	360000	zstd	72619
3	5625	none	113909
3	5625	zstd	113062
3	22500	none	96869
3	22500	zstd	96758
3	90000	none	40732
3	90000	zstd	37876
3	360000	none	54177
3	360000	zstd	56426
4	5625	none	117075

Continua na próxima página...

Tabela 45 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
4	5625	zstd	121875
4	22500	none	95710
4	22500	zstd	90238
4	90000	none	36917
4	90000	zstd	33386
4	360000	none	53285
4	360000	zstd	51430

Tabela 46 – Taxa (memória secundária x resposta) por limiar em JA

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	0	none	10.615
5625	0	zstd	13.685
5625	1	none	12.657
5625	1	zstd	16.060
5625	2	none	14.272
5625	2	zstd	17.871
5625	3	none	14.245
5625	3	zstd	17.594
5625	4	none	14.209
5625	4	zstd	17.510
22500	0	none	4.839
22500	0	zstd	6.315
22500	1	none	6.373
22500	1	zstd	8.226
22500	2	none	7.542
22500	2	zstd	9.656
22500	3	none	7.764
22500	3	zstd	9.802
22500	4	none	7.766
22500	4	zstd	9.767
90000	0	none	2.092

Continua na próxima página...

Tabela 46 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
90000	0	zstd	2.767
90000	1	none	3.087
90000	1	zstd	4.075
90000	2	none	3.488
90000	2	zstd	4.617
90000	3	none	4.145
90000	3	zstd	5.443
90000	4	none	4.167
90000	4	zstd	5.467
360000	0	none	0.566
360000	0	zstd	0.753
360000	1	none	1.032
360000	1	zstd	1.371
360000	2	none	1.606
360000	2	zstd	2.140
360000	3	none	1.907
360000	3	zstd	2.528
360000	4	none	1.934
360000	4	zstd	2.564

Tabela 47 – Média de tempo das consultas por limiar em JA

Limiar	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
5625	0	none	165930
5625	0	zstd	145593
5625	1	none	120952
5625	1	zstd	119067
5625	2	none	122700
5625	2	zstd	116554
5625	3	none	113909
5625	3	zstd	113062
5625	4	none	117075

Continua na próxima página...

Tabela 47 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
5625	4	zstd	121875
22500	0	none	155737
22500	0	zstd	161748
22500	1	none	116778
22500	1	zstd	116957
22500	2	none	99202
22500	2	zstd	100022
22500	3	none	96869
22500	3	zstd	96758
22500	4	none	95710
22500	4	zstd	90238
90000	0	none	133636
90000	0	zstd	137306
90000	1	none	74972
90000	1	zstd	73973
90000	2	none	49746
90000	2	zstd	49757
90000	3	none	40732
90000	3	zstd	37876
90000	4	none	36917
90000	4	zstd	33386
360000	0	none	194346
360000	0	zstd	191079
360000	1	none	118728
360000	1	zstd	112963
360000	2	none	80001
360000	2	zstd	72619
360000	3	none	54177
360000	3	zstd	56426
360000	4	none	53285
360000	4	zstd	51430

Tabela 48 – Taxa (memória secundária x resposta) por tamanho de janela em JA

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
2	0	none	4.528
2	0	zstd	5.906
2	1	none	5.001
2	1	zstd	6.442
2	2	none	4.286
2	2	zstd	5.467
2	3	none	3.148
2	3	zstd	3.970
2	4	none	2.521
2	4	zstd	3.164
5	0	none	4.528
5	0	zstd	5.881
5	1	none	5.542
5	1	zstd	7.116
5	2	none	6.075
5	2	zstd	7.739
5	3	none	6.164
5	3	zstd	7.772
5	4	none	5.426
5	4	zstd	6.816
9	0	none	4.528
9	0	zstd	5.867
9	1	none	5.956
9	1	zstd	7.636
9	2	none	7.195
9	2	zstd	9.152
9	3	none	7.389
9	3	zstd	9.292
9	4	none	7.868
9	4	zstd	9.893
14	0	none	4.528
14	0	zstd	5.865
14	1	none	6.174

Continua na próxima página...

Tabela 48 – Continuação

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
14	1	zstd	7.917
14	2	none	7.872
14	2	zstd	10.022
14	3	none	8.861
14	3	zstd	11.164
14	4	none	9.259
14	4	zstd	11.650
20	0	none	4.528
20	0	zstd	5.880
20	1	none	6.264
20	1	zstd	8.054
20	2	none	8.208
20	2	zstd	10.477
20	3	none	9.514
20	3	zstd	12.010
20	4	none	10.019
20	4	zstd	12.613

Tabela 49 – Média de tempo das consultas por tamanho de janela em JA

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
2	0	none	75642
2	0	zstd	65943
2	1	none	67541
2	1	zstd	66363
2	2	none	67725
2	2	zstd	67510
2	3	none	69067
2	3	zstd	65948
2	4	none	64459
2	4	zstd	66293
5	0	none	112246

Continua na próxima página...

Tabela 49 – Continuação

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
5	0	zstd	105109
5	1	none	79154
5	1	zstd	80861
5	2	none	75159
5	2	zstd	72681
5	3	none	66937
5	3	zstd	70978
5	4	none	71296
5	4	zstd	69980
9	0	none	162795
9	0	zstd	162409
9	1	none	105534
9	1	zstd	97966
9	2	none	85867
9	2	zstd	81053
9	3	none	76627
9	3	zstd	75355
9	4	none	75407
9	4	zstd	72121
14	0	none	212698
14	0	zstd	212938
14	1	none	129825
14	1	zstd	129827
14	2	none	99215
14	2	zstd	96074
14	3	none	82458
14	3	zstd	81972
14	4	none	81995
14	4	zstd	77578
20	0	none	248680
20	0	zstd	248259
20	1	none	157233
20	1	zstd	153685
20	2	none	111594

Continua na próxima página...

Tabela 49 – Continuação

Tamanho da Janela	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
20	2	zstd	106373
20	3	none	87020
20	3	zstd	85900
20	4	none	85578
20	4	zstd	85189

A.10 Consultas de Instante Temporal para o Cenário Sintético

Tabela 50 – Taxa (memória secundária x resposta) por tipo de ROI em IT

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
0	0	none	0.341
0	0	zstd	0.443
0	1	none	0.320
0	1	zstd	0.408
0	2	none	0.304
0	2	zstd	0.381
0	3	none	0.270
0	3	zstd	0.334
0	4	none	0.249
0	4	zstd	0.306
1	0	none	1.978
1	0	zstd	2.604
1	1	none	1.754
1	1	zstd	2.256
1	2	none	1.441
1	2	zstd	1.829
1	3	none	1.034
1	3	zstd	1.297
1	4	none	0.880
1	4	zstd	1.097
2	0	none	11.265

Continua na próxima página...

Tabela 50 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
2	0	zstd	14.813
2	1	none	7.963
2	1	zstd	10.276
2	2	none	5.673
2	2	zstd	7.252
2	3	none	3.776
2	3	zstd	4.776
2	4	none	2.837
2	4	zstd	3.571

Tabela 51 – Média de tempo das consultas por tipo de ROI em IT

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
0	0	none	50964
0	0	zstd	55158
0	1	none	57290
0	1	zstd	67129
0	2	none	58515
0	2	zstd	55735
0	3	none	61357
0	3	zstd	59402
0	4	none	60148
0	4	zstd	56785
1	0	none	57287
1	0	zstd	53818
1	1	none	58347
1	1	zstd	58471
1	2	none	60551
1	2	zstd	70998
1	3	none	63682
1	3	zstd	63187
1	4	none	57655

Continua na próxima página...

Tabela 51 – Continuação

Tipo de ROI	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
1	4	zstd	65119
2	0	none	68536
2	0	zstd	62974
2	1	none	69747
2	1	zstd	61170
2	2	none	76624
2	2	zstd	62011
2	3	none	70734
2	3	zstd	76650
2	4	none	75524
2	4	zstd	72104

Tabela 52 – Tempo médio das consultas de IT por nível de particionamento

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
0	5625	none	76707
0	5625	zstd	68235
0	22500	none	81693
0	22500	zstd	88976
0	90000	none	34893
0	90000	zstd	32881
0	360000	none	42422
0	360000	zstd	39176
1	5625	none	86065
1	5625	zstd	86106
1	22500	none	86159
1	22500	zstd	93457
1	90000	none	31470
1	90000	zstd	30994
1	360000	none	43484
1	360000	zstd	38470
2	5625	none	101249

Continua na próxima página...

Tabela 52 – Continuação

Nível de Particionamento	Limiar	Tipo de Compressão	Tempo Médio Médio (μs)
2	5625	zstd	96755
2	22500	none	87005
2	22500	zstd	84806
2	90000	none	31829
2	90000	zstd	29274
2	360000	none	40837
2	360000	zstd	40823
3	5625	none	94223
3	5625	zstd	105115
3	22500	none	95290
3	22500	zstd	86960
3	90000	none	33264
3	90000	zstd	31674
3	360000	none	38253
3	360000	zstd	41902
4	5625	none	107977
4	5625	zstd	102476
4	22500	none	76602
4	22500	zstd	85424
4	90000	none	30154
4	90000	zstd	29829
4	360000	none	43037
4	360000	zstd	40949

Tabela 53 – Taxa (memória secundária x resposta) por limiar em IT

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	0	none	10.615
5625	0	zstd	13.860
5625	1	none	7.343
5625	1	zstd	9.352
5625	2	none	5.311

Continua na próxima página...

Tabela 53 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
5625	2	zstd	6.660
5625	3	none	3.535
5625	3	zstd	4.370
5625	4	none	2.773
5625	4	zstd	3.409
22500	0	none	4.839
22500	0	zstd	6.395
22500	1	none	3.669
22500	1	zstd	4.753
22500	2	none	2.714
22500	2	zstd	3.480
22500	3	none	1.824
22500	3	zstd	2.308
22500	4	none	1.419
22500	4	zstd	1.782
90000	0	none	2.092
90000	0	zstd	2.799
90000	1	none	1.777
90000	1	zstd	2.357
90000	2	none	1.276
90000	2	zstd	1.690
90000	3	none	0.968
90000	3	zstd	1.273
90000	4	none	0.746
90000	4	zstd	0.978
360000	0	none	0.566
360000	0	zstd	0.760
360000	1	none	0.594
360000	1	zstd	0.792
360000	2	none	0.589
360000	2	zstd	0.786
360000	3	none	0.445
360000	3	zstd	0.591
360000	4	none	0.350

Continua na próxima página...

Tabela 53 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Taxa Média (%)
360000	4	zstd	0.464

Tabela 54 – Média de tempo das consultas por limiar em IT

Limiar	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
5625	0	none	76707
5625	0	zstd	68235
5625	1	none	86065
5625	1	zstd	86106
5625	2	none	101249
5625	2	zstd	96755
5625	3	none	94223
5625	3	zstd	105115
5625	4	none	107977
5625	4	zstd	102476
22500	0	none	81693
22500	0	zstd	88976
22500	1	none	86159
22500	1	zstd	93457
22500	2	none	87005
22500	2	zstd	84806
22500	3	none	95290
22500	3	zstd	86960
22500	4	none	76602
22500	4	zstd	85424
90000	0	none	34893
90000	0	zstd	32881
90000	1	none	31470
90000	1	zstd	30994
90000	2	none	31829
90000	2	zstd	29274
90000	3	none	33264

Continua na próxima página...

Tabela 54 – Continuação

Limiar	Nível de Particionamento	Tipo de Compressão	Tempo Médio (μs)
90000	3	zstd	31674
90000	4	none	30154
90000	4	zstd	29829
360000	0	none	42422
360000	0	zstd	39176
360000	1	none	43484
360000	1	zstd	38470
360000	2	none	40837
360000	2	zstd	40823
360000	3	none	38253
360000	3	zstd	41902
360000	4	none	43037
360000	4	zstd	40949