Estudo Comparativo de Banco de Dados NoSQL para Gerenciamento de Séries Espaço-Temporais

Luís Eduardo Damasceno[1], Melise Maria V. de Paula[1], Vanessa Cristina O. de Souza[1], Flávio Belizário da Silva Mota[2]

[1]Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)
Caixa Postal 50 CEP: 37500–903 – Itajubá – MG – Brasil

[2]Descubra Soluções em Decisões Estratégicas – Itajubá – MG – Brasil

{d2022010320, melise, vanessasouza}@unifei.edu.br

flavio.belizario.mota@gmail.com

Abstract. *This study evaluates the performance of the MongoDB and InfluxDB database management systems in handling spatiotemporal data from the Brazil Data Cube (BDC) project, with a focus on applications related to coffee cultivation. The data were processed and modeled for storage in each database, both deployed in Docker containers. The methodology included data load tests and queries simulating real-world scenarios: one retrieving a time series for a single pixel, and another querying a spatial region on a specific date. The results showed that MongoDB outperformed InfluxDB in data loading and spatial region queries, while InfluxDB achieved better performance in pixel-based time series queries. This work offers a practical analysis to support the selection of appropriate technologies for efficient spatiotemporal data management.*

Resumo. *Este artigo avalia o desempenho dos SGBDs MongoDB e InfluxDB no gerenciamento de dados espaço-temporais oriundos do projeto Brazil Data Cube (BDC), utilizados em análises relacionadas à cafeicultura. Os dados foram tratados, modelados e armazenados em instâncias executadas em containers Docker. A metodologia envolveu testes de carga e consultas simulando dois cenários com dados reais: recuperação de séries temporais para um único pixel e consultas espaciais em uma data específica. Os resultados indicaram que o MongoDB apresentou melhor desempenho nas operações de carga e nas consultas espaciais, enquanto o InfluxDB se destacou nas consultas por pixel. O estudo contribui com uma análise prática que pode orientar a escolha de tecnologias para aplicações que lidam com dados espaço-temporais em contextos agrícolas.*

1. Introdução

A agricultura enfrenta diversos desafios impulsionados por pressões crescentes sobre a produção e pela intensificação de eventos climáticos extremos [Queiroz et al. 2022]. Nesse contexto, o uso de sensores remotos surge como uma alternativa capaz de promover práticas mais sustentáveis e produtivas. A partir de dados espectrais, é possível monitorar a saúde e o desenvolvimento das plantas ao longo do ciclo fenológico, mapear áreas cafeeiras, identificar zonas mais produtivas e acompanhar condições agronômicas como perda foliar, estresse hídrico e teores de nutrientes [Formaggio and Sanches 2017].

Apesar dos avanços tecnológicos, o acesso às informações agronômicas de qualidade ainda é limitado [Bassoi et al. 2019]. Nesse contexto, destaca-se a iniciativa Brazil Data Cube (BDC), do Instituto Nacional de Pesquisas Espaciais (INPE), que disponibiliza dados abertos, organizados em cubos espaço-temporais [Zaglia et al. 2019]. O BDC fornece séries históricas de sensoriamento remoto para todo o território brasileiro, incluindo bandas espectrais e índices de vegetação. Esses dados são estruturados em matrizes multidimensionais que peritem o acesso, a análise e a visualização de grandes volumes de dados geoespaciais [Batina 2023].

Neste estudo, foram analisadas características das lavouras de café a partir de bandas espectrais específicas verde (500–550 nm), vermelha (650–700 nm) e infravermelha próxima (NIR, 850–950 nm) e do índice de diferença normalizada da vegetação (NDVI), calculado com base na refletância dessas bandas. Esses dados foram extraídos do BDC e associados às coordenadas geográficas específicas representando pontos (píxeis) na região considerada. No BDC, os píxeis são submetidos a um processo de composição temporal que seleciona, dentro de um intervalo, os melhores registros [Zaglia et al. 2019], permitindo análises tanto espaciais quanto temporais do desenvolvimento das culturas ao longo do ciclo agrícola.

Considerando a natureza espaço-temporal dos dados, este trabalho avalia dois mecanismos de persistência com suporte a esse tipo de estrutura. O MongoDB [MongoDB Documentation], que oferece suporte nativo a dados espaciais e funcionalidades específicas para séries temporais, como a TimeSeries Collection [Hachimi et al. 2023]. Por sua vez, o InfluxDB [InfluxDB Documentation] é um SGBD projetado para séries temporais que incorporou suporte a dados espaciais, viabilizando consultas por localização geográfica [Choi et al. 2022].

Dessa forma, este trabalho teve como objetivo avaliar o desempenho do MongoDB e do InfluxDB em operações de carga e consulta sobre séries espaço-temporais, utilizando dados da cafeicultura em cenários simulados. Os resultados mostraram que o MongoDB apresenta melhor desempenho na inserção de dados e nas consultas espaciais que envolvem regiões delimitadas, enquanto o InfluxDB se destaca em consultas pontuais que retornam séries temporais associadas a um único pixel.

2. Trabalhos relacionados

Os bancos de dados NoSQL têm se destacado na gestão de dados geo-temporais por sua escalabilidade, flexibilidade e capacidade de lidar com grandes volumes de dados dinâmicos, como os gerados por sensores e coordenadas espaciais [Queiroz et al. 2013]. Essas características são especialmente úteis na agricultura de precisão, como na cafeicultura, onde a diversidade e o volume de dados exigem soluções eficientes para armazenamento e análise espaço-temporal [Bassoi et al. 2019].

O artigo de [John et al. 2023] apresenta uma análise comparativa entre diferentes bancos de dados de séries temporais aplicados a plataformas de Internet das Coisas (IoT) em contextos urbanos inteligentes. O estudo testou a eficácia do InfluxDB em relação a outros sistemas, como MongoDB, o Mongodb com *collection TimeSeries*, PostgreSQL e o DynamoDB, utilizando uma metodologia que incluiu a execução de operações comuns de inserção e consulta em um ambiente controlado. Os resultados indicaram que o InfluxDB superou os bancos de dados tradicionais em termos de eficiência de armazenamento e

velocidade de consulta, reduzindo um conjunto de dados de 350 MB para apenas 69 MB, em comparação aos 787 MB do PostgreSQL. O MongoDB apresentou o segundo melhor desempenho, ficando à frente do DynamoDB e do PostgreSQL nas operações de consulta.

Em [Tripathi et al. 2023], também foi analisado o desempenho e a escalabilidade do MongoDB com o Influxdb em aplicações de Internet das Coisas (IoT). Os autores utilizam um *framework* de *benchmarking* com contêineres Docker, realizando testes com um *script* em Python que gera solicitações simultâneas para ambos SGBD. Foram analisadas métricas como latência média, uso de CPU e memória durante operações de escrita e consulta em diferentes níveis de concorrência. Os resultados mostraram que, em cargas de trabalho altas, o MongoDB mantém latência estável e uso consistente de CPU, enquanto o InfluxDB, embora inicialmente eficiente em memória, apresenta picos de latência e aumento do uso de CPU em situações de maior concorrência.

O estudo apresentado por [Zehra 2017] investiga de forma aprofundada o desempenho de diferentes sistemas de bancos de dados de séries temporais (TSDB), com destaque para o InfluxDB. A pesquisa compara o InfluxDB com outras soluções como Cassandra, OpenTSDB e Elasticsearch por meio de um *benchmarking*, utilizando um cenário que simula métricas de servidores e aplicações em ambientes DevOps. Foram analisadas métricas como taxa de ingestão, tempo de resposta em consultas com e sem agregação e armazenamento. Os resultados mostraram que o InfluxDB apresenta desempenho superior em operações de escrita e leitura, indicando sua eficiência em ambientes com alta demanda de dados temporais.

No estudo de [Kim et al. 2023], o GeoYCSB é um *framework* de benchmark concebido para avaliar o desempenho e a escalabilidade de bancos de dados NoSQL em cargas de trabalho geoespaciais, estendendo as funcionalidades do YCSB (Yahoo! Cloud Serving Benchmark). Ele realiza *microbenchmarks* que avaliam operações espaciais específicas, como consultas de proximidade, dentro e interseção. Além disso, executa *macrobenchmarks* que simulam cenários de uso mais complexos, como a identificação e remoção de artes de grafites em áreas urbanas, utilizando *datasets* representativos. A taxa de transferência (*throughput*) e a latência são as principais métricas de avaliação, revelando a eficiência e responsividade dos sistemas. Para validar o funcionamento do sistema, foram realizados testes utilizando os dados de artes de grafite da cidade de Tempe. Os experimentos, conduzidos com os bancos de dados MongoDB e Couchbase, demonstraram que ambos escalam bem sob diferentes cargas de trabalho, embora apresentem variações de desempenho conforme a complexidade das operações e os níveis de consistência. Esses resultados reforçam a importância do GeoYCSB na construção de um fluxo de *benchmark* específico para dados espaciais.

Entretanto, ainda são limitados os estudos que comparam diretamente o desempenho do MongoDB e do InfluxDB em cenários que integram simultaneamente dimensões espaciais e temporais, especialmente em aplicações voltadas à agricultura de precisão. Este trabalho busca preencher essa lacuna ao realizar uma avaliação comparativa entre dois SGBDs NoSQL com naturezas distintas, um orientado nativamente a dados espaciais com suporte a séries temporais (MongoDB) [Mongodb Storage], e outro projetado para séries temporais com extensões geoespaciais (InfluxDB) [InfluxDB Geo Package]. Dessa forma, o estudo contribui com uma análise do comportamento desses sistemas frente a dados espaço-temporais.

3. Metodologia

Este estudo foi conduzido em cinco etapas: (i) seleção dos SGBDs, considerando alternativas com suporte a dados espaço-temporais; (ii) aquisição e tratamento dos dados; (iii) modelagem dos dados espaço-temporais, com definição das estruturas adotadas em cada SGBD; (iv) execução de testes de carga e consulta; e (v) avaliação comparativa com base em métricas de desempenho.

Em relação à seleção dos SGBDs, considerando a natureza espaço-temporal dos dados e a necessidade de armazenar e processar grandes volumes de informações, requisitos como escalabilidade e flexibilidade de esquemas tornam-se fundamentais [Mehmood et al. 2017]. Por esse motivo, optou-se por SGBDs baseados em modelos não relacionais (NoSQL) que oferecem suporte nativo a dados geoespaciais e séries temporais.

Dentre as alternativas disponíveis, o MongoDB foi selecionado por atender a esses critérios e oferecer suporte específico ao armazenamento de séries temporais por meio de coleções do tipo *time series* [Hachimi et al. 2023]. Esse tipo de coleção é otimizado para reduzir o uso de disco, simplificar o tratamento dos dados e minimizar as operações de entrada/saída (E/S) durante a leitura, o que contribui diretamente para a eficiência do sistema [Colosi et al. 2022].

No suporte a dados espaciais, o MongoDB utiliza o formato GeoJSON, permitindo a representação de pontos, linhas e polígonos. Além disso, permite a execução de consultas geoespaciais avançadas, como interseção, inclusão e proximidade [Makris et al. 2019], além de possibilitar a busca por pontos específicos, o cálculo de distâncias em relação a um ponto de referência e o retorno de correspondências exatas em consultas por coordenadas geográficas. Essas características tornam o MongoDB uma possível solução para aplicações espaço-temporais e justificam sua análise neste estudo.

O InfluxDB é um banco de dados NoSQL de alto desempenho, projetado para armazenar e consultar grandes volumes de séries temporais com alta taxa de ingestão, compressão eficiente e suporte a consultas em tempo real [Petre et al. 2019]. Sua *engine* de armazenamento organiza os dados em longas sequências temporais, otimizando tanto a compactação quanto o acesso [InfluxDB Documentation]. A partir da versão 0.63.0 do Flux(linguagem de consulta), o InfluxDB passou a oferecer suporte nativo a dados geoespaciais por meio do pacote *geo*, que permite filtrar e agrupar informações com base em localização geográfica [InfluxDB Geo Package].

Também foram considerados bancos de dados NoSQL voltados especificamente para séries temporais, como o OpenTSDB [OpenTSDB Documentation], um sistema distribuído construído sobre o Hadoop e o HBase, e o CnosDB, uma solução *cloud-native* com foco em dados temporais [CnosDB Documentation]. No entanto, através do ranking da DB-Engines [DB-Engines], o InfluxDB se destacava como líder entre os bancos de dados de séries temporais, apresentando características de modelo espacial, ao contrário do OpenTSDB e CnosDB, que não possuíam essa funcionalidade.

Na etapa de Aquisição e Tratamento dos dados, foi selecionado o município de Três Pontas que se destaca pela sua sólida tradição no cultivo de café de alta qualidade, consolidando-se como uma região estratégica com forte inserção no mercado global [Sousa 2023]. A área do município é de 689,794 km².

As séries temporais foram obtidas a partir do BDC. A aquisição foi realizada por meio da API STAC, utilizando a linguagem R (versão 4.4.1). O detalhamento do processo de aquisição pode ser consultado no repositório do trabalho[1]. Os dados foram extraídos da coleção S2-16D-2, referente a imagens dos satélites Sentinel-2A/2B, com resolução espacial de 10 metros e resolução temporal de 16 dias. O intervalo considerado foi de 01/01/2017 a 16/03/2025. Foram recuperadas as bandas espectrais *Red*, *Green* e *NIR*, além do índice de vegetação (IV) NDVI.

O município de Três Pontas possui um total de 6.398.277 píxeis do satélite Sentinel. Devido à elevada quantidade de dados e às limitações da infraestrutura utilizada nos testes, foi necessário aplicar uma estratégia de amostragem para reduzir o volume de píxeis a serem armazenados. Essa redução foi realizada por meio de um *loop* que percorre os píxeis de cada *raster*, selecionando apenas um a cada 500, o que resultou em uma amostra final de 12.784 píxeis. Apesar da redução, a representatividade da área foi mantida, uma vez que a densidade amostral permaneceu suficiente para garantir a validade da análise de desempenho dos bancos de dados. Para cada píxel selecionado, foram associadas quatro séries temporais, uma para cada banda espectral e para o índice de vegetação, compostas por registros em intervalos regulares de 16 dias. Como resultado, o banco de dados final totalizou 2.403.392 registros.

Após o tratamento dos dados, foi necessário definir a modelagem adequada para armazenamento em cada um dos SGBDs utilizados. No caso do MongoDB, utilizando a coleção do tipo *Time Series*, cada documento da coleção representa um único pixel em uma determinada data, contendo suas coordenadas geográficas, valores das bandas espectrais e o IV, conforme ilustrado na Modelagem MongoDB (Figura 1).

Para o InfluxDB, os dados foram organizados nas *measurements*, que são estruturas equivalentes às tabelas em bancos relacionais e não requerem definição prévia de esquema [Zhou et al. 2017]. Em uma *measurements*, as *tags* são pares chave-valor usados para indexar e acelerar consultas, especialmente filtragens por atributos como local ou categoria [Zehra 2017]. Para utilizar o pacote Geo, o campo *tag* da *measurement* deve ser definido a partir do identificador S2CellId,

As S2CellIds são identificadores únicos de células geográficas definidos pelo sistema S2, que divide a superfície da Terra em uma hierarquia de quadriláteros geodésicos [S2 Documentation]. Cada célula é representada por um ID de 64 bits que codifica sua posição e nível de subdivisão. Esses IDs seguem uma curva de preenchimento espacial, o que garante proximidade entre células vizinhas no índice, otimizando a busca e a indexação espacial [S2 Documentation]. Para o InfluxDB, utilizou-se a biblioteca S2 na linguagem R para gerar os identificadores s2_cell_id a partir das coordenadas dos pontos em WGS84.

Além da tag com o S2CellId, os registros precisam obrigatoriamente conter dois campos no formato *field*: lat e lon, que indicam a latitude e longitude do ponto na projeção WGS84 e seus valores de bandas espectrais e IV, representados pela Modelagem InfluxDB na Figura 1.

Com o objetivo de assegurar uma comparação equitativa entre os bancos de dados, ambos foram implantados em *containers* Docker. O uso dos *containers* se justifica por

[1] `https://github.com/vanessavcos/IC_CAFE_BDC/tree/main`

Figura 1. Modelagem dos dados espaço-temporais

sua capacidade de oferecer ambientes isolados, padronizados e facilmente replicáveis, garantindo condições justas e consistentes para a execução dos testes [Tripathi et al. 2023].

As métricas consideradas foram a latência média e o *throughput* e foram definidas conforme os trabalhos [Kim et al. 2023] e [Tripathi et al. 2023]. Além disso, a avaliação das consultas foi conduzida com o aumento progressivo do número de usuários simultâneos, com o objetivo de identificar o impacto dessa variação no desempenho de cada banco de dados.

Os testes foram implementados em Python versão 3.13.3 e executados em um notebook VAIO com processador AMD Ryzen 5 5500U com Radeon Graphics (2.1 GHz), 16GB de memória RAM e 237GB de armazenamento. Os códigos desenvolvidos para o estudo estão disponíveis em repositório no GitHub[2].

O fluxo para execução dos testes foi dividido em três etapas principais. A primeira consistiu em uma análise preliminar dos *containers* Docker utilizados para cada SGBD, avaliando o tamanho da imagem, bem como o consumo inicial de memória e CPU, conforme descrito no Cenário 1 da Tabela 1.

Na segunda etapa, realizou-se o teste de carga, simulando dois momentos distintos de inserção de dados, representado pelo Cenário 2 da Tabela 1. A primeira carga representa o cenário em que o satélite passa pela região de estudo pela primeira vez, capturando os dados espectrais e armazenando-os no banco. Já a segunda carga simula uma nova passagem do satélite sobre a mesma área, inserindo os dados mais recentes, reproduzindo o comportamento contínuo das séries temporais espaciais. Nesse teste, foi medida a latência de inserção para um único usuário, representando um processo automatizado de ingestão de dados.

Por fim, foi realizado o teste de consulta, com o objetivo de avaliar o desempenho em operações comuns em uma aplicação voltada ao monitoramento da cafeicultura. A primeira consulta está representada pelo Cenário 3 da Tabela 1, e consistiu em recuperar toda a série temporal dos valores das bandas espectrais e índices de vegetação para um pixel específico, dada sua latitude e longitude.

No MongoDB, o Cenário 3 - Teste 1 da Tabela 1 foi executado a partir das coordenadas geográficas (latitude e longitude), acessando diretamente o atributo *metadata.coordinates*, conforme definido na modelagem ilustrada na Figura 1.

No InfluxDB, essa consulta foi implementada de duas maneiras, denominadas

[2]https://github.com/vanessavcos/IC_CAFE_BDC/tree/main

Abordagem 1 e Abordagem 2. Na primeira abordagem, correspondente ao Cenário 3 - Teste 2 da Tabela 1, o filtro é aplicado diretamente sobre os campos de latitude e longitude, conforme ilustrado na consulta InfluxDB – A1.

No entanto, como a extensão geoespacial do InfluxDB utiliza S2CellId como identificador único de células geográficas (armazenado como *tag*), na segunda abordagem, é realizada uma conversão das coordenadas geográficas para esse identificador, correspondente ao Cenário 3 - Teste 3 da Tabela 1. Desta forma, o filtro é aplicado diretamente sobre a *tag* contendo o S2CellId, como demonstrado na consulta (InfluxDB – A2).

1. InfluxDB - A1

```
flux_query = f'''
from(bucket: "{bucket}")
    |> range(start: 0)
    |> filter(fn: (r) => r._measurement == "total")
    |> pivot(rowKey: ["_time"], columnKey: ["_field"],
    valueColumn: "_value")
    |> filter(fn: (r) => r.lat == {lat} and r.lon == {lon})
'''
```

2. InfluxDB - A2

```
s2_token = get_s2_cell_token(lat, lon, level=30)
flux_query = f'''
from(bucket: "{bucket}")
    |> range(start: 0)
    |> filter(fn: (r) => r._measurement == "total")
    |> filter(fn: (r) => r.s2_cell_id == "{s2_token}")
'''
```

Tabela 1. Plano de Execução dos Testes de Benchmark

Cenário	Teste	SGBD	Descrição
1- Análise Docker	1	MongoDB	Análise dos containers: imagem, memória e CPU iniciais.
	2	InfluxDB	
2 - Carga Inicial e Incremental	1	MongoDB	Carga inicial e incremental de dados, usando Modelagem MongoDB Fig. 1.
	2	InfluxDB	Carga inicial e incremental de dados, usando Modelagem InfluxDB Fig. 1.
3 - Consulta por pixel	1	MongoDB	Dada uma coordenada (lat/long), consultar a série temporal de todas as bandas para um pixel no MongoDB.
	2	InfluxDB	Dada uma coordenada (lat/long), consultar a série temporal de todas as bandas para um pixel utilizando a InfluxDB - A1.
	3	InfluxDB	Dada uma coordenada (lat/long), consultar a série temporal de todas as bandas para um pixel utilizando a InfluxDB - A2.
4 - Consulta espacial	1	MongoDB	Dada uma coordenada (lat/long) e um raio de 1,5 km, recuperar os valores das bandas em uma data específica para todos os píxeis da região.
	2	InfluxDB	

A segunda consulta, consistiu na recuperação dos valores de bandas em uma data determinada para um conjunto de pontos geograficamente próximos, representado pelo Cenário 4 da Tabela 1. Em ambos os SGBDs, MongoDB e InfluxDB, utilizou-se a funcionalidade de consulta espacial, na qual um ponto central é definido junto a um raio para compor uma região (*buffer*). Neste caso, 1.5 km, retornando todos os píxeis contidos nesta área.

Para realizar a segunda consulta no MongoDB, foi utilizada a função *$geoNear* que é usada para realizar consultas espaciais, retornando documentos próximos a um ponto geográfico. A função exige um índice geoespacial do tipo *2dsphere*, que permite

calcular distâncias esféricas reais. O *$geoNear* ordena os documentos por proximidade ao ponto de referência e calcula a distância até cada documento, facilitando buscas por raio de maneira eficiente [Tugores and Colet 2014].

Já no InfluxDB, foi utilizada a função *geo.filterRows()* que realiza filtragens espaciais em consultas sobre dados geográficos. A função combina duas operações internas: *geo.gridFilter()*, que seleciona séries de dados cujas células S2 (s2_cell_id) intersectam uma região geográfica especificada, e *geo.strictFilter()*, que refina o resultado considerando as coordenadas exatas (latitude e longitude) dos pontos. Para funcionar corretamente, os dados devem conter a *tag* s2_cell_id e os campos de coordenadas, permitindo identificar os pontos dentro de um raio ou polígono [InfluxDB Geo Package].

Para as consultas, foram coletadas a latência e o *throughput*, variando-se a quantidade de usuários simultâneos. Cada teste de consulta foi repetido 30 vezes, utilizando coordenadas distintas a cada execução, representando pontos diferentes dentro da área de estudo. Dessa forma, foi possível simular cenários de acesso concorrente em aplicações de análise geoespacial utilizando dados reais. Ao final, foi calculada a média da latência e do *throughput* para cada teste, permitindo uma comparação consistente entre os SGBDs sob diferentes consultas.

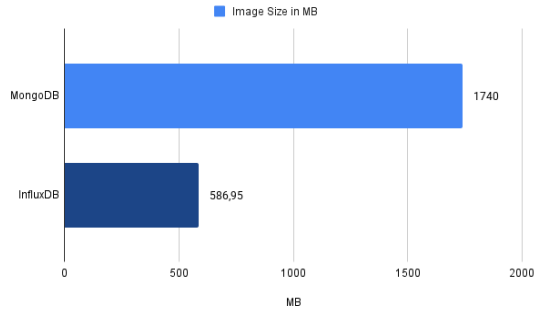
4. Resultados e Discussão

Inicialmente, são apresentados os resultados do Cenário 1 da Tabela 1 que consiste na análise preliminar dos *containers* Docker de cada SGBD, conforme ilustrado na Figura 2. Observa-se que o InfluxDB possui uma imagem Docker menor e consome menos memória e CPU quando em repouso, em comparação ao MongoDB. Esses resultados sugerem que o InfluxDB é uma solução mais leve em termos de uso de recursos computacionais. No entanto, é importante destacar que essa análise refere-se apenas ao estado inicial dos *containers*, sem considerar operações de escrita ou leitura nos bancos de dados.

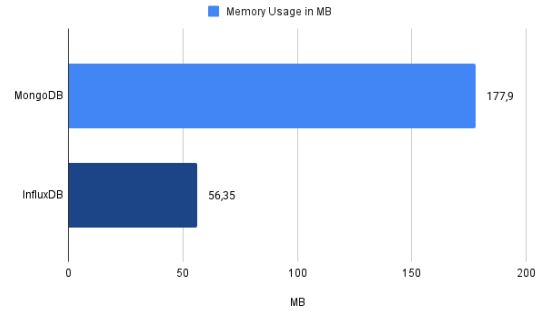
A Figura 3 apresenta os resultados de desempenho do Cenário 2 da Tabela 1, para as cargas inicial e de atualização nos bancos MongoDB e InfluxDB. Conforme mencionado anteriormente, ambas as cargas inseriram 12.784 registros cada, simulando o carregamento incremental dos dados ao longo do tempo, totalizando 2.403.392 registros. O MongoDB apresentou desempenho significativamente superior, sendo aproximadamente 7,11 vezes mais rápido na carga inicial e 7,34 vezes na carga de atualização. Apesar das diferenças na modelagem, a quantidade de dados inseridos em ambos os bancos é equivalente.

O menor desempenho do InfluxDB pode ser atribuído à sua arquitetura interna e ao modelo de dados utilizado no experimento. No InfluxDB, cada combinação única de *measurement, tag e field* gera uma nova *series key*. Como foi utilizada uma *tag* baseada no S2 Cell ID, cada ponto geográfico resultou em uma nova *serie key*. Além disso, cada *tag* possuía seis *fields*, o que multiplicou ainda mais a quantidade de *series*, elevando significativamente a cardinalidade do banco [InfluxDB Documentation].

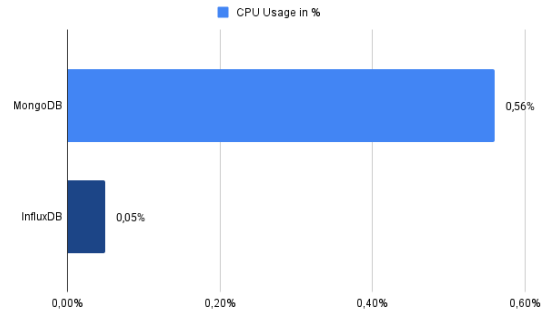
Essas *series keys* são importantes na estrutura de armazenamento do InfluxDB, que utiliza o formato Time-Structured Merge Tree (TSM). Nesse formato, os dados são organizados por *serie*, em colunas, e ordenados temporalmente. Cada *serie* é armazenada em blocos separados dentro dos arquivos TSM. Assim, um número elevado de séries



(a) Tamanho da imagem



(b) Uso de memória inicial



(c) Uso de CPU inicial

Figura 2. Análise do ambiente Docker

implica na criação de muitos blocos, o que aumenta o custo de serialização, escrita e gerenciamento em disco. Além disso, cada série gera um índice adicional Time Series Index(TSI), que adiciona ainda mais sobrecarga ao sistema [InfluxDB Documentation].

Por outro lado, o MongoDB, ao utilizar coleções do tipo *time series*, agrupa documentos com o mesmo *metafield* em *buckets* ordenados por tempo [Mongodb Storage]. Nesse caso, como a modelagem utilizou a coordenada como *metafield*, a única complexidade é o agrupamento de documentos pela coordenada, resultando em uma estrutura de armazenamento mais simples e eficiente do que a do InfluxDB.

Embora o InfluxDB seja otimizado para séries temporais, seu desempenho de escrita pode ter sido comprometido pela estrutura de dados necessária para utilizar o pacote geoespacial do Flux, que exigiu os campos de latitude e longitude como *fields*, além dos outros 4 campos *fields* de bandas espectrais e IV por registro.

Os resultados do Cenário 3 da Tabela 1, apresentados na Tabela 2, demonstram que o MongoDB obteve melhor desempenho em comparação ao InfluxDB Abordagem 1 em todos os cenários com diferentes números de *threads*. Destaca-se que essa primeira estrutura de consulta do InfluxDB sobrecarregou o banco nas execuções com 8, 16 e 32 *threads*. Por outro lado, o InfluxDB Abordagem 2 apresentou desempenho superior em relação às outras duas abordagens em todos os casos. Em especial, com uma única *thread*, a latência da consulta InfluxDB - A2 foi 22,9 vezes menor que a do MongoDB e 1.492,6 vezes mais rápida que a da estrutura InfluxDB - A1, evidenciando a vantagem do uso do identificador espacial S2CellId para consultas geográficas nesse banco.

O menor desempenho da estrutura de consulta do InfluxDB — A1 na consulta da

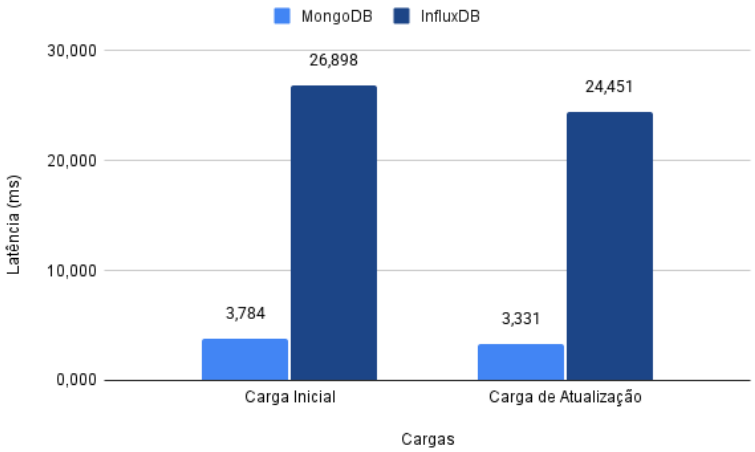


Figura 3. Teste de Carga Inicial e de Atualização

Tabela 2. Média de 30 consultas por pixel - Cenário 3

Threads	Consulta	Latência (s)	Throughput (ops/s)
1	MongoDB	2,06	0,49
	InfluxDB - T1	134,34	0,00744
	InfluxDB - T2	0,09	11,17
4	MongoDB	2,35	0,43
	InfluxDB - T1	175,36	0,0057
	InfluxDB - T2	0,10	9,54
8	MongoDB	3,13	0,32
	InfluxDB - T1	Timeout	Timeout
	InfluxDB - T2	0,16	6,31
16	MongoDB	5,63	0,18
	InfluxDB - T1	Timeout	Timeout
	InfluxDB - T2	0,30	3,37
32	MongoDB	10,36	0,1
	InfluxDB - T1	Timeout	Timeout
	InfluxDB - T2	0,42	2,36

Tabela 2, deve-se à forma como o banco organiza seus dados internamente: cada *field* é armazenado como uma coluna separada no armazenamento físico, isolada por série. No entanto, na saída padrão das consultas do Flux, os valores dos *fields* são apresentados em linhas separadas, com o nome do campo indicado em uma coluna (geralmente chamada "*_field*"). Como latitude e longitude são *fields*, seus valores aparecem em linhas diferentes, ainda que se refiram ao mesmo instante de tempo.

Para aplicar filtros que dependem simultaneamente de valores em múltiplos *fields*, como r.lat == x and r.lon == y, é necessário utilizar a função *pivot()*, que transforma os dados agregando os *fields* como colunas. Essa reorganização cria uma linha única por *timestamp* (ou outro critério definido), contendo as colunas lat e lon, o que permite aplicar filtros diretos e compostos. Contudo, essa transformação impacta o desempenho, pois exige reestruturação completa da tabela em memória, convertendo os dados em uma nova representação tabular com todos os *fields* como colunas. Esse processo torna a consulta

mais pesada e menos eficiente [InfluxDB - pivot].

Já o InfluxDB – A2, apresentou melhor desempenho em relação aos outros modelos possivelmente devido ao uso do identificador geográfico s2_cell_id como *tag*. No InfluxDB, as *tags* são indexadas pelo mecanismo Time Series Index (TSI), o que permite que consultas filtrando por s2_cell_id acessem diretamente a série correspondente via índice, lendo apenas o bloco necessário nos arquivos TSM (Time-Structured Merge Tree). Esses arquivos armazenam os dados organizados por *series key* (combinação de *measurement, tags e fields*), agrupando todos os registros com o mesmo s2_cell_id de forma sequencial no disco. Isso torna possível recuperar rapidamente todos os dados relacionados àquela *tag* específica, sem a necessidade de varrer outras séries. Portanto, o uso de *tags* para representar identificadores espaciais únicos, como o s2_cell_id, pode favorecer o desempenho em cenários com alta cardinalidade e consultas frequentes por localização.

O MongoDB apresentou um desempenho melhor que o do InfluxDB - A1, porém menor que o InfluxDB - A2. O bom desempenho do MongoDB nas consultas por pixel pode ser atribuído ao uso eficiente do *metaField*. No experimento, a coordenada geográfica foi usada como *metaField*, o que fez com que todos os registros de um mesmo ponto fossem agrupados no mesmo *bucket*. Além disso, o MongoDB cria automaticamente um índice composto entre *metaField* e *timeField*, otimizando a busca por séries temporais específicas [Mongodb Storage]. Assim, ao realizar a consulta, o banco acessa diretamente o *bucket* correspondente e retorna os dados ordenados temporalmente.

Dessa forma, entende-se que o InfluxDB, ao utilizar a Abordagem 2 representada na Tabela 2, oferece maior eficiência na recuperação de séries temporais completas para um determinado pixel.

Tabela 3. Média de 30 consultas por região - Cenário 4

Threads	Consulta	Latência (s)	Throughput (ops/s)
1	MongoDB	0,1467	6,817
	InfluxDB	16,663	0,060
4	MongoDB	0,1678	5,960
	InfluxDB	16,255	0,0615
8	MongoDB	0,2149	4,653
	InfluxDB	25,459	0,0392
16	MongoDB	0,3471	2,881
	InfluxDB	61,66	0,0162
32	MongoDB	0,6388	1,565
	InfluxDB	Timeout	Timeout

Já os resultados do Cenário 4 da Tabela 1, apresentados pela Tabela 3, mostram que o MongoDB apresentou desempenho superior ao InfluxDB, principalmente em razão da forma como gerencia coordenadas geográficas e realiza a indexação. Utilizando o operador $geoNear, o MongoDB realiza a busca diretamente sobre o campo de coordenadas com o auxílio de um índice geoespacial 2dsphere. Esse índice permite calcular distâncias de maneira eficiente e retorna apenas os documentos dentro do raio especificado, já ordenados por proximidade, que só depois serão filtrados por uma data específica. Além disso, como os dados estão organizados em *buckets* temporais por coordenada (*metaField*), as

consultas geotemporais se beneficiam de um layout físico otimizado no disco.

Em contraste, o InfluxDB utilizou a função *geo.filterRows()* para realizar a mesma consulta espacial. Embora funcionalmente equivalente, essa abordagem introduz uma sobrecarga computacional maior. A região consultada em latitude e longitude precisa primeiro ser convertida em um conjunto de células S2, que são utilizadas para selecionar séries com base em interseções espaciais. Em seguida, o filtro é refinado pelas coordenadas reais dos pontos [InfluxDB Geo Package]. Esse processo implica em múltiplas etapas de verificação e exige a leitura de diversos blocos físicos do armazenamento (arquivos TSM), já que os dados estão particionados por s2_cell_id, e não por proximidade geográfica direta. Como consequência, mesmo com a filtragem espacial aplicada, há maior volume de dados lidos e maior custo de I/O. Assim, embora o InfluxDB tenha ótima performance para consultas temporais, neste caso específico, que envolve uma filtragem espacial sobre um conjunto de pontos com alta cardinalidade geográfica, o MongoDB demonstrou ser mais eficiente.

5. Conclusão

Este trabalho avaliou o desempenho dos bancos de dados MongoDB e InfluxDB em operações de carga e consulta sobre séries espaço-temporais, com foco em cenários aplicados à cafeicultura. Para isso, foram conduzidos testes práticos utilizando dados reais provenientes do Brazil Data Cube.

Os resultados indicaram que o MongoDB apresentou melhor desempenho nas operações de inserção, tanto inicial quanto incremental. Já o InfluxDB se destacou nas consultas pontuais, voltadas à recuperação da série temporal de um único pixel, possivelmente em razão de sua arquitetura orientada ao tempo. Por outro lado, o MongoDB foi superior nas consultas espaciais sobre regiões delimitadas, especialmente em contextos com alta concorrência, nos quais o InfluxDB apresentou falhas de execução.

Com base nesses resultados, pode-se considerar que o InfluxDB é mais indicado para aplicações centradas em consultas temporais localizadas, enquanto o MongoDB se mostra mais eficiente para cargas volumosas e análises espaciais regionais. Isso reforça a importância de alinhar a escolha da tecnologia ao perfil de uso: aplicações com alta demanda de ingestão e consultas espaciais podem se beneficiar mais do MongoDB, ao passo que aplicações voltadas a consultas que envolvem píxeis específicos em séries temporais tendem a obter melhor desempenho com o InfluxDB.

Este estudo contribui com evidências práticas sobre a persistência e consulta de dados espaço-temporais em SGBDs que oferecem suporte nativo ou adaptado a essas características, fornecendo subsídios técnicos relevantes para decisões em projetos baseados em dados de sensoriamento remoto.

Como continuidade desta pesquisa, futuros trabalhos podem considerar a inclusão de outros SGBDs com suporte a dados espaço-temporais, bem como a exploração de novos tipos de consulta e métricas de *benchmark*, aprofundando a análise de desempenho e escalabilidade nesse domínio.

6. Agradecimentos

Agradeço ao CNPq pela concessão da bolsa de Iniciação Científica e à UNIFEI pelo apoio institucional e pelos recursos oferecidos para a realização desta pesquisa.

Referências

Bassoi, L. H., Inamasu, R. Y., Bernardi, A. C. d. C., Vaz, C. M. P., Speranza, E. A., and Cruvinel, P. E. (2019). Agricultura de precisão e agricultura digital. *TECCOGS: Revista Digital de Tecnologias Cognitivas*, (20).

Batina, A. (2023). Data Cubes – A Modern Approach for Handling Earth Observation Data. In *2023 International Conference on Earth Observation and Geo-Spatial Information (ICEOGI)*, pages 1–6.

Choi, W. G., Kim, S., Kim, J., Song, M.-H., and Lee, S.-S. (2022). Real-Time Data Processing Framework for Things with time-series and spatial features. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1694–1696. ISSN: 2162-1241.

CnosDB Documentation. Introduction | CnosDB.

Colosi, M., Martella, F., Parrino, G., Celesti, A., Fazio, M., and Villari, M. (2022). Time Series Data Management Optimized for Smart City Policy Decision. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 585–594.

DB-Engines. DB-Engines Ranking.

Formaggio, A. R. and Sanches, I. D. (2017). *Sensoriamento remoto em Agricultura*. Oficina de Textos. Google-Books-ID: hk88DwAAQBAJ.

Hachimi, C. E., Belaqziz, S., Khabba, S., Sebbar, B., Dhiba, D., and Chehbouni, A. (2023). Smart Weather Data Management Based on Artificial Intelligence and Big Data Analytics for Precision Agriculture. *Agriculture*, 13(1):95. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

InfluxDB - pivot. pivot() function | Flux Documentation.

InfluxDB Documentation. InfluxDB OSS v2 Documentation.

InfluxDB Geo Package. Experimental geo package | Flux Documentation.

John, P., Hynek, J., Hruška, T., and Valný, M. (2023). Application of Time Series Database for IoT Smart City Platform. In *2023 Smart City Symposium Prague (SCSP)*, pages 1–6. ISSN: 2691-3666.

Kim, S., Hoang, Y., Yu, T. T., and Kanwar, Y. S. (2023). GeoYCSB: A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial NoSQL Databases. *Big Data Research*, 31:100368.

Makris, A., Tserpes, K., Spiliopoulos, G., and Anagnostopoulos, D. (2019). Performance Evaluation of MongoDB and PostgreSQL for spatio-temporal data.

Mehmood, N. Q., Culmone, R., and Mostarda, L. (2017). Modeling temporal aspects of sensor data for MongoDB NoSQL database. *Journal of Big Data*, 4(1):8.

MongoDB Documentation. MongoDB Documentation.

Mongodb Storage. Time Series - Database Manual v8.0 - MongoDB Docs.

OpenTSDB Documentation. OpenTSDB - A Distributed, Scalable Monitoring System.

Petre, I., Boncea, R., Radulescu, C. Z., Zamfiroiu, A., and Sandu, I. (2019). A Time-Series Database Analysis Based on a Multi-attribute Maturity Model. *Studies in Informatics and Control*, 28(2).

Queiroz, D. M. d., Valente, D. S. M., Pinto, F. d. A. d. C., and Borém, A. (2022). *Agricultura digital*. Oficina de Textos. Google-Books-ID: 9ehvEAAAQBAJ.

Queiroz, G. R. D., Monteiro, A. M. V., and Câmara, G. (2013). BANCOS DE DADOS GEOGRÁFICOS E SISTEMAS NOSQL: ONDE ESTAMOS E PARA ONDE VAMOS. *Revista Brasileira de Cartografia*, 65(3).

S2 Documentation. Documentação s2 cell.

Sousa, G.; Leandro, M. F. H. (2023). O papel da cafeicultura no município de três pontas (mg). *[s.l: s.n.]*.

Tripathi, P., Miraz, M. H., and Joshi, S. (2023). Comparative Analysis of MongoDB and InfluxDB for Time Series Data Management in IoT Environments: A Study on Performance, Scalability, and Concurrency. In *2023 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA)*, pages 39–42.

Tugores, A. and Colet, P. (2014). Mining online social networks with Python to study urban mobility. arXiv:1404.6966 [cs].

Zaglia, M., Vinhas, L., Queiroz, G., and Simões, R. (2019). Catalogação de Metadados do Cubo de Dados do Brasil com o SpatioTemporal Asset Catalog. pages 280–285, São José dos Campos, SP, Brazil.

Zehra, S. N. (2017). Time Series Databases and InfluxDB.

Zhou, Y., De, S., Wang, W., Moessner, K., and Palaniswami, M. S. (2017). Spatial Indexing for Data Searching in Mobile Sensing Environments. *Sensors*, 17(6):1427. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.