Uma Arquitetura para Big Data Geoespacial de Dispositivos IoT com Níveis Heterogêneos de Mobilidade

Dennis Sávio Silva[1,2], **Maristela Holanda**[1]

[1] Departamento de Ciência da Computação, Universidade de Brasília, Brasília, Brasil

[2]Universidade Federal do Piauí, Picos, Piauí, Brasil

dennis.silva@ufpi.edu.br, mholanda@unb.br

Abstract. *The growth of geospatial data in internet of things raises questions about the influence of device mobility on storage strategies. This work proposes an approach for geospatial big data from IoT devices with heterogeneous mobility levels, with two main contributions: i) a survey of the key technologies used for storing and processing geospatial big data, highlighting their characteristics and functionalities; and ii) a three-layer architecture for storing data from entities with varying mobility levels. A prototype of the architecture was implemented, and the data storage and analysis layer was tested in three scenarios simulating different levels of mobility.*

Resumo. *O aumento dos dados geoespaciais em internet das coisas levanta questões sobre a influência da mobilidade dos dispositivos nas estratégias de armazenamento. Este trabalho propõe uma abordagem para big data geoespacial de dispositivos IoT com níveis heterogêneos de mobilidade, com duas contribuições principais: i) um levantamento das principais tecnologias empregadas no armazenamento e processamento de big data geoespacial, com suas características e funcionalidades; e ii) uma arquitetura em três camadas para o armazenamento de dados de dispositivos com diferentes níveis de mobilidade. Um protótipo da arquitetura foi implementado, e a camada de armazenamento e análise de dados foi testada em três cenários simulando níveis distintos de mobilidade.*

1. Introdução

Com a popularização dos dispositivos de Internet das Coisas (*Internet of Things* - IoT) com suporte à geolocalização, big data tem se consolidado como uma área cada vez mais influenciada por dados geoespaciais, cuja riqueza informacional potencializa a geração de conhecimento em múltiplas áreas de aplicação [al Jawarneh et al. 2020]. Na literatura é possível encontrar diversas aplicações contendo dados geoespaciais, cada uma contemplando objetos IoT com diferentes níveis de mobilidade [Li et al. 2020, Silva e Holanda 2022]. Neste artigo, o nível de mobilidade é definido como a frequência com que os dispositivos alteram sua informação de localização ao longo do tempo – ou seja, quanto mais alterações na localização geográfica, maior a mobilidade [Dai e Lu 2011].

Uma aplicação IoT pode ter dispositivos com diferentes níveis de mobilidade. Por exemplo, uma aplicação contendo dispositivos heterogêneos poderia, em um cenário, armazenar dados de dispositivos móveis, como veículos e *smartphones*, com registro da

localização dos usuários. Em outro, a aplicação poderia ser utilizada no gerenciamento de dispositivos estacionários ou cuja localização raramente se altera ao longo do tempo, como sensores de temperatura, umidade ou estacionamento. Em um terceiro cenário a aplicação pode envolver dispositivos móveis e estacionários. Considerar a mobilidade dos dispositivos gerenciados por uma aplicação pode contribuir para a melhoria do desempenho das operações espaciais, além de possibilitar um uso mais eficiente dos recursos de armazenamento.

Diante desse contexto, o objetivo deste trabalho é propor uma arquitetura para o armazenamento e a consulta de grandes volumes de dados geoespaciais provenientes de dispositivos IoT com variados níveis de mobilidade. A metodologia foi composta das seguintes fases: i) foi realizado um levantamento dos recursos oferecidos pelas ferramentas de gerenciamento e análise de big data geoespacial mais citados na literatura relacionada a IoT; ii) com base nesse levantamento, propôs-se uma arquitetura em três camadas para o gerenciamento e a análise desses dados; iii) foi desenvolvido, como prova de conceito, um protótipo da camada de armazenamento e análise de dados da arquitetura proposta, utilizando duas configurações de ferramentas de armazenamento e análise geoespacial; iv) por fim, uma avaliação da camada de armazenamento e análise foi conduzida por meio da simulação de três contextos de uso, contemplando diferentes níveis de mobilidade, e da análise de desempenho de consultas nas duas configurações.

O restante do trabalho está organizado da seguinte forma: a Seção 2 traz os trabalhos relacionados, a Seção 3 apresenta um levantamento das ferramentas mais presentes na literatura, tanto para armazenamento quanto para análise de big data geoespacial. A Seção 4 descreve a arquitetura proposta. A Seção 5 descreve os experimentos realizados, a Seção 6 apresenta uma análise dos resultados obtidos e, por fim, a Seção 7 apresenta a conclusão do trabalho.

2. Trabalhos Relacionados

Muitos trabalhos na literatura apontam aplicações de IoT que lidam tanto com dados geoespaciais provenientes de dispositivos estacionários quanto de informações relacionadas a trajetórias de dispositivos móveis [Silva e Holanda 2022]. A seguir, algum desses trabalhos são apresentados.

Fernández et al. [Fernández et al. 2018] apresentam uma implementação de *smart port* que manipula dados heterogêneos, oriundos tanto de sensores estacionários quanto sensores móveis presentes nas embarcações. Alvarez et al. [Alvarez et al. 2019] e Sukmaningsih et al. [Sukmaningsih et al. 2020] apresentam soluções para *smart cities* que utilizam sensores móveis e estacionários. Finogeev et al. em [Finogeev et al. 2019] realizam monitoramento de incidentes de tráfego, utilizando tanto dados de foto-radares quanto de dispositivos móveis. Baig et al. em [Baig et al. 2021] apresentam o SPEAR, um *framework* para processamento de *streams* espaço-temporais, e propõe um conjunto generalizado de tipos de dados espaço-temporais estacionários e móveis, e funções de *range* e *nearest neighbor* com suporte para ajuste dinâmico para objetos móveis.

Wang et al. apresentam em [Wang et al. 2023] um sistema para reduzir a sobrecarga na alocação de recursos de recarga para uma frota de mais de 13.000 táxis elétricos (ETaxi) na cidade de Shenzhen, na China. Além dos registros geolocalizados dos ETaxis, são armazenados metadados de 117 estações de recarga. Wang et al. propõem em

[Wang et al. 2024] um método de preenchimento de dados espaço-temporais de *Mobile CrowdSensing* em grande escala, realizando análises em dois *datasets*: o U-Air, com dados de qualidade do ar oriundos de dispositivos móveis geolocalizados, e o Sensor-Scope, com dados de sensores estacionários.

Mesmo em trabalhos reportando aplicações que gerenciam dispositivos estacionários e móveis, não foram encontradas abordagens considerando o impacto dos diferentes níveis de mobilidade dos dispositivos na eficiência dos mecanismos de armazenamento e consulta a dados geoespaciais. Neste trabalho, é proposta uma arquitetura que abrange tanto contextos com dispositivos estacionários quanto móveis, e considera questões referentes tanto ao armazenamento, como o consumo de espaço, quanto à análise, como o tempo de recuperação nas consultas, de big data geoespacial.

3. Levantamento de ferramentas para big data geoespacial em IoT

O interesse por soluções para o armazenamento e processamento de dados geoespaciais tem crescido ao longo do tempo, acompanhando a evolução das tecnologias de informação. Atualmente, a pesquisa geoespacial tem direcionado esforços para soluções de computação paralela e distribuída, como bancos de dados NoSQL, sistemas de arquivos distribuídos e *frameworks* de processamento de big data [al Jawarneh et al. 2020, Al-Yadumi et al. 2021, Breunig et al. 2020]. A escalabilidade proporcionada pela distribuição e replicação de dados e serviços pode aumentar a disponibilidade e elasticidade dos sistemas, permitindo a oferta de recursos computacionais para big data geoespacial com uma melhor relação custo-benefício [Al-Yadumi et al. 2021].

A Figura 1 ilustra algumas das plataformas *open-source* comumente citadas na literatura relacionada a IoT para o gerenciamento de big data geoespacial [Silva e Holanda 2022], oferecendo diversas opções para armazenamento, consultas e geovisualização. A figura também descreve a integração básica entre essas ferramentas, sendo que linhas contínuas representam ferramentas que compõem ou servem de base para o desenvolvimento de outras, e linhas tracejadas indicam plataformas que se relacionam a outras por meio de *plugins* ou outras formas de integração. As próximas subseções detalham as funcionalidades oferecidas por essas plataformas para o armazenamento e análise de big data geoespacial.

3.1. Armazenamento de big data geoespacial

Dentre os principais requisitos para SGBDs (Sistemas Gerenciadores de Banco de Dados) voltados a aplicações geoespaciais, destacam-se o custo de armazenamento, o desempenho das consultas, a flexibilidade dos esquemas, a capacidade de manipular grandes volumes de dados e a integração com ferramentas de análise [Azad et al. 2020]. Essas características colocam os bancos NoSQL em evidência, uma vez que são capazes de armazenar dados estruturados, semi-estruturados e não estruturados, além de utilizarem linguagens de consulta distintas do SQL tradicional [al Jawarneh et al. 2020]. Adicionalmente, sua eficiência, facilidade de distribuição e escalabilidade em *clusters* de alta disponibilidade e tolerância a falhas os tornam adequados para o suporte a *streams* de dados massivos [Lee et al. 2019]. No entanto, o suporte desses bancos a dados geoespaciais ainda é bastante limitado, uma vez que muitas estruturas de indexação e operações geoespaciais não foram originalmente projetadas para funcionar de forma eficiente em ambientes distribuídos [Shah 2019].

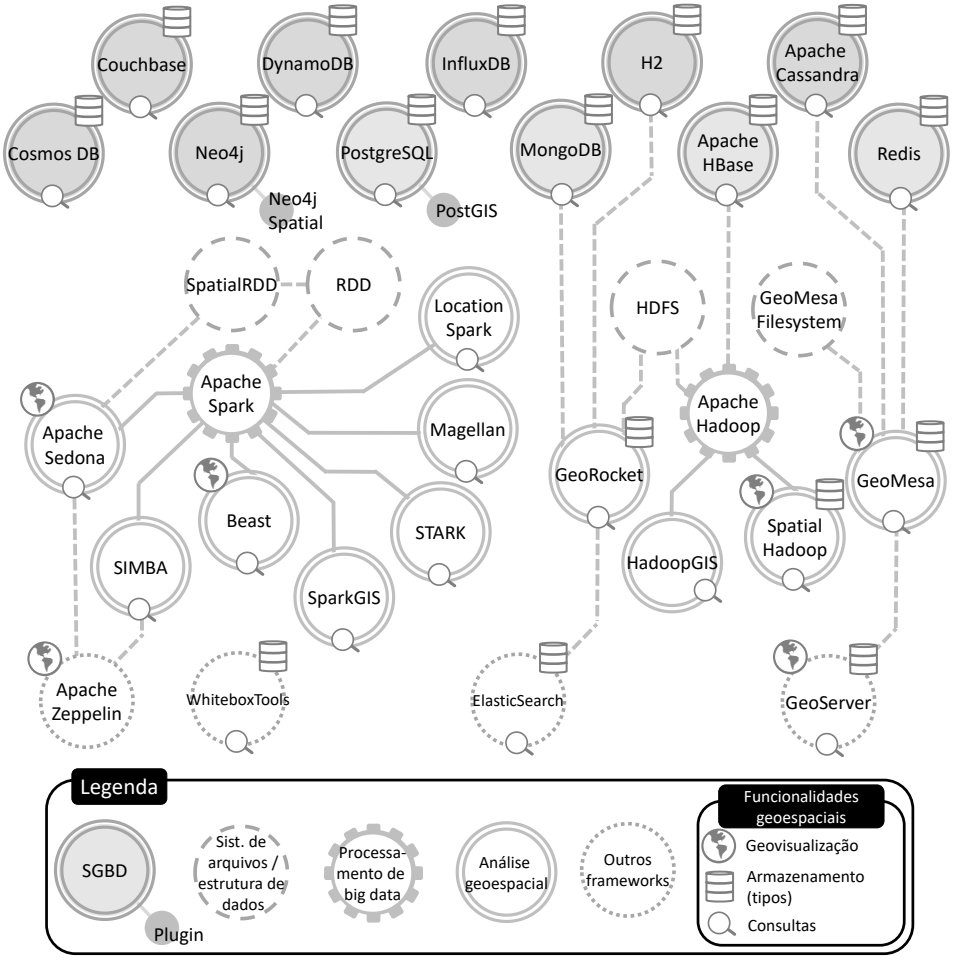

Figura 1. Plataformas para gerenciamento de big data geoespacial

A Tabela 1 apresenta alguns dos SGBDs mais populares em aplicações que armazenam e analisam big data geoespacial, juntamente com o seu modelo de dados predominante (relacional, documentos, colunas, chave-valor, grafos, séries temporais, ou multi-modelos) e seus principais recursos geoespaciais, como disponibilidade de indexação, linguagens de consulta e formatos de dados suportados.

3.2. Processamento de big data geoespacial

A Tabela 2 apresenta alguns dos *frameworks*, bibliotecas e ferramentas mais utilizados em aplicações que armazenam e analisam dados geoespaciais, acompanhados de um resumo de suas principais características, como o ano de criação (+) e da última atualização (*) do projeto, geometrias compatíveis, principais operadores e funções geoespaciais disponíveis, e recursos relacionados à geovisualização.

O GeoRocket é um *data store* de alto desempenho voltado para formatos de arquivo geoespaciais. Com ênfase em escalabilidade, ele suporta diferentes formas de armazenamento de dados, como sistemas de arquivos locais ou distribuídos, os SGBDs relacional H2 e NoSQL MongoDB, e o serviço de armazenamento em nuvem Amazon S3.

Os principais *frameworks* voltados à análise de dados massivos contam com ex-

Tabela 1. SGBDs massivos populares e seus recursos geoespaciais

SGBD	Classificação	Indexação	Linguagem de consulta	Formatos
PostgreSQL e PostGIS	Relacional	✔	SQL	(PostGIS) Importação: KML, GML, GeoJSON, GeoHash, WKT; Renderização: GeoTIFF, PNG, JPG, NetCDF.
MongoDB	Documentos	✔	MQL (MongoDB Query Language)	GeoJSON
Redis	Chave-valor	✔	Comandos Redis	GeoJSON
Cassandra	Colunas	✔	Cassandra Query Language (CQL)	WKT
DynamoDB	Multi-modelos	✔	Queries DynamoDB	GeoJSON
Neo4j e Neo4j Spatial	Grafos	✔	Cypher	Neo4j: Graph Model, RDF Neo4j Spatial: Shapefile, GeoJSON, OSM
Cosmos DB	Multi-modelos	✔	Azure Cosmos DB SQL API	GeoJSON
InfluxDB	Séries temporais	✔	InfluxQL	InfluxDB Line Protocol, JSON, Graphite
Couchbase	Documentos	✔	Couchbase REST API, N1QL	GeoJSON

tensões para suportar funções e tipos de dados espaciais. O Apache Hadoop[1], utilizado em conjunto com o HDFS (*Hadoop Distributed File System*), serve como base para o desenvolvimento de diversos *frameworks*, como o Hadoop-GIS[2], um sistema de *data warehousing* para dados espaciais que estende o Apache Hive; e o SpatialHadoop[3] (descontinuado). O Apache Spark[4] tem sido utilizado como base para *frameworks* de dados geoespaciais, com desempenho superior aos baseados no Hadoop [Pandey et al. 2018]. Alguns deles são: Magellan[5], biblioteca que estende o Spark SQL oferecendo uma abstração relacional para análise geoespacial; STARK[6], que adiciona suporte a tipos de dados e operações espaciais; SparkGIS[7], que é capaz de reescrever dinamicamente consultas para gerenciar cargas de trabalho que excedam os recursos distribuídos disponíveis; SIMBA[8] (*Spatial In-Memory Big data Analytics*), que incorpora índices espaciais em contextos de RDDs (*Resilient Distributed Datasets*) e SQL, além de possuir otimizadores físicos e lógicos para selecionar os melhores planos de execução de consultas; LocationSpark[9], que utiliza filtros em seus índices para evitar comunicação desnecessária na rede durante o processamento de dados espaciais; Beast[10] (*Big Exploratory Analytics for Spatio-temporal data*), atual projeto dos criadores do SpatialHadoop; e o Apache Sedona[11] (anteriormente conhecido como GeoSpark), que permite o processamento em *cluster* de dados espaciais massivos em memória, utilizando SRDDs (SpatialRDDs), uma extensão geoespacial dos RDDs do Spark, para particionamento espacial dos dados no *cluster* com base na proximidade geográfica.

[1]https://hadoop.apache.org/
[2]http://bmidb.cs.stonybrook.edu/hadoopgis/index
[3]https://github.com/aseldawy/spatialhadoop2
[4]https://spark.apache.org/
[5]https://github.com/harsha2010/magellan
[6]https://github.com/dbis-ilm/stark
[7]http://bmidb.cs.stonybrook.edu/sparkgis/index
[8]https://github.com/InitialDLab/Simba
[9]https://github.com/purduedb/LocationSpark
[10]https://bitbucket.org/bdlabucr/beast/src/master/
[11]https://sedona.apache.org/

Tabela 2. Ferramentas para análise de dados massivos geoespaciais

Ferramenta	Geometrias	Principais operadores e funções	Geovisualização
Apache Sedona **+2016 *2025**	Point, MultiPoint, LineString, Polygon, MultiLineString, MultiPolygon, GeometryCollection	Range, Range join, Distance join, KNN, Conversão entre Sistemas de Coordenadas de Referência	Geração de mapas em alta resolução via SedonaViz, com operadores para visualização de mapas e *map tiles*
Beast **+2021 *2024**	Point, MultiPoint, LineString, Polygon, MultiLineString, MultiPolygon, GeometryCollection	Spatial join	Visualização *single image* e multilevel para exploração interativa.
GeoRocket **+2020 *2024**	Point, MultiPoint, LineString, Polygon, MultiLineString, MultiPolygon, GeometryCollection	Via ElasticSearch	×
GeoMesa **+2015 *2025**	Point, MultiPoint, LineString, Polygon, MultiLineString, MultiPolygon, GeometryCollection	Via SparkSQL ou API GeoJSON	Integração com Leaflet e GeoServer.
HadoopGIS **+2013 *2013**	Point, Box, Polygon, LineString	Intersects, KNN, Contains, Touches, Area, Distance, funções agregadas, dentre outras	×
Spatial Hadoop **+2015 *2018**	Point, Rectangle, Polygon	Range queries, KNN, Spatial joins, dentre outras.	Duas operações: *plot* (gera mapas a partir de arquivos) e *plotp* (gera *map tiles*)
STARK **+2017 *2022**	Point, MultiPoint, LineString, Polygon, MultiLineString, MultiPolygon, GeometryCollection	Filter (intersects, contains, containedBy, withinDistance), joins, clustering (DBSCAN), kNN	×
SparkGIS **+2015 *2016**	Point, Line, MultiPoint, MultiLine, Polygon	Dimension, Area, Equals, Envelope, Length, Crosses, Centroid, Contains, Disjoint, Distance, Intersects, Within, Overlaps, Touches, Union, Boundary	×
Magellan **+2015 *2017**	Point, MultiPoint, Polygon, MultiPolygon, LineString	range queries, spatial joins	×
SIMBA **+2016 *2017**	Point, LineString, Polygon, Circle	range, kNN, Distance join, kNN join	Visualização interativa via Zeppelin (modificação)
Location Spark **+2016 *2017**	Point, Rectangle	Range, kNN, Spatial join, distance join, kNN join	×
Whitebox Tools **+2017 *2022**	Point, MultiPoint, PolyLine, Polygon, PolyLineZ, PolygonZ, MultiPointZ	Análise GIS, process. de imagens, análise hidrológica espacial, análise geomorfométrica e process. de dados LiDAR.	×

Outro destaque é o GeoMesa[12], um pacote de ferramentas voltado ao armazenamento, busca e análise de dados geoespaciais que utiliza como *backend* bancos como Accumulo, HBase, Google Bigtable ou Cassandra. O WhiteboxTools[13] é um pacote de análise geoespacial que disponibiliza mais de 450 ferramentas para o processamento de dados espaciais.

A geovisualização de dados é um componente fundamental de big data geoespacial, que permite explorar seu potencial visual por meio de mapas 2D, 3D e 4D,

[12]https://www.geomesa.org/

[13]https://www.whiteboxgeo.com/geospatial-software/

painéis interativos (*dashboards*) e aplicações em realidade virtual [Shrivastava 2020, Fernández et al. 2018, Lwin et al. 2019]. Plataformas como Spatial Hadoop, SIMBA, Beast, GeoMesa, e Apache Sedona oferecem funcionalidades de geovisualização.

4. Arquitetura Proposta

A Figura 2 ilustra, de forma simplificada, a arquitetura proposta para o armazenamento e a análise de big data geoespacial proveniente de dispositivos com diferentes níveis de mobilidade, e a combinação das ferramentas utilizadas em sua implementação. A arquitetura é composta por três camadas, que são descritas a seguir: Interface com o Ambiente Físico, Armazenamento e Análise de Dados e Aplicação.

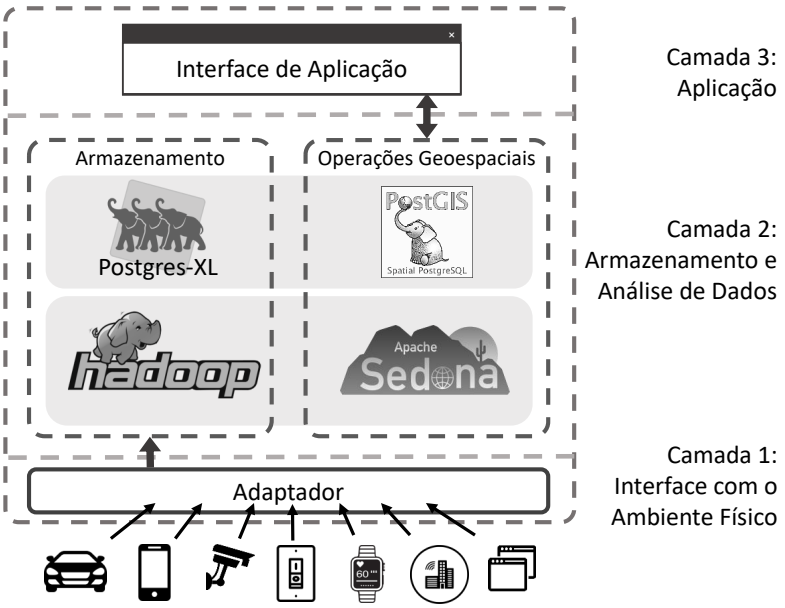

Figura 2. Arquitetura IoT proposta

A primeira camada, Interface com o Ambiente Físico, é responsável pela coleta de dados provenientes dos dispositivos distribuídos no ambiente. Esses dispositivos possuem informações de geolocalização e, no caso das dispositivos móveis, podem registrar múltiplas localizações ao longo do tempo. Além disso, elas são capazes de capturar diferentes observações do ambiente por meio de sensores. Os dados de localização e as observações são coletadas por um adaptador, convertidos para o modelo de dados da arquitetura e, em seguida, enviados para a camada seguinte.

A camada de Armazenamento e Análise de Dados tem como função armazenar, gerenciar e oferecer suporte à análise dos dados provenientes dos dispositivos da camada física. Para isso, contempla duas funcionalidades principais: armazenamento de dados e suporte a operações geoespaciais. O armazenamento dos dados massivos da aplicação requer mecanismos com capacidade para lidar com formatos de dados heterogêneos e capacidade de ingestão em tempo real de grandes volumes de dados. Os critérios adotados para a escolha das ferramentas incluíram o suporte ao processamento e armazenamento distribuído de dados geoespaciais em larga escala, a disponibilidade de linguagens de consulta com suporte a consultas não apenas espaciais, mas também sobre as observações registradas pelos dispositivos, a compatibilidade com outros artefatos de software voltados

ao armazenamento e análise de dados geoespaciais, a aplicação recorrente em pesquisas na área de big data geoespacial, e a disponibilidade de documentação adequada.

Dessa forma, foram consideradas as seguintes combinações de ferramentas para o armazenamento e processamento de dados geoespaciais, dentre as apresentadas na Seção 3: Postgres-XL (solução distribuída baseada no PostgreSQL, versão 10r1.1) em conjunto com PostGIS (versão 2.5.0); e HDFS (Hadoop 3.4.1) integrado ao Apache Sedona (versão 1.2.1). As funcionalidades de ingestão e consulta dos dados geoespaciais foram implementadas na linguagem de programação Java.

No nível superior da arquitetura, a camada de Aplicação atua como interface entre o usuário e os dados armazenados, permitindo a realização de consultas aos dados armazenados por meio de formulários e a visualização dos resultados obtidos. A implementação dessa camada requer o uso de *frameworks* de desenvolvimento web para a criação dos formulários de consulta, bem como ferramentas voltadas à apresentação dos resultados em *dashboards* espaciais e mapas interativos.

5. Experimentação

A experimentação conduzida neste trabalho teve como objetivo analisar o desempenho da camada de Armazenamento e Análise de Dados, verificando quais soluções apresentam melhor desempenho (menor tempo) na execução de consultas a dados de localização e observações diante de um grande volume de registros representando contextos caracterizados por diferentes níveis de mobilidade.

Foram geradas imagens Docker específicas para atender às necessidades de cada experimento, possibilitando a criação de *clusters* do Postgres-XL, Apache Hadoop (HDFS) e Apache Spark (para utilização do Apache Sedona). O ambiente de execução dos experimentos foi configurado com o uso do Docker Swarm, distribuído em 16 máquinas (nós) com processador Intel® Core™ i5-3470 CPU 2.90GHz (quad-core), 500 GB de armazenamento em HD, 4 GB de memória RAM e sistema Ubuntu 22.04. Os serviços empregados em cada combinação de ferramentas foram organizados da seguinte forma:

- **Configuração 1 – Postgres-XL e PostGIS**: composta de um nó mestre, para o serviço GTM *Master*, e quinze nós de trabalho, cada um contendo um serviço *Datanode* e um coordenador, cada um associado a um *proxy* GTM.
- **Configuração 2 – Apache Sedona e HDFS**: composta de um nó mestre, para o serviço Spark *Master* e o Hadoop *Namenode*, e quinze nós de trabalho, contendo um serviço de *Datanode* do Hadoop e um Spark *Worker* cada.

Com o objetivo de representar ambientes com diferentes níveis de mobilidade, foram realizadas simulações baseadas em dados reais, todos provenientes da cidade de Chicago (EUA). As rotas dos dispositivos móveis foram simuladas com base nos conjuntos de dados *Bike Routes* [Chicago Data Portal 2022] e *Chicago Campus Bus GPS Trajectory Data* [Ahmed et al. 2015]. Para simular as observações realizadas no ambiente, foram utilizados dados históricos do projeto *Array of Things* [Catlett et al. 2017] (AoT), sendo selecionados, para a experimentação, sensores de umidade, temperatura e pressão.

As simulações abrangem três cenários de mobilidade distintos. O primeiro envolve apenas dispositivos móveis, que geram informações de localização geoespacial em intervalos regulares. O segundo é composto apenas por dispositivos estacionários, com

localização geoespacial fixa, sendo que cada dispositivo possui um sensor que gera observações em intervalos regulares. O terceiro combina tanto dispositivos móveis quanto estacionários. Em todos os cenários, a cada dispositivo é atribuído um tipo de sensor, que gera observações em intervalos regulares. A cada atualização, o valor do sensor é ajustado com base nos valores dos sensores similares do *dataset* AoT. Para o cálculo do valor, foi utilizado o algoritmo IDW (*Inverse Distance Weighting*) [Burrough e McDonnell 2015], um método de interpolação espacial que estima valores desconhecidos com base nos pontos vizinhos, atribuindo pesos inversamente proporcionais à distância. Os *datasets* gerados simulam 3 anos de observações e possuem o mesmo volume de dados (200 GB), totalizando aproximadamente 850 milhões de registros cada.

Para avaliar o desempenho da arquitetura proposta, foram elaboradas doze consultas (Tabela 3), divididas em três categorias: quatro consultas com operadores espaciais básicos (`Range`, `Containment` e `Nearest Neighbor (NN)`) [Eldawy e Mokbel 2016] para os dados de localização, quatro consultas envolvendo dados de observações, com operações de filtragem por valor e agregações dos dispositivos, e quatro consultas mistas, que combinam os filtros de localização e de observações apresentados nas consultas anteriores. Para cada *dataset* gerado, as ferramentas avaliadas receberam os dados, e as consultas foram executadas dez vezes cada. Os resultados obtidos (tempos de execução máximo, mínimo, médio e desvio padrão) foram compilados e comparados, permitindo identificar quais ferramentas apresentaram os melhores resultados (menores tempos de execução) em diferentes cenários de mobilidade dos dispositivos, além de observar a relação entre registros de localização e de observações.

6. Resultados

Os tempos de execução das consultas para as duas ferramentas nos *datasets* de dados móveis, mistos e estacionários estão apresentados nas Tabelas 4, 5 e 6, respectivamente. De acordo com a análise do desvio padrão dos tempos de execução das consultas, o Postgres-XL apresentou uma maior variação: algumas consultas mostraram baixo desvio padrão, indicando tempos de execução consistentes, enquanto outras apresentaram alta variação, refletindo maior instabilidade. Em contraste, o Apache Sedona demonstrou desvios padrão mais uniformes entre as consultas, sugerindo um comportamento mais previsível e estável em relação ao tempo de resposta. O Postgres-XL inicialmente apresentou tempos de execução mais altos, mas estes diminuíram conforme as consultas eram repetidas, o que pode ser um fator decisivo na escolha do mecanismo de armazenamento, dependendo do perfil de execução das consultas pelos usuários. Na média dos tempos de execução, o Postgres-XL obteve melhor desempenho nas consultas 1, 3, 5, 6, 7, 8, 9, 10 e 11, enquanto o Apache Sedona teve melhor desempenho nas consultas 2, 4 e 12. O Postgres-XL se destacou nas consultas a dados de observação, mostrando-se mais eficiente para contagens simples com filtros em colunas específicas, pois distribui os cálculos entre os *datanodes* e realiza a agregação final nos coordenadores. Além disso, obteve um bom desempenho em sub-consultas e agregações como média (AVG) e menor valor (MIN).

Na Consulta 1, o Postgres-XL apresentou melhor desempenho em termos de menor tempo e tempo médio, apesar de, na variação, ter registrado o maior tempo de execução em uma das consultas. Nas médias, os *datasets* com dispositivos móveis apresentaram melhores resultados do que o *dataset* de dispositivos estacionários. Isso pode ser devido ao fato de, embora a busca envolvesse um atributo espacial, o filtro ter sido aplicado em

Tabela 3. Consultas

Consulta 1 - Localizações de um dispositivo entre os instantes y e z? `SELECT ST_AsText(location) FROM tese WHERE entityId = ? AND observation Time BETWEEN 'yyyy-MM-dd HH:mm:ss.SSS' AND 'yyyy-MM-dd HH:mm:ss.SSS'`
Consulta 2 - Dispositivos que estiveram na região delimitada por um retângulo. `SELECT DISTINCT entityId FROM tese WHERE ST_Within(location, ST_MakeEnvelope (x1, y1, x2, y2, 4326))`
Consulta 3 - Observações realizadas a até 600 metros de um ponto. `SELECT ST_AsText(location) AS location FROM tese WHERE ST_DWithin(location, ST_GeogFromText('SRID=4326;POINT(x, y)'), 600)`
Consulta 4 - Três observações mais próximas de determinado ponto (kNN). `SELECT idObservation, ST_AsText(location) AS location, ST_Distance(location, ref_geom) AS distance FROM tese, LATERAL ST_GeogFromText('SRID=4326; POINT (x, y)') AS ref_geom ORDER BY location <-> ref_geom LIMIT 3`
Consulta 5 - Valores foram coletados por determinado objeto em um intervalo de tempo? `SELECT COUNT(observationValue) FROM tese WHERE entityId = 0 AND observationTime BETWEEN 'yyyy-MM-dd HH:mm:ss.SSS' AND 'yyyy-MM-dd HH:mm:ss.SSS'`
Consulta 6 - Média dos valores para uma propriedade em um intervalo de tempo? `SELECT AVG(CAST(observationValue AS double precision)) AS media FROM tese WHERE entityId = 0 AND observationTime BETWEEN 'yyyy-MM-dd HH:mm:ss.SSS' AND 'yyyy-MM-dd HH:mm:ss.SSS' AND obsPropertyName = 'pressure'`
Consulta 7 - Menor registro de umidade `WITH min_humidity AS (SELECT MIN(CAST(observationValue AS double precision)) AS min_value FROM tese WHERE obspropertyname = 'humidity') SELECT observation value FROM tese, min_humidity WHERE obspropertyname = 'humidity' AND ABS(CAST(observationvalue AS double precision) - min_humidity.min_value) < 1e-8`
Consulta 8 - Registros de pressão superior a 2200 hPa `SELECT idRecord, observationValue FROM tese WHERE obspropertyname = 'pressure' AND CAST(observationvalue AS double precision) >= 2200`
Consulta 9 - Em que localização determinado objeto encontrou o maior valor para determinada propriedade observada? `WITH max_pressure AS (SELECT MAX(CAST(observationValue AS double precision)) AS max_value FROM tese WHERE entityid = 0 AND obspropertyname = 'pressure') SELECT ST_AsText(location) AS location FROM tese, max_pressure WHERE entityid = 0 AND obspropertyname = 'pressure' AND ABS(CAST(observationvalue AS double precision) - max_pressure.max_value) < 1e-8`
Consulta 10 - Localização das Observações de pressão que ultrapassaram 1020 hPa na região delimitada por um retângulo [x1,y1,x2,y2]. `SELECT ST_AsText(location) AS geolocation, observationvalue FROM tese WHERE entityid = 0 AND obspropertyname = 'pressure' AND CAST(observationvalue AS double precision) >= 1020 AND ST_Within(location, ST_MakeEnvelope(x1, y1, x2, y2, 4326))`
Consulta 11 - Registros de temperatura superior a 40ºC realizados a até 20 metros do ponto [x, y] (*range query*) `SELECT ST_AsText(location) AS location, observationValue FROM tese WHERE ST_DWithin(location, ST_GeogFromText('SRID=4326;POINT(x, y)'), 600) AND obspropertyname = 'temperature' AND CAST(observationvalue AS double precision) >= 40`
Consulta 12 - Três registros de umidade mais próximos a um determinado ponto (*kNN query*). `SELECT idObservation, ST_AsText(location) AS location, ST_Distance(location, ref_geom) AS distance, observationValue FROM tese, LATERAL ST_GeogFromText('SRID=4326; POINT(x, y)') AS ref_geom WHERE obspropertyname = 'humidity' AND CAST(observationvalue AS double precision) >= 20 ORDER BY location <-> ref_geom LIMIT 3`

um atributo inteiro e em um *timestamp*. As Consultas 2 e 10 empregam um filtro espacial `ST_Within`. Na Consulta 2 o Apache Sedona teve melhor desempenho em todos os contextos analisados, sendo que o Postgres-XL teve os piores tempos médios nos *datasets* de dispositivos móveis. Já na consulta 10, envolvendo além do filtro espacial um filtro de valor em um campo específico, o Postgres-XL se saiu melhor. As Consultas 3 e 11 empregam um filtro espacial `ST_DWithin`, e o Postgres-XL apresentou melhor desempenho para o menor tempo e tempo médio, apesar de, na variação, ter tido o pior registro de tempo de uma consulta. A Consulta 4 busca os três registros mais próximos de um ponto de referência. O Postgres-XL teve desempenho bastante inferior ao do Apache Sedona, devido a um *nested loop*. Já na consulta 12, apesar de o Apache Sedona ainda ter se saído melhor, os filtros não-espaciais empregados ajudaram a reduzir a diferença. Nas Consultas 5, 6, 7 e 8, todas baseadas em filtros e agregações empregando atributos

Tabela 4. Tempos de execução (ms) médio, máximo, mínimo e desvios padrão para o *dataset* de dispositivos móveis

	Postgres-XL + PostGIS				Sedona + HDFS			
	Média	**Maior**	**Menor**	**Desvio**	**Média**	**Maior**	**Menor**	**Desvio**
Consulta 1	216.249	756.733	136.378	186.917	466.036	492.361	452.451	10.283
Consulta 2	743.948	854.917	576.406	106.616	509.974	527.929	497.042	7.430
Consulta 3	283.268	1.505.927	134.135	408.838	506.021	529.394	485.509	16.289
Consulta 4	1.834.908	2.794.621	1.559.009	367.959	491.562	508.780	476.853	10.473
Consulta 5	168.256	276.621	155.337	36.132	473.132	492.326	455.818	10.968
Consulta 6	155.801	156.563	155.430	348	469.603	478.664	458.691	6.942
Consulta 7	386.179	405.988	378.376	7.642	958.898	986.443	930.356	18.097
Consulta 8	156.310	159.547	155.421	1.228	473.420	487.215	458.866	10.131
Consulta 9	338.475	383.666	330.231	15.822	938.790	964.474	889.545	21.156
Consulta 10	156.024	156.574	155.679	266	468.911	482.400	456.373	8.133
Consulta 11	156.008	156.611	155.507	332	488.110	508.210	477.188	10.486
Consulta 12	508.589	540.139	489.882	17.074	490.505	500.486	481.756	6.465

Tabela 5. Tempos de execução (ms) médio, máximo, mínimo e desvios padrão para o *dataset* de dispositivos mistos.

	Postgres-XL + PostGIS				Sedona + HDFS			
	Média	**Maior**	**Menor**	**Desvio**	**Média**	**Maior**	**Menor**	**Desvio**
Consulta 1	297.309	496.586	262.745	66.702	474.378	494.625	455.101	12.214
Consulta 2	665.963	986.804	542.244	131.176	512.968	533.266	483.647	14.087
Consulta 3	161.685	166.191	159.457	1.817	501.022	519.192	476.482	12.178
Consulta 4	1.585.416	1.738.275	537.211	56.018	510.207	520.640	492.189	8.180
Consulta 5	146.858	149.406	145.949	918	475.814	495.694	457.366	11.687
Consulta 6	147.119	149.316	145.906	973	479.758	502.727	450.649	14.318
Consulta 7	379.406	409.250	373.110	10.579	963.272	993.900	940.770	15.509
Consulta 8	146.885	148.914	146.186	758	472.752	499.298	444.367	14.110
Consulta 9	319.938	348.487	314.859	9.703	952.817	979.705	919.891	19.786
Consulta 10	146.572	147.485	145.956	523	486.064	503.987	478.134	7.412
Consulta 11	146.463	147.261	145.874	391	496.218	519.108	479.576	9.372
Consulta 12	512.761	529.937	497.331	11.599	494.632	515.063	469.830	15.291

Tabela 6. Tempos de execução (ms) médio, máximo, mínimo e desvios padrão para o *dataset* de dispositivos estacionários

	Postgres-XL + PostGIS				Sedona + HDFS			
	Média	**Maior**	**Menor**	**Desvio**	**Média**	**Maior**	**Menor**	**Desvio**
Consulta 1	326.134	538.343	292.822	71.838	479.559	497.287	462.908	11.558
Consulta 2	616.071	1.155.786	468.815	199.215	518.600	536.717	496.675	12.275
Consulta 3	148.024	172.028	144.198	8.285	510.833	526.907	490.790	9.651
Consulta 4	1.570.164	1.616.075	1.516.330	24.680	519.088	541.895	494.286	12.741
Consulta 5	144.841	150.997	143.454	2.078	478.286	501.059	457.787	13.461
Consulta 6	143.940	144.239	143.626	179	477.542	497.105	446.702	14.830
Consulta 7	904.962	1.601.973	376.755	278.603	957.199	993.769	934.419	16.265
Consulta 8	153.994	158.790	153.074	1.640	482.151	502.123	466.789	12.006
Consulta 9	314.083	362.707	307.546	16.265	958.081	978.814	933.979	15.355
Consulta 10	144.911	146.446	144.101	665	476.796	489.110	464.230	9.309
Consulta 11	150.431	173.264	144.204	11.194	490.271	520.113	464.896	17.640
Consulta 12	536.240	626.030	494.060	35.934	498.325	525.624	480.935	12.702

não espaciais, o Postgres-XL se sobressaiu. Entretanto, na Consulta 7 em particular, o Postgres-XL apresentou um comportamento peculiar no *dataset* de dispositivos estacionários, tendo tempos de execução que ultrapassaram os do Apache Sedona. Em todas as consultas envolvendo agregações (Consultas 5, 6, 7 e 9) o Postgres-XL se sobressaiu.

Apesar de as observações aqui destacadas para cada consulta serem válidas para os três cenários de mobilidade, foi possível observar que a mobilidade dos dispositivos impactou as médias dos tempos de execução do Postgres-XL, que aumentaram de acordo

com a quantidade de dispositivos móveis no *dataset*. Também foi possível perceber que, no Apache Sedona, a mobilidade dos dispositivos influenciou os tempos de execução, com médias geralmente menores no *dataset* contendo dispositivos móveis, seguido, na maioria dos casos, pelo *dataset* misto.

7. Conclusão

Dados geoespaciais constituem uma parte significativa do volume de big data e seu valor tem despertado crescente interesse em diversos domínios, como planejamento urbano, monitoramento ambiental, logística, agricultura e transporte. O objetivo deste trabalho é propor uma abordagem para o armazenamento e a consulta de grandes volumes de dados geoespaciais provenientes de dispositivos IoT com variados níveis de mobilidade. Foi realizado um levantamento das principais ferramentas referenciadas na literatura para o armazenamento e análise de dados massivos geoespaciais, suas principais características e recursos geoespaciais, e a forma como essas ferramentas se integram.

Foi proposta também uma abordagem para big data geoespacial de dispositivos com diferentes níveis de mobilidade, por meio de uma arquitetura em três camadas que abrangem a coleta, o armazenamento e a análise dos dados. Um protótipo da camada de Armazenamento e Análise de Dados foi implementado, permitindo a avaliação do desempenho de algumas das principais ferramentas de big data, sob a perspectiva de diferentes níveis de mobilidade, considerando tanto dados de observações de sensores quanto dados de localização. A abordagem proposta pode beneficiar gestores de sistemas que lidam com grandes volumes de dados geoespaciais, oferecendo um modelo que leva em consideração o nível de mobilidade dos dispositivos para o armazenamento e processamento das informações.

Como trabalhos futuros, pretende-se dar continuidade à implementação da arquitetura, incluindo interfaces de consulta na camada de Aplicação e a integração com *middlewares* de Internet das Coisas na interface com o ambiente físico. Além disso, serão realizados experimentos com indexação espacial na camada de Armazenamento e Análise de Dados, e a implementação do Apache Cassandra em conjunto com o GeoMesa. Também será desenvolvida uma métrica para quantificar a mobilidade dos dispositivos representados. Com base nessa métrica, será criado um sistema para inferência do mecanismo de armazenamento mais adequado, considerando a relação entre o volume de dados de localização e de observações, o grau de mobilidade dos dispositivos e os tipos de operações mais frequentes sobre os dados. Por fim, pretende-se aplicar a arquitetura em um cenário real, no sistema de transporte de um campus universitário.

Referências

Ahmed, M., Karagiorgou, S., Pfoser, D., e Wenk, C. (2015). A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632. https://doi.org/10.1007/s10707-014-0222-6.

al Jawarneh, I., Bellavista, P., Corradi, A., e Foschini, L. (2020). Big Spatial Data Management for the Internet of Things: A Survey. *Journal of Network and Systems Management*, 28(4):990–1035. https://doi.org/10.1007/s10922-020-09549-6.

Al-Yadumi, S., Xion, T. E., Wei, S. G. W., e Boursier, P. (2021). Review on Integrating Geospatial Big Datasets and Open Research Issues. *IEEE Access*, 9:10604–10620. https://doi.org/10.1109/ACCESS.2021.3051084.

Alvarez, M. G., Morales, J., e Kraak, M.-J. (2019). Integration and Exploitation of Sensor Data in Smart Cities through Event-Driven Applications. *Sensors (Basel, Switzerland)*, 19. https://doi.org/10.3390/s19061372.

Azad, P., Navimipour, N. J., Rahmani, A. M., e Sharifi, A. (2020). The Role of Structured and Unstructured Data Managing Mechanisms in the Internet of Things. *Cluster Computing*, 23:1185–1198. https://doi.org/10.1007/s10586-019-02986-2.

Baig, F., Teng, D., Kong, J., e Wang, F. (2021). SPEAR: Dynamic Spatio-Temporal Query Processing over High Velocity Data Streams. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2279–2284. https://doi.org/10.1109/ICDE51399.2021.00237.

Breunig, M., Bradley, P. E., Jahn, M., Kuper, P., Mazroob, N., Rösch, N., Al-Doori, M., Stefanakis, E., e Jadidi, M. (2020). Geospatial Data Management Research: Progress and Future Directions. *ISPRS International Journal of Geo-Information*, 9(2). https://doi.org/10.3390/ijgi9020095.

Burrough, P. A. e McDonnell, R. A. (2015). *Principles of Geographical Information Systems*. OUP Oxford.

Catlett, C. E., Beckman, P. H., Sankaran, R., e Galvin, K. K. (2017). Array of Things: a Scientific Research Instrument in the Public Way: Platform Design and Early Lessons Learned. In *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, SCOPE '17, page 26–33, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3063386.3063771.

Chicago Data Portal (2022). Chicago - Bike Routes. https://data.cityofchicago.org/Transportation/Bike-Routes/3w5d-sru8. Acessado em 21 de janeiro de 2025.

Dai, J. e Lu, C.-T. (2011). DIME: Disposable Index for Moving Objects. In *2011 IEEE 12th International Conference on Mobile Data Management*, volume 1, pages 68–77. https://doi.org/10.1109/MDM.2011.69.

Eldawy, A. e Mokbel, M. F. (2016). The Era of Big Spatial Data: A Survey. *Foundations and Trends® in Databases*, 6(3-4):163–273. https://doi.org/10.1561/1900000054.

Fernández, P., Suárez, J. P., Trujillo, A., Domínguez, C., e Santana, J. M. (2018). 3D-Monitoring Big Geo Data on a Seaport Infrastructure Based on FIWARE. *Journal of Geographical Systems*, 20(2):139–157. https://doi.org/10.1007/s10109-018-0269-2.

Finogeev, A., Finogeev, A., Fionova, L., Lyapin, A., e Lychagin, K. A. (2019). Intelligent Monitoring System for Smart Road Environment. *Journal of Industrial Information Integration*, 15:15–20. https://doi.org/10.1016/j.jii.2019.05.003.

Lee, K., Liu, L., Ganti, R. K., Srivatsa, M., Zhang, Q., Zhou, Y., e Wang, Q. (2019). Lightweight Indexing and Querying Services for Big Spatial Data. *IEEE Transactions on Services Computing*, 12(3):343–355. https://doi.org/10.1109/TSC.2016.2637332.

Li, P., Lu, H., Zheng, Q., Yang, L., e Pan, G. (2020). LISA: A Learned Index Structure for Spatial Data. In Maier, D., Pottinger, R., Doan, A., Tan, W., Alawini, A., e Ngo, H. Q., editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 2119–2133. ACM. https://doi.org/10.1145/3318464.3389703.

Lwin, K. K., Takeuchi, W., Sekimoto, Y., e Zettsu, K. (2019). City Geospatial Dashboard: IoT and Big Data Analytics for Geospatial Solutions Provider in Disaster Management. In *2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–4. IEEE. https://doi.org/10.1109/ICT-DM47966.2019.9032921.

Pandey, V., Kipf, A., Neumann, T., e Kemper, A. (2018). How Good Are Modern Spatial Analytics Systems? *Proc. VLDB Endow.*, 11(11):1661–1673. https://doi.org/10.14778/3236187.3236213.

Shah, P. (2019). Developing Big Data Analytics Architecture for Spatial Data. In Bartolini, I. e Li, F., editors, *Proceedings of the VLDB 2019 PhD Workshop, co-located with the 45th International Conference on Very Large Databases (VLDB 2019), Los Angeles, California, USA, August 26-30, 2019*, volume 2399 of *CEUR Workshop Proceedings*. CEUR-WS.org. https://doi.org/10.1007/978-3-030-04780-1_17.

Shrivastava, S. (2020). A Review of Spatial Big Data Platforms, Opportunities, and Challenges. *IETE Journal of Education*, 61(2):80–89. https://doi.org/10.1080/09747338.2020.1835564.

Silva, D. S. e Holanda, M. (2022). Applications of Geospatial Big Data in the Internet of Things. *Transactions in GIS*, 26(1):41–71. https://doi.org/10.1111/tgis.12846.

Sukmaningsih, D. W., Suparta, W., Trisetyarso, A., Abbas, B. S., e Kang, C. H. (2020). *Proposing Smart Disaster Management in Urban Area*, pages 3–16. Springer International Publishing, Cham. 10.1007/978-3-030-14132-5_1.

Wang, E., Zhang, M., Yang, B., Yang, Y., e Wu, J. (2024). Large-scale spatiotemporal fracture data completion in sparse crowdsensing. *IEEE Transactions on Mobile Computing*, 23(7):7585–7601. https://doi.org/10.1109/TMC.2023.3339089.

Wang, G., Chen, Y., Wang, S., Zhang, F., e Zhang, D. (2023). ForETaxi: Data-Driven Fleet-Oriented Charging Resource Allocation in Large-Scale Electric Taxi Networks. *ACM Transactions on Sensor Networks*, 19(3). https://doi.org/10.1145/3570958.