UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ludmila Ribeiro Bôscaro Yung

Uma Abordagem *Polystore* Baseada em *Fog-Cloud*

Juiz de Fora
2024

Ludmila Ribeiro Bôscaro Yung

Uma Abordagem *Polystore* Baseada em *Fog-Cloud*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Dr. Mário Antônio Ribeiro Dantas

Juiz de Fora

2024

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Yung, Ludmila.
Uma Abordagem *Polystore* Baseada em *Fog-Cloud* / Ludmila Ribeiro Bôscaro Yung. – 2024.
71 f. : il.

Orientador: Mário Antônio Ribeiro Dantas
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2024.

1. IoT. 2. *Polystore*. 3. *Cloud*, *Fog* e *Edge*. I. Dantas, M. A. R., orient. II. Título.

Ludmila Ribeiro Bôscaro Yung

Uma Abordagem Polystore Baseada em um Ambiente Edge-Fog-Cloud

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre(a) em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 04 de março de 2024.

BANCA EXAMINADORA

Prof. Dr. Mario Antonio Ribeiro Dantas - Orientador
Universidade Federal de Juiz de Fora

Profª. Dra. Regina Maria Maciel Braga Villela
Universidade Federal de Juiz de Fora

Profª. Dra. Denise Stringhini
Universidade Federal de São Paulo

Juiz de Fora, 27/02/2024.

Documento assinado eletronicamente por **mario antonio ribeiro dantas**, **Usuário**

Externo, em 17/07/2024, às 17:13, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Regina Maria Maciel Braga Villela**, **Professor(a)**, em 05/09/2024, às 09:24, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Denise Stringhini**, **Usuário Externo**, em 05/09/2024, às 14:21, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1721286** e o código CRC **6E9EB819**.

Ludmila Ribeiro Bôscaro Yung

Uma Abordagem *Polystore* Baseada em *Fog-Cloud*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Aprovada em 04 de março de 2023

BANCA EXAMINADORA

Dr. Mário Antônio Ribeiro Dantas - Orientador
Universidade Federal de Juiz de Fora

Dra. Regina Maria Maciel Braga Villela
Universidade Federal de Juiz de Fora

Dra. Denise Stringhini
Universidade Federal de São Paulo

Dedicos este trabalho à todas as pessoas queridas por mim.

AGRADECIMENTOS

À Deus por sua infinita misericórdia e amor. Por me proporcionar tantas vitórias conquistadas ao longo de minha jornada no mestrado e fora dele; por me ensinar, orientar em alta voz e no silêncio também. Muitas vezes desacreditada estava, mas consegui chegar até aqui por Tua Graça.

Aos meus pais e irmãos por tanto carinho, amor, compreensão e motivação para finalizar este trabalho. Pelas diversas vezes que pararam para me ouvir nos momentos em que precisei desabafar.

Ao meu esposo Lucas por todo amor e compreensão, pelos diversos momentos em que me incentivou a continuar, por todos os "E o mestrado?" que me fizeram focar no que eu precisava entregar e por me apoiar a finalizar este ciclo em minha vida. Por ter sido meu companheiro do início ao fim, meu melhor amigo de mestrado na mesma área de atuação, desde a graduação.

Aos meus amigos que o Bom Deus colocou em meu caminho: os daqui do mestrado, em especial João Paulo Radd (e Alice e Cecília) e Gilson Fonseca. Aos meus amigos do Ministério de Música e da minha Paróquia Nossa Senhora Mãe de Deus, de modo especial à Lenise que sempre me apoiou, rezou por mim; não consigo citar todas as pessoas queridas desta comunidade. Aos amigos do meu clã *Dark Dragons*, às minhas amigas da computação e dos nossos *encontrinhos* (sejam estes presenciais ou não), a todos demais amigos: os que conquistei ao longo desta jornada, como também os que já existiam em meu caminho, meu muito obrigada por tantos maravilhosos momentos, pelo entendimento de minha ausência, pelo incentivo e tanto carinho.

Aos professores do PPGCC pelo conhecimento passado dentro e fora de sala: Jairo, Luiz Maurílio, Fernanda, Regina, Victor e Mário. A todos os funcionários do PPGCC que foram essenciais para esta conclusão de curso. Ao professor Victor, pelas dicas de melhoria deste trabalho.

À CAPES e à Fapemig pela oportunidade de ter sido bolsista por duas vezes ao longo de minha jornada como mestranda, pela possibilidade de ter sido tutora de duas disciplinas de cursos da graduação e poder ter aprendido mais com essas experiências que, juntamente com todo o trabalho realizado ao longo do curso, pude entregar o melhor ao meu alcance e aprender muito com as oportunidades.

Ao meu orientador, Mário, por todo seu apoio, carinho, palavras amigas, dicas, orientação e incentivo que fizeram com que eu chegasse até aqui, por sugerir muitas coisas que foram acrescentadas a este trabalho, trazendo riqueza para o conteúdo escrito e prático.

Aos professores que aceitaram participar da banca examinadora desta dissertação, em especial às professoras Denise e Regina. Meu muito obrigada pela disponibilidade, pelos pontos discutidos e por todas as sugestões para deixar este trabalho finalizado.

“É na Tua bondade sempre infinita que quero me perder, ó Coração de Jesus!”. Santa Teresinha

RESUMO

São inúmeros os desafios encontrados na integração e manipulação de dados em ambientes de IoT (*Internet of Things*). A demanda por processamento, consulta e integração de dados heterogêneos distribuídos tem crescido, tornando essencial a concepção de um ambiente capaz de lidar com tais exigências.

Considerando a complexidade desses cenários, é proposta a utilização de uma arquitetura *Polystore*, que visa facilitar a integração de dados em tempo de execução, permitindo a união de diversos tipos de bancos de dados e a construção de mecanismos de armazenamento integrados e heterogêneos. O estudo se concentra em resolver a dificuldade de processamento, organização, integração e armazenamento de dados na *Cloud*, levando em consideração as camadas *Cloud* e *Fog*. A abordagem proposta visa minimizar o uso intensivo de recursos na *Cloud*, promovendo o processamento de dados cada vez mais próximo das bordas da rede.

Para atingir esse objetivo, é fundamental compreender as interações entre as camadas de *Cloud*, *Fog* e *Edge*, considerando as demandas de desempenho, energia, latência e custo computacional típicas dos ambientes IoT. A arquitetura *Polystore* é apresentada como uma solução promissora para abordar esses desafios, permitindo uma integração eficiente de dados e um uso mais eficaz dos recursos disponíveis. Uma vez que, integrando bases heterogêneas, espera-se que, ao serem executadas as buscas, sejam retornados todos os dados inerentes à busca em cada uma dessas bases.

O foco do trabalho é validar a utilização dessa arquitetura em ambiente *Fog*, proporcionando uma abordagem mais eficiente e econômica para o processamento de dados em tempo real. A partir da análise e experimentação, busca-se responder questões específicas relacionadas à implementação e aos impactos dessa arquitetura, visando promover seu uso e sua eficácia em larga escala.

A proposta apresentada utiliza simulação, paralelismo e *Polystore* para processar dados em um ambiente simulado com camadas *Cloud* e *Fog*. Prepara nodos *fog* para simulação, formando um *framework* com alocação dinâmica de recursos usando *MySQL*, *MongoDB* e uma base heterogênea. O sistema resultante permite busca em bases heterogêneas e monitoramento do uso de recursos computacionais.

Palavras-chave: *Polystore*. Bancos de dados distribuídos. *Fog computing*. *Cloud computing*. *Fog-cloud computing*.

ABSTRACT

There are several challenges encountered in integrating and manipulating data in IoT (Internet of Things) environments. The demand for processing, querying and integrating heterogeneous distributed data has grown, making it essential to design an environment capable of dealing with such criteria.

Considering the complexity of these scenarios, the use of a Polystore architecture is proposed, which aims to facilitate data integration at runtime, allowing the union of different types of databases and the construction of integrated storage mechanisms and heterogeneous. The study focuses on solving the difficulty of processing, organizing, integrating and storing data in the cloud, taking into account the Cloud and Fog layers. The proposed approach aims to minimize the intensive use of resources in the cloud, promoting data processing increasingly closer to the edges of the network.

To achieve this goal, it is essential to understand the interactions between the cloud, edge and fog layers, considering the performance, energy, latency and computational costs demands typical of IoT environments. The Polystore architecture is presented as a promising solution to address these challenges, enabling efficient data integration and more effective use of available resources. Since, by integrating heterogeneous bases, it is expected that, when searches are carried out, all infected data will be returned to the search in each of these bases.

The focus of the work is to validate the use of the architecture in the Fog environment, providing a more efficient and economical approach to real-time data processing. Based on analysis and experimentation, we seek to answer specific questions related to the implementation and impacts of this architecture, promoting its use and effectiveness on a large scale.

The proposal presented uses simulation, parallelism and Polystore to process data in a simulated environment with Cloud and Fog layers. It prepares fog nodes for simulation, forming a framework with dynamic resource allocation using MySQL, MongoDB and a heterogeneous database. The resulting system allows searching in heterogeneous databases and monitoring the use of computational resources.

Keywords: Polystore. Distributed Databases. Fog computing. Cloud computing. Fog-cloud computing.

LISTA DE ILUSTRAÇÕES

Figura 1 - Visão geral do funcionamento de uma arquitetura que utiliza conceitos de IoT. Figura da autora. 21
Figura 2 - Um exemplo de arquitetura que utiliza *Cloud*, *Fog* e *Edge*. Adaptado de (41). 22
Figura 3 - Uma representação generalizada da utilização de BDs em ambientes com grandes volumes de dados, resultando em um sistema distribuído. Figura da autora. 23
Figura 4 - Representação da junção de IoT, *Cloud*, *Fog*, *Edge* e *Polystore* em um ambiente que opera de forma distribuída. Figura da autora. 25
Figura 5 - Interação do cliente ao utilizar a arquitetura *HKPoly* (9). 32
Figura 6 - A proposta tem três níveis essenciais: armazenamento de dados e acesso, processamento e nodos *edge*. Figura da autora. 38
Figura 7 - Características específicas para a concepção da proposta e respectivas ações esperadas de cada camada. Figura da autora. 40
Figura 8 - Um *schema* básico da implementação da comunicação do *Polystore* com os respectivos BDs. Figura da autora. 42
Figura 9 - Como as conexões dos BDs são feitas localmente: a esquerda a conexão com o MySQL e a direita a conexão com o MongoDB. Figura da autora. . . . 44
Figura 10 - Representação gráfica do ambiente de simulação. Figura da autora. . . 45
Figura 11 - Representação gráfica do ambiente de simulação e de sua aplicação. Figura da autora. 46
Figura 12 - Representação gráfica das funcionalidades e pontos-chave do ambiente. Figura da autora. 46
Figura 13 - Representação gráfica das funcionalidades e pontos-chave de métodos de alocação. Figura da autora. 47
Figura 14 - Etapas da simulação de uma rede *Cloud*, *Fog* e *Edge* em conjunto com *Polystore*. Figura da autora. 49
Figura 15 - Funcionalidades do servidor Cloud. Figura da autora. 49
Figura 16 - Como cada nodo *Fog* é representado dentro da simulação. Figura da autora. 50
Figura 17 - Funcionalidades do ambiente de simulação. Figura da autora. 51
Figura 18 - Exemplo de configuração dentro da função de distribuição de recursos. Figura da autora. 53
Figura 19 - O começo da execução retorna os recursos iniciais disponíveis. 55
Figura 20 - Resultados obtidos para um limite de utilização de 80% dos nodos disponíveis. 55
Figura 21 - Resultados de exemplo ao serem executadas três iterações do simulador. Figura da autora. 56
Figura 22 - Resultados de exemplo sob mudança de cenário de poder de processamento e disponibilidade. Figura da autora. 57
Figura 23 - ormato do resultado de uma busca dentro do *Polystore*. Figura da autora. 58
Figura 24 - Exemplo de resultado de utilização de nodos *fog* e do *Cloud Server*. Figura da autora. 59
Figura 25 - Exemplo de demonstração de cenário onde a *Cloud* possui um maior poder de processamento que a *Fog* e, portanto, é escolhida para execução das buscas. Figura da autora. 60
Figura 26 - Demonstração visual da aplicação de execução da proposta. Figura da autora. 61

LISTA DE TABELAS

Tabela 1 – Objetivos gerais abordados na presente dissertação 16
Tabela 2 – Características gerais que compõem um sistema *Polystore*. 26
Tabela 3 – String de busca 27
Tabela 4 – Resumo dos registros encontrados e trabalhos selecionados para análise 27
Tabela 5 – Principais pontos do trabalho realizado por (1). 32
Tabela 6 – Comparação de Referências sobre IoT, *Polystores* e Computação Distribuída. 36
Tabela 7 – Relação das funcionalidades do simulador com suas respectivas descrições. 48

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
BD	Banco de Dados
SGBD	Sistemas de Gerenciamento de Banco de Dado

SUMÁRIO

1 INTRODUÇÃO 14
1.1 MOTIVAÇÃO 15
1.2 OBJETIVOS 16
1.3 QUESTÕES DE PESQUISA 17
1.4 RESULTADOS 17
1.4.1 ORGANIZAÇÃO 18
2 AMBIENTE COMPUTACIONAL E TRABALHOS RELACIONADOS 20
2.1 AMBIENTE COMPUTACIONAL CONTEMPORÂNEO 20
2.1.1 *INTERNET OF THINGS* 20
2.1.2 *CLOUD*, *FOG*, *EDGE* 21
2.1.3 BANCOS DE DADOS DISTRIBUÍDOS E TECNOLOGIAS 22
2.1.4 *POLYSTORE* 23
2.2 TRABALHOS RELACIONADOS 26
2.2.1 *STRING* DE BUSCA 26
2.2.2 FERRAMENTAS DE BUSCA 27
2.2.2.1 ANÁLISE DE RESULTADOS OBTIDOS 28
2.2.3 MÉTODO DE PESQUISA: IDENTIFICANDO TRABALHOS RELEVANTES 28
2.2.4 ANÁLISE DE TRABALHOS 29
2.3 CONSIDERAÇÕES DO CAPÍTULO 35
3 PROPOSTA DA ARQUITETURA *FOG-CLOUD POLYSTORE* 37
3.1 ARQUITETURA 37
3.1.1 APLICAÇÃO DA PROPOSTA 39
3.2 DADOS UTILIZADOS 41
3.3 DESENVOLVIMENTO 42
3.3.1 *MERGE* 42
4 AMBIENTE E RESULTADOS EXPERIMENTAIS 45
4.1 SIMULAÇÃO 47
4.2 RESULTADOS 52
4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO 60
5 CONCLUSÕES E TRABALHOS FUTUROS 63
5.1 CONTRIBUIÇÕES 63
5.2 LIMITAÇÕES 63
5.3 TRABALHOS FUTUROS 64
REFERÊNCIAS 65
APÊNDICE A – Informações extras 71

1 INTRODUÇÃO

Como já é de conhecimento na sociedade contemporânea, a tecnologia ganha espaço cada dia mais e, com isso, vem exercendo um papel fundamental não somente no avanço de peças de *hardware* cada vez mais eficientes e potentes, como também, principalmente, no sentido de abordagens mais inteligentes no que se diz respeito a solução de problemas.

No contexto de *Internet* das Coisas (*Internet of Things* - IoT), existe um campo vasto a ser explorado. Como o próprio nome diz, IoT relaciona a conexão em uma rede de objetos/coisas (34), estando presente desde a área de saúde (45), agricultura (48), educação (60), *Ambient Assisted Living* (Vida Assistida em Ambiente) (3), etc. Tanto em bases de dados homogêneas como em heterogêneas, sistemas computacionais que permitem o gerenciamento desses dados são requeridos. Isso também vale para sistemas de gerenciamento de dados IoT (58).

De modo geral, sistemas de gerenciamento de dados IoT permitem ao usuário vastas opções de buscas, medições, controle e captura, além do processamento e armazenamento de todos os dados do sistema; muitos desses processos requerem uma boa segurança pois estarão circulando dados a todo momento, atualizações sendo requisitadas e colocadas no sistema.

São desafios como esses de gerenciar, proteger e analisar dados que tornam Sistemas de Gerenciamento de Dados IoT em um foco de estudo pois, além da interdisciplinaridade existente, há muitos desafios no quesito de armazenamento, distribuição dos dados, além de camadas *Cloud*, *Fog* e *Edge* presentes cada vez mais nesses sistemas (16).

Dados IoT são constantemente atualizados, uma vez que possuem a capacidade de mudar drasticamente em poucos segundos (um aferimento de pressão arterial em um paciente hipertenso, por exemplo) e, com isso, vem o desafio de armazenamento, devido ao fato de sistemas que não são somente voltados ao SQL padrão permitirem opções de escrita, leitura, junções com uma certa facilidade se comparados ao modelo clássico.

Por outro lado, ainda falando sobre armazenamento, têm-se conceitos computacionais que envolvem *Cloud*, *Fog* e *Edge*. Com a finalidade de se trabalhar com redes e sistemas descentralizados temos a necessidade de reduzir custos, latência, etc. como abordado em (37), de modo que ao operar na borda da rede (*Edge*) possa ser possível utilizar outro paradigmas, como é o caso da *Fog Computing*. Sendo assim, *Fog* e *Edge computing* podem ser vistos como paradigmas tecnológicos que buscam, através de aplicações/sistemas, o processo de dados próximo às suas fontes, a fim de reduzir latências e economizar largura de banda (32). Já a *Cloud* provê serviços às camadas *Fog* e *Edge* e, caso seja requisitado, essas camadas podem acessar a *internet* em qualquer momento de qualquer lugar (4).

A depender de características do sistema, têm-se uma necessidade de unir as bases heterogêneas de dados em forma de texto não estruturado, por exemplo, de dados relacionais, de imagens e vídeos, etc. Sendo assim, o conceito de *Polystore* torna-se utilizável gadepally2016bigdawg, uma vez que Bancos de Dados (BDs) que utilizam essa abordagem podem vir a aproveitar pontos fortes relativos dos Sistemas de Gerenciamento de Bancos de Dados (*Data Base Management Systems* - DBMSs).

A interessante característica da abordagem *Polystore* é da possbilidade de incluir em sua solução BDs federados e paralelos (24). Sendo assim, os componentes da abordagem trabalham para que exista uma melhoria nos mecanismos de consulta e análise de alto desempenho.

Independente de qual seja a abordagem, dados IoT são representados, de forma geral, como sensores (23), sejam eles de captura ou de envio de mensagens; mensagens estas que tem como destinatário a aplicação que, por sua vez, retorna resultados ao usuário ou até mesmo para uma equipe de pesquisadores e profissionais de TI e de outras áreas de conhecimento.

Esse processo de captura e envio costuma ser custoso em sistemas que possuem não somente uma grande quantidade de objetos que interagem com o sistema, mas também são constantemente manipulados e atualizados. Daí, portanto, a grande importância de tornar este processo rápido e de procurar a menor complexidade para realizar estas tarefas.

1.1 MOTIVAÇÃO

Apesar de se ter inúmeros dispositivos e máquinas que processam e geram dados, muitas vezes a parte de acesso, manipulação, consulta a esses dados é dada de forma sutil, ou seja, de uma forma mais transparente ao usuário. Existem etapas bem conhecidas da abordagem IoT, como integração, pré-processamento e armazenamento de dados, que comumente vêm sendo consideradas em vários sistemas como seus principais desafios.

Sendo assim, cenários de uso de Bancos de Dados heterogêneos e distribuídos crescem cada vez mais e, com isso, a necessidade de se manipular dados, realizar consultas, possibilidades de inferências de resultados e pesquisas, torna-se imprescindível. Etapas bem conhecidas no âmbito de IoT, como integração, pré-processamento e armazenamento de dados, fazem parte de inúmeros sistemas e, portanto, têm seus desafios.

De acordo com (42), há uma intensa produção de dados que são gerados e armazenados através de plataformas IoT e, com isso, têm-se a necessidade de se ter um ambiente propício para realizar integrações de dados, consultas coerentes e acesso aos mais diversos perfis profissionais, sejam esses profissionais capacitados em tecnologia ou não.

Existem etapas que são importantes neste contexto e que requerem atenção, tais como: processamento dos dados, organização de onde cada etapa vai acontecer no projeto, armazenamento, etapa de integração dos dados, ferramenta de busca e a utilização de *Polystore* como solução.

Por questões de custo computacional, tempo empregado, etc., não há a necessidade de trabalhar e levar todas essas etapas acima ditas para a *Cloud*. Sendo assim, portanto, cada vez mais esse processamento é levado para a *Edge* (bordas da rede). A solução *Polystore* entra como um adicional que promove facilitar essas integrações dentro das camadas edge e fog, sem causar impacto na *Cloud*.

Com o crescimento exponencial de dados têm-se o aumento da adoção de dispositivos IoT, há uma explosão na quantidade de dados gerados e armazenados. Cria-se, portanto, uma necessidade urgente de lidar com esses dados de maneira eficiente. Sendo assim, são diversos desafios na manipulação de dados: apesar da abundância de dispositivos e máquinas que geram dados, a parte de acesso, manipulação e consulta desses dados muitas vezes não é trivial. Isso ocorre porque os dados podem ser heterogêneos e distribuídos em diferentes dispositivos e sistemas, tornando a manipulação e a consulta uma tarefa desafiadora.

Existem etapas bem definidas no processamento de dados em ambientes de IoT, como integração, pré-processamento e armazenamento de dados. Essas, por sua vez, são cruciais e muitas vezes são consideradas os principais desafios em sistemas IoT. Com o crescimento dos BDs heterogêneos e distribuídos, surge uma necessidade premente de

integrar dados, realizar consultas e inferir resultados. Isso é fundamental para diversos perfis profissionais, sejam eles especializados em tecnologia ou não.

Devido a questões apontadas, anteriormente em 1, o custo computacional e tempo de processamento são presentes nesse tipo de sistema utiliza as tecnologias e abordagens IoT, *Polystore*, *Cloud*, *Fog* e *Edge*. Sendo assim, esses sistemas possuem uma tendência crescente de levar o processamento de dados para a *Fog* e *Edge*, em vez de depender exclusivamente de soluções da *Cloud*, a fim de evitar gastos desnecessários. Ou seja, especialmente relevante para ambientes de IoT, onde a latência e a largura de banda podem ser preocupações importantes.

Trazer como solução o *Polystore*, portanto, pode ser visto como uma maneira de facilitar a integração de dados nas camadas *Fog* e *Edge*, sem sobrecarregar os recursos da *Cloud*. Isso sugere uma abordagem eficiente para lidar com a complexidade da manipulação de dados em ambientes distribuídos de IoT.

1.2 OBJETIVOS

O objetivo desta dissertação é abordar os desafios e oportunidades associados à manipulação e acesso eficientes de dados em ambientes de IoT, considerando a abordagem *Polystore* como solução em um ambiente baseado em *Fog-Cloud*. Pois, com o crescente número de dispositivos conectados e a proliferação de dados gerados por esses dispositivos, torna-se fundamental desenvolver estratégias eficazes para lidar com essa imensa quantidade de informações de forma ágil e precisa. Abaixo, a tabela 1 apresenta, de forma sintetizada, cada um dos objetivos deste trabalho.

Objetivo	**Descrição**
Explorar os desafios da manipulação de Dados	Investigar os principais obstáculos enfrentados na manipulação de dados em ambientes IoT, incluindo questões relacionadas à heterogeneidade de dispositivos, distribuição de dados e complexidade na integração e consulta.
Analisar estratégias de Processamento de Dados em Ambientes Distribuídos	Avaliar as abordagens atuais para o processamento de dados em ambientes distribuídos, com foco na eficiência, escalabilidade e adaptabilidade a diferentes contextos de implementação, como *Cloud*, *Fog* e *Edge*.
Explorar o Potencial da Computação *Fog*	Investigar as vantagens e desafios da migração do processamento de dados para a *Fog*, incluindo a redução da latência, o alívio da carga na infraestrutura de rede e a capacidade de operar em ambientes com conectividade limitada.
Avaliar o papel da *Polystore* como solução de Integração de Dados	Examinar o uso de *Polystore* como uma solução promissora para facilitar a integração de dados em ambientes distribuídos de IoT, avaliando sua eficácia na redução da complexidade e no aumento da flexibilidade na manipulação de dados.

Tabela 1 – Objetivos gerais abordados na presente dissertação

A proposta de trabalho desta dissertação é, portanto, promover a validação de um ambiente que utiliza de tecnologia voltada ao *Polystore* que realiza a tarefa de

executar com maior transparência para o usuário a possibilidade de adequar o acesso às fontes heterogêneas de dados e utilização das mesmas sem que necessariamente se tenha conhecimento específico prévio da linguagem de acesso aos bancos em questão. Vale destacar a consideração da capilarização de dados digitais do *Polystore* que pode acabar piorando muitos problemas que temos hoje com relação a processamento, dados energéticos, digitalizados, etc.

Por fim, contribuir para o avanço do conhecimento, estudos e experimentos em IoT, *Polystore*, noções de *Cloud*, *Fog* e *Edge* e processamento de dados. Sendo assim, procurar fornecer conhecimento, recomendações práticas e diretrizes para profissionais e pesquisadores interessados em abordar os desafios associados à manipulação de dados em ambientes que utilizem esses tópicos abordados. Propiciar uma visão das questões relacionadas à manipulação de dados em ambientes de IoT e contribuir para o desenvolvimento de soluções inovadoras que impulsionem o progresso nesse campo em constante evolução. Além de promover o uso da arquitetura *Polystore* dentro da *Fog* para que as execuções e resultados obtidos sejam processados, cada vez mais, próximos às bordas da rede.

1.3 QUESTÕES DE PESQUISA

Esta seção visa explorar questões cruciais diante do contexto da implementação da proposta desta dissertação em questão, além de serem intrínsecas à pesquisa em si. Estas questões são fundamentais para orientar o desenvolvimento e compreensão da arquitetura proposta.

Q1 Quais são os passos para realizar essa validação da arquitetura?

Q2 Como a proposta ajudará os usuários finais, garantindo que os resultados sejam coesos?

Q3 Como serão realizadas as etapas dentro de cada uma das camadas computacionais?

Q4 Como realizar a integração, pré-processamento dos dados na camada *Fog*?

Q5 Quais são os benefícios e efeitos de se reduzir esses custos relacionados a *Cloud*?

Q6 Qual o impacto pode-se esperar que a arquitetura proposta tenha em larga escala?

Q7 É possível reduzir o custo de processamento do *Polystore* usando a *Fog*?

Todas as perguntas levantadas acima serão respondidas posteriormente, porém, de antemão, destaca-se a questão 7 dentre as demais. Independente do que os dados representam, um dos principais objetivos deste trabalho é fazer com que os mesmos sejam armazenados para que seja utilizado o *Polystore*, a fim de diminuir esse custo do uso da *Cloud*, foi proposta uma arquitetura que realizará uma integração dos dados de forma inicial, para que as aplicações que usem *Polystore* não precisem fazer muitas integrações na *Cloud*.

1.4 RESULTADOS

O grande desafio computacional observado atualmente pode ser considerado em como conceber um ambiente para lidar com bilhões de dispositivos IoT. O objetivo maior

é obter e processar os dados desses dispositivos, que apresentam desafios como acesso, manipulação e consulta desses dados digitais de forma mais transparente aos usuários finais. Portanto, a necessária utilização de uma arquitetura contemporânea deve ser concebida para suportar a relação entre as camadas de *Edge*, *Fog* e *Cloud*. Visando enfrentar esta questão, temos uma proposta que visa uma integração inicial de dados.

A proposta desta dissertação considera que as aplicações utilizam um ambiente *Polystore* que oferece facilidades para reduzir tempo e custos de processamento, em comparação com configurações comuns de *Cloud*.

De acordo com (54) as aplicações IoT têm sido cada vez mais utilizadas em vários tipos de aplicações: saúde, agricultura, cidades, casas inteligentes e veículos autônomos, por exemplo.

Muitos desses desafios envolvem desempenho, energia, dados recebidos em tempo real, latência e custo computacional (38), por exemplo. Em ambientes IoT, temos camadas de ação para cada parte do sistema/arquitetura, como *Cloud*, *Fog* e *Edge*, onde cada uma desempenha um papel diferente.

Existem vários cenários de utilização de Bancos de Dados (BD) heterogêneos e distribuídos. Esses, por sua vez, vêm crescendo cada vez mais, de acordo com (26). Portanto, torna-se imprescindível a necessidade de manipulação de dados, realização de consultas, possibilidades de inferência de resultados e pesquisas.

Alguns pontos são essenciais: acesso aos BDs, consultas de pesquisa, manipulação e operações em diferentes tipos de BDs. Para isso, a arquitetura *Polystore* surge como um diferencial que promove a reunião dos BDs e possibilita o acesso em tempo de execução, permitindo assim a construção de diversos mecanismos de armazenamento integrados e heterogêneos.

Além disso, existe a dificuldade de processamento dos dados, organização, integração e armazenamento na *Cloud* por uma questão de tempo e custo. Portanto, é interessante considerar o uso de uma arquitetura na qual consideremos as camadas *Cloud*, *Fog* e *Edge*. Para tanto, foi utilizada uma ideia/paradigma dessas camadas. Sendo assim, uma solução *Polystore* é abordada para facilitar essas integrações dentro dessas camadas sem impactar esse uso pesado *Cloud*, devido ao fato de se terem múltiplas tarefas, serviços, sendo executados simultaneamente em uma aplicação que envolve IoT.

Como resultados obtidos ao longo da pesquisa e desenvolvimento destaca-se a utilização de mecanismos de simulação, paralelismo e *Polystore* para processar dados em uma rede com camadas *Cloud* e *Fog*. Através de funções e métricas, nodos computacionais são executados, representando tanto um servidor *Cloud*, como nodos *Fog* com localizações e capacidades de processamento aleatórias, formando um *framework* que facilita a simulação dessas camadas e a alocação dinâmica de recursos. O sistema desenvolvido permite buscas em bases de dados heterogêneas e monitora o uso de recursos computacionais, considerando a latência da rede e determinando a execução de bases na *Fog* ou na *Cloud*.

1.4.1 ORGANIZAÇÃO

A organização desta dissertação é dada da seguinte forma: o capítulo 2 apresenta o ambiente computacional no qual este trabalho está inserido, bem como a motivação para o desenvolvimento da proposta. Além disso, o capítulo dois traz os trabalhos relacionados. O capítulo 3 mostra a Proposta da Arquitetura *Fog-Cloud Polystore*, assim como apresenta o cenário de aplicação inicial e os experimentos realizados. Já o capítulo 4

traz a Ambiente e Resultados Experimentais, bem como a implementação da proposta, bem como experimentos e resultados obtidos. Por último, o capítulo 5 apresenta as considerações finais e planos para trabalhos futuros.

2 AMBIENTE COMPUTACIONAL E TRABALHOS RELACIONADOS

Como já é de conhecimento na sociedade contemporânea, a tecnologia ganha espaço cada dia mais e, com isso, vem exercendo um papel fundamental não somente no avanço de peças de *hardware* cada vez mais eficientes e potentes, como também, principalmente, no sentido de abordagens mais inteligentes no que se diz respeito a solução de problemas.

Este capítulo traz um mapeamento dos conceitos das palavras-chave inerentes ao trabalho realizado, a fim de melhor contextualizar as tecnologias empregadas e utilizadas durante a pesquisa e desenvolvimento da proposta, bem como exemplos de uso e características importantes de IoT, Bancos de Dados Distribuídos, *Cloud*, *Fog*, *Edge* e *Polystore*.

2.1 AMBIENTE COMPUTACIONAL CONTEMPORÂNEO

2.1.1 *INTERNET OF THINGS*

No contexto de Internet of Things (IoT), tem-se um campo vasto a ser explorado. Como o próprio nome diz, *Internet* das Coisas em tradução ao português, IoT relaciona a conexão em uma rede de objetos/coisas, estando presente desde a área de saúde (46), agricultura (47), educação (61), *Ambient Assisted Living* (2), etc. Tanto em bases de dados homogêneas como em heterogêneas, sistemas computacionais que permitem o gerenciamento desses dados são requeridos. Isso também vale para sistemas de gerenciamento de dados IoT (57).

De modo geral, sistemas de gerenciamento de dados IoT permitem ao usuário vastas opções de buscas, medições, controle e captura, além do processamento e armazenamento de todos os dados do sistema; muitos desses processos requerem uma boa segurança pois estarão circulando dados a todo momento, atualizações sendo requisitadas e colocadas no sistema e, portanto, de acordo com (35) evitar ataques é algo primordial.

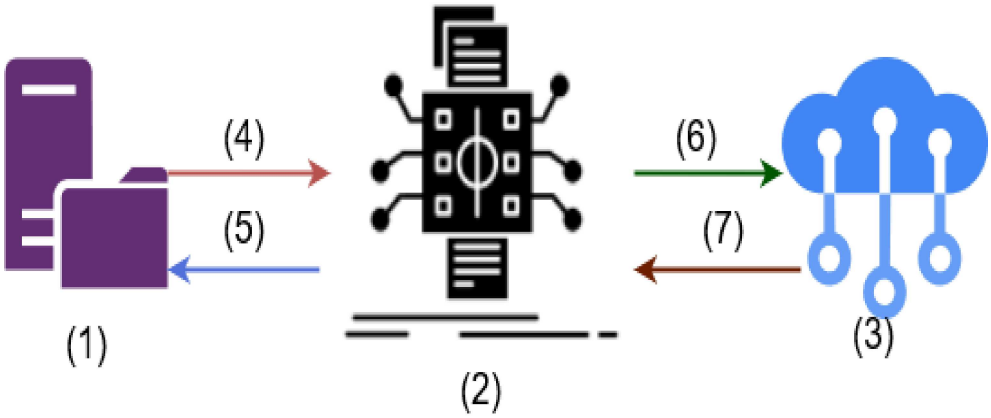
De acordo com (11), são desafios como esses de gerenciar, proteger e analisar dados que tornam Sistemas de Gerenciamento de Dados IoT em um foco de estudo pois, além da interdisciplinaridade existente, há muitos desafios no quesito de armazenamento, distribuição dos dados, além de camadas *Cloud*, *Fog* e *Edge* presentes cada vez mais nesses sistemas.

Muitos dos dados utilizados em IoT possuem formatos diversos, tais como: textos, documentos (16), áudios (28), vídeos (20), imagens (19), etc. que precisam ser armazenados e que estão relacionados a um determinado contexto, sendo assim, portanto, precisam ser integrados caso façam parte de um mesmo sistema de gerenciamento.

Dados IoT são constantemente atualizados, uma vez que possuem a capacidade de mudar drasticamente em poucos segundos (um aferimento de pressão arterial em um paciente hipertenso, por exemplo) e, com isso, vem o desafio de armazenamento, devido ao fato de sistemas que não são somente voltados ao SQL padrão permitirem opções de escrita, leitura, junções com uma certa facilidade se comparados ao modelo clássico.

Em suma, resumindo o exposto anteriormente, a figura 1 representa uma visão mais alto nível do que compõe um ambiente IoT. Na figura estão expostos números de 1 a 7 que representam os principais elementos de uma arquitetura que se beneficia de IoT.

Temos que:



– Figura 1 - Visão geral do funcionamento de uma arquitetura que utiliza conceitos de IoT. Figura da autora.

- (1) - Representa os dispositivos, quer eles sejam sensores *Edge* ou sensores distribuídos.
- (2) - Representa o processamento dos dados. Aqui pode ser em um centro de dados ou até mesmo na *Cloud*, por exemplo.
- (3) - Plataforma IoT que provê a parte de análise de gerenciamento de dispositivos.
- (4) - O dado é passado ao processamento como sua versão mais primitiva, assim que coletado.
- (5) - O centro de processamento envia os comandos, execuções, etc. necessários aos dispositivos.
- (6) - Os dados processados são passados à plataforma IoT.
- (7) - Configurações de implantação são enviadas ao centro de processamento de dados.

2.1.2 *CLOUD*, *FOG*, *EDGE*

Ainda falando sobre armazenamento, existem conceitos computacionais que envolvem *Cloud*, *Fog* e *Edge*. Com a finalidade de se trabalhar com redes e sistemas descentralizados, temos a necessidade de reduzir custos, latência, etc. Isso significa que ao operar na borda da rede, podemos utilizar outras abordagens, como a *Fog*.

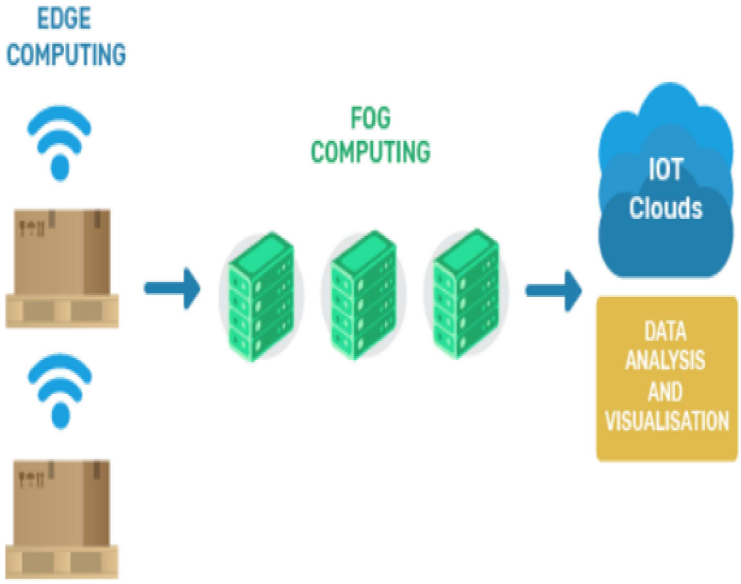
Sendo assim, pode-se chamar *Fog* e *Edge* de um paradigma tecnológico (36) que busca, através de aplicações/sistemas, o processamento de dados próximo às suas fontes, a fim de reduzir latências e economizar largura de banda.

- ***Cloud Computing***: Provê serviços às camadas *Fog* e *Edge* e, caso seja requisitada, as camadas que requisitaram serviços podem acessar a *internet* em qualquer momento e de qualquer lugar (32).
- ***Fog Computing***: É uma extensão do conceito da *Edge*, onde o processamento de dados ocorre em nodos mais próximos ao usuário final, mas ainda dentro da

infraestrutura de rede. Isso permite uma execução mais rápida de aplicativos e serviços, mantendo uma parte do processamento mais próximo do usuário.

- ***Edge Computing***: Refere-se ao processamento de dados próximo às fontes de dados, como sensores ou dispositivos IoT, em vez de enviá-los para um *data center* remoto. Portanto, auxiliar a reduzir a latência e a minimizar a sobrecarga da rede.

A figura 2 representa a junção de *Cloud*, *Fog* e *Edge* numa aplicação IoT. Nota-se que os nodos *Edge* situam-se mais externos ao servidor *Cloud*, que, por sua vez, possui análise de dados e visualização dos mesmos. Quanto à *Fog*, situada ao centro, atua como uma comunicação de intermédio entre a *Edge* e a *Cloud*.



– Figura 2 - Um exemplo de arquitetura que utiliza *Cloud*, *Fog* e *Edge*. Adaptado de (41).

2.1.3 BANCOS DE DADOS DISTRIBUÍDOS E TECNOLOGIAS

Para gerenciar dados na *Cloud*, é comum recorrer a Sistemas de Gerenciamento de Banco de Dados (SGBDs) relacionais (31). Muitos desses sistemas possuem versões distribuídas e operam eficientemente na *Cloud*. Embora sejam capazes de integrar suporte para vários tipos de dados, como objetos multimídia e documentos, muitas vezes resulta em perda de desempenho, simplicidade e flexibilidade, especialmente para aplicativos com requisitos de desempenho específicos e rígidos. Assim, argumenta-se que são necessários mecanismos de SGBDs mais especializados para casos específicos (29).

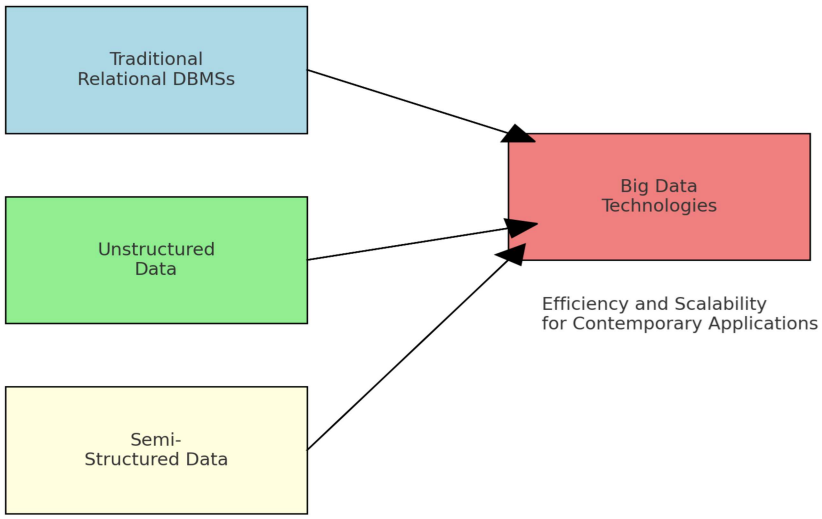
Por exemplo, em (49), mostra que SGBDs orientados a colunas, que armazenam dados em colunas ao invés de linhas como nos SGBDs relacionais tradicionais, demonstram um desempenho significativamente superior em cargas de trabalho. Da mesma forma, sistemas de gerenciamento de fluxo de dados são projetados especificamente para lidar de maneira eficiente com fluxos contínuos de dados.

Diante da crescente diversidade e volume de dados, diversas soluções especializadas de gerenciamento de dados foram propostas para atender diferentes tipos de dados e tarefas. Essas soluções muitas vezes superam os sistemas de gerenciamento de banco de

dados (SGBDs) relacionais tradicionais em ordens de magnitude em termos de eficiência e escalabilidade. Exemplos notáveis dessas novas tecnologias incluem sistemas de arquivos distribuídos, como o *Hadoop Distributed File System* (HDFS) (56), e estruturas de processamento paralelo de dados, como *Apache Spark*, que são projetadas para lidar com os desafios impostos pelo *Big Data* (14).

Os sistemas de arquivos distribuídos permitem o armazenamento e a gestão de grandes volumes de dados de forma eficiente (18), distribuindo-os entre múltiplos nodos para aumentar a disponibilidade e a tolerância a falhas. Por outro lado, as estruturas de processamento paralelo de dados permitem a execução de operações de análise de dados em larga escala, aproveitando o poder de processamento de múltiplas máquinas para acelerar o tempo de resposta e melhorar o desempenho.

A figura 3 representa a ligação dos BDs tradicionais relacionais e dos tipos de dados, com tecnologias que utilizam para fins de escalabilidade e eficiência quando se tem um grande volume de dados.



– Figura 3 - Uma representação generalizada da utilização de BDs em ambientes com grandes volumes de dados, resultando em um sistema distribuído. Figura da autora.

2.1.4 *POLYSTORE*

Tecnologias de gerenciamento de dados resultaram em uma oferta diversificada de serviços que podem ser utilizados para construir aplicativos intensivos em dados na *Cloud*, capazes de escalar e apresentar alto desempenho.

No entanto, essa diversidade também levou à fragmentação das interfaces de armazenamento de dados, dificultando a construção de aplicativos que utilizem múltiplos armazenamentos de dados, como sistemas de arquivos distribuídos, SGBDs relacionais e SGBDs NoSQL. Essa complexidade motivou o desenvolvimento dos *Polystores*, que fornecem acesso integrado a vários armazenamentos de dados na *Cloud* através de uma ou mais linguagens de consulta, facilitando a construção de aplicativos que necessitam de múltiplos tipos de armazenamento de dados.

A depender de características do sistema, têm-se uma necessidade de unir as bases heterogêneas de dados em forma de texto não estruturado, por exemplo, de dados relacionais, de imagens e vídeos, etc. Sendo assim, o conceito de *Polystore* torna-se

utilizável, uma vez que Bancos de Dados (BDs) que utilizam essa abordagem podem vir a aproveitar pontos fortes relativos dos SGBDs de acordo com (24).

A interessante característica da abordagem *Polystore* é a possibilidade de incluir em sua solução BDs federados e paralelos (24). Sendo assim, os componentes da abordagem trabalham para que exista uma melhoria nos mecanismos de consulta e análise de alto desempenho.

Independente de qual seja a abordagem, dados IoT são representados, de forma geral, como sensores, conforme visto em (23). Sejam esses sensores de captura ou de envio de mensagens; mensagens estas que tem como destinatário a aplicação que, por sua vez, retornam resultados ao usuário ou até mesmo para uma equipe de pesquisadores e profissionais de TI e de outras áreas de conhecimento. Sendo assim, um mecanismo que abrange melhor a integração dos dados e opera funções, pesquisas e operações nesse conjunto de dados, é uma alternativa mais adequada.

Uma importante etapa de sistemas que utilizam IoT é o processo de captura e, tanto este como o envio de dados, costumam ser custosos em sistemas que possuem não somente uma grande quantidade de objetos que interagem com o sistema. Também são constantemente manipulados e atualizados, daí, portanto, a grande importância de tornar este processo rápido e de procurar a menor complexidade para realizar estas tarefas.

Através da necessidade de realizar dados integrados e mecanismos de armazenamento heterogêneos, se tem também a exigência de um tipo de arquitetura atue de forma eficiente. Sendo assim, o *Polystore* vem com a proposta de trabalhar com ambientes heterogêneos, possibilitando acesso em tempo de execução, consultas de pesquisa que promovam um acesso transparente aos seus usuários (59).

Com o acima exposto, fica evidente que um destaque da característica de um *Polystore* se dá por sua capacidade de lidar com a integração de diversos armazenamentos de dados na *Cloud*. Existem algumas divisões que podem ser caracterizadas em níveis de acoplamento (51): fortemente acoplado, fracamente acoplado e híbrido, conforme a forma como interligam e comunicam os diferentes sistemas de armazenamento. Essa comunicação é feita através de uma API (*Application Programming Interface*), em português "Interface de Programação de Aplicação".

As características de acoplamento indicam que um *Polsytore* quando é:

- **Fracamente Acoplado** (52): indica que o sistema de armazenamento é similar a um conceito de mediador[1], possui uma interface intuitiva ao usuário e a possibilidade de localmente controlar o armazenamento de dados que é independente.
- **Fortemente acoplado** (12): indica que o sistema de múltiplo armazenamento permita, de forma local, interação do usuário para melhor performance através do compartilhamento de carga de trabalho (*workload*). Além disso, permite a junção de dados de distintos BDs.
- **Híbrido** (52): indica a combinação de um *Polystore* fracamente e fortemente acoplados. Permite a otimização através do uso de *queries* de pesquisa nativas e um operador ordenador para múltiplas *queries* de dados que são armazenados na *Cloud*.

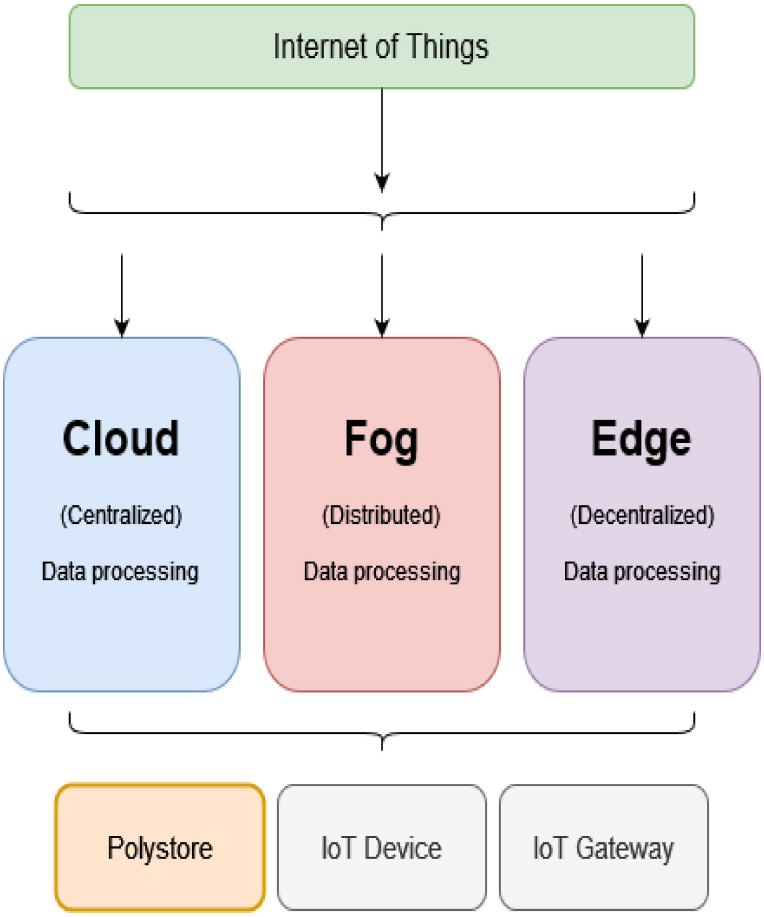
Por mediador, ou *wrapper*, entende-se que é um atuador, como uma sub-rotina em uma biblioteca de *software* ou um programa de computador cujo objetivo principal

[1] IgniteTech

é chamar uma segunda sub-rotina ou uma chamada de sistema com pouca ou nenhuma computação adicional.

Em resumo, o *Polystore* é, essencialmente, um SGBD construído sobre múltiplos mecanismos de armazenamento integrados e heterogêneos. Conforme visto em (13), é possível operá-lo de maneira eficiente, mapeando diversas fontes de dados, independentemente das diferenças nas tecnologias subjacentes de cada uma dessas fontes. Daí portanto a escolha de se utilizar essa arquitetura no contexto de IoT, em que diversas fontes de dados enviam ao sistema o mapeamento de sensores, dados de medidas, etc.

Ao mapear essas diversas fontes de dados, o *Polystore* consegue oferecer um acesso integrado e unificado aos dados, independentemente de onde estejam armazenados na *Cloud*. Essa característica pode ser vista na figura 4, em que se torna fundamental para garantir a flexibilidade e escalabilidade necessárias em ambientes corporativos que dependem de múltiplas plataformas de armazenamento na *Cloud* para gerenciar seus dados.



– Figura 4 - Representação da junção de IoT, *Cloud*, *Fog*, *Edge* e *Polystore* em um ambiente que opera de forma distribuída. Figura da autora.

Em resumo, *Polystores* fornecem acesso integrado ou transparente a uma série de armazenamentos de dados na *Cloud* por meio de uma ou mais linguagens de consulta. A tabela 2 contém as principais características encontradas em um *Polystore*, bem como os tipos existentes e como se dá, de forma geral, a interação com os dados.

Por fim, o presente trabalho cita, através de nomenclaturas ao longo da dissertação, formas diferentes de retratar o *Polystore*: arquitetura e abordagem. Neste caso, ambas nomenclaturas são utilizadas ao longo da escrita e, vale salientar, que ambas estão corretas.

Integração de diversos armazenamentos de dados na *Cloud*
Divisão em níveis de acoplamento: fortemente, fracamente, híbrido
Construído sobre múltiplos mecanismos de armazenamento integrados e heterogêneos
Mapeamento de diversas fontes de dados
Acesso integrado e unificado aos dados
Flexibilidade e escalabilidade em ambientes corporativos

Tabela 2 – Características gerais que compõem um sistema *Polystore*.

Em alguns trabalhos, como o visto em (8), o *Polystore* é retratado como um serviço. Independente da nomenclatura utilizada, vale lembrar que o *Polystore* é uma abordagem e, portanto, ao se utilizar seus conceitos e aplicações, forma-se uma arquitetura. Esta arquitetura, por sua vez, pode operar como um serviço no ambiente em que se propõe aplicação.

2.2 TRABALHOS RELACIONADOS

Nos últimos anos, a necessidade de integrar e gerenciar grandes volumes de dados distribuídos em diversas plataformas de armazenamento na *Cloud* levou ao desenvolvimento de SGBDs e, em particular, ao *Polystore*. Estes sistemas têm como principal objetivo fornecer um acesso integrado e eficiente a dados armazenados em diferentes ambientes, superando as limitações dos tradicionais sistemas de banco de dados monolíticos.

Diversos estudos e pesquisas têm explorado diferentes aspectos dos sistemas *Polystore*, desde a arquitetura e os mecanismos de integração até a performance e a escalabilidade em ambientes corporativos. Esta seção revisa os principais trabalhos relacionados ao tema, destacando as abordagens utilizadas para a integração de múltiplos armazenamentos de dados, as estratégias de acoplamento (fortemente, fracamente e híbrido) e as soluções propostas para o mapeamento e acesso unificado a dados heterogêneos. Além disso, são discutidas as principais vantagens e desafios identificados nesses estudos, proporcionando uma visão abrangente do estado da arte.

2.2.1 *STRING* DE BUSCA

O conceito básico de um *string* de busca[2] é o de utilizar termos textuais, expressões, palavras-chave em uma única forma de pesquisar acerca de um assunto. Tornando assim, portanto, o resultado das buscas relacionada ao assunto mais pertinente, de modo que sejam retornadas informações adequadas ao contexto. Primeiramente foram feitas consultas usando os termos principais deste trabalho tais como: IoT, *Cloud*, *Fog*, *Polystore* e Bancos de Dados Distribuídos.

Essa etapa de inserção da *string* de busca nas ferramentas de pesquisa foi feita baseada em leitura de título e *abstract* desses artigos encontrados para que, assim que fosse constatado de que estes fazem parte do contexto deste trabalho, a leitura de modo

[2] DIC

detalhado de cada um pudesse continuar para que uma melhor compreensão dos mesmos pudesse ser feita posteriormente.

Abaixo, encontra-se a *string* de busca utilizada para mapear os trabalhos relacionados.

```
("iot"OR "internet of things"OR "internet of
everything"OR "internet of objects")
AND ("distributed database"OR "heterogeneous
database"OR "polystore")
AND ("cloud"OR "fog"OR "edge")
```

Tabela 3 – String de busca

O funcionamento da *string* nas ferramentas de busca se dá pela procura de termos que essencialmente os trabalhos retornados devem ter, os termos *OR* (*ou*, em português) dentro dos grupos indicam que possa existir um ou mais termos encontrados nas buscas dentro de cada grupo e os termos *AND* (*e*, em português) englobam os termos de cada um dos grupos nos resultados. Vale destacar que cada termo possui alguma variação ou acréscimo ao resultado final, sendo assim, portanto, viável considerar termos diferentes para um mesmo conceito e aplicação.

2.2.2 FERRAMENTAS DE BUSCA

Para que a *string* fosse submetida vale ressaltar que cada plataforma de busca funciona de maneira própria e não necessariamente de forma semelhante na busca, ou seja, a estrutura da *string*, apresentada na subseção anterior, pode ter sua estrutura variar de ferramenta para ferramenta. Vale lembrar que o processo de pesquisa por trabalhos relevantes foi feito através das buscas avançadas de cada ferramenta.

Para que esse procedimento retornasse resultados, foram consideradas três plataformas de pesquisa de artigos, periódicos, etc. dentre os anos de 2018 até 2024. As ferramentas de busca utilizadas são: *IEEExplore*, *Springer* e ACM *Digital Library*. Os resultados obtidos podem ser vistos na tabela 4.

Ferramenta	**Registros encontrados**	**Trabalhos selecionados**
IEEExplore	96	7
Springer	17*	7
ACM *Digital Library*	286	7

Tabela 4 – Resumo dos registros encontrados e trabalhos selecionados para análise

O objetivo desta busca é, inicialmente, testar a capacidade das ferramentas de busca ao filtrar através da *string* de busca os termos inerentes ao escopo deste presente trabalho. Para obtenção de um resultado mais detalhado e com melhor acurácia, o ideal é dar enfoque no objetivo de investigar e identificar na literatura abordagens que conectam os tópicos relacionado ao tema foi proposto.

2.2.2.1 ANÁLISE DE RESULTADOS OBTIDOS

Ao voltar à tabela 3, verifica-se a demonstração da *string* de busca que fora utilizada para a colheita de trabalhos relacionados. Porém, nota-se que há uma certa discrepância de valores entre ferramentas de buscas e, ainda mais, na quantidade final escolhida.

Essa diferença de resultados se dá pelas publicadoras das ferramentas, a depender da quantidade de trabalhos publicados e da forma que as ferramentas buscam a *string*.

Na tabela 4 nota-se que a ferramenta ACM *Digital Library* teve inúmeros resultados retornados. Esses, por sua vez, em sua maioria, não eram correspondentes ao escopo. Uma análise mais aprofundada de Revisão Sistemática poderia retornar valores mais condizentes com os das demais ferramentas.

No resultados obtidos da ACM a maioria dos trabalhos possuía quase nenhuma relação com o tema deste trabalho, abordando outras temáticas, tais como: *Blockchain*, problemas e soluções generalizadas de BD, além de poucas referências voltadas à junção dos principais termos deste trabalho.

Nota-se que, na tabela 4, o resultado da ferramenta *Springer* (marcada com um asterisco $*$) é do total de 17 aplicando a *string* de busca. Porém, ao analisar os trabalhos, os resultados mostram que a maioria dos trabalhos é de livros, anais, periódicos, etc. Sendo assim, há uma extensão muito grande de trabalhos dentro desses destacados, o que dificulta o processo ainda mais de análise de trabalhos relevantes. Nesse caso, foi feita uma pesquisa de forma mais ampla na ferramenta tradicional de busca da *Springer*, ao invés da busca avançada, para verificar quantos trabalhos seriam retornados. O resultado desse experimento mostrou-se ruim, retornando mais de 2.900 trabalhos. Optou-se, portanto, a primeira pesquisa e nela avaliar quais trabalhos seriam relevantes dentro da ferramenta no contexto de aplicação atual.

A metodologia de pesquisa aplicada para seleção de trabalhos relacionados se dá na próxima subseção.

2.2.3 MÉTODO DE PESQUISA: IDENTIFICANDO TRABALHOS RELEVANTES

Inicialmente, a proposta é de, justamente, submeter a *string* nas ferramentas de busca de artigos e verificar a quantidade dos mesmos que foram retornados destas pesquisas. Aqui a questão do tempo de publicação é importante também, afinal procura-se por abordagens mais recentes, de, no máximo, até seis anos atrás da atual data deste trabalho (2024). Essa janela de tempo levou em consideração a maioria dos trabalhos encontrados, porém, alguns resultados mais antigos igualmente relevantes e necessários para a compreensão da abordagem foram considerados ao final da análise. Tendo em vista esta etapa, todos os trabalhos tiveram títulos e *abstracts* lidos de modo que, os que não se encaixavam logo na etapa de lida do título e *abstract*, fossem descartados.

Alguns requisitos para a inclusão de trabalhos para análise foi considerar se os mesmos possuem vínculo com os termos da *string* para que assim a próxima etapa, leitura detalhada, pudesse ocorrer.

Resumidamente, espera-se que os trabalhos retornados tenham como objetivo trabalhar com *Polystore*, IoT e questões e desafios de armazenamento de dados. Sendo assim, caso o *abstract* e título do artigo que foi analisado não contenha pelo menos dois termos da *string* de busca e tampouco trabalhar com questões de *Polystore* e os paradigmas *Cloud*, *Fog* e *Edge*, ele logo é descartado e passa-se para a leitura do próximo.

Ainda que os resultados retornados contenham os termos da *string* em questão, mas não apresentam algo concreto nas possíveis abordagens e soluções dos mesmos, portanto, são considerados como apenas citação de exemplos. Ou seja, o artigo pode até citar os termos, porém, se este não promove nenhuma forma de utilização prática ou teórica dos mesmos, ele é descartado.

Ainda que a restrição de escolha de trabalhos relevantes tenha sido citada como exemplo do uso de, ao menos, dois termos da *string* que estejam no artigo selecionado, alguns artigos retornaram compatibilidade com os questionamentos e contribuição levantados na seção 1 e, portanto, foram aceitos como relevantes. Portanto, a questão da quantidade de termos da *string* não necessariamente é um caráter eliminatório de trabalhos relacionados.

Portanto, a etapa de análise inicial tornou-se complexa uma vez que não se tratava apenas de identificar quantidades de termos presentes nos artigos mas, também, principalmente, de avaliar se todo o contexto estava associado a problemas/soluções/abordagens/modelos de armazenamento e suas tecnologias.

Esse levantamento de resultados ajuda a compreender a evolução dos trabalhos selecionados, assim como semelhanças, diferentes técnicas de solução e abordagem, desafios encontrados, entre outros. Todos voltados ao escopo principal deste trabalho.

2.2.4 ANÁLISE DE TRABALHOS

Conforme discutido na subseção 1.1, o interesse da comunidade científica, civil e acadêmica nos dispositivos da Internet das Coisas, nas Camadas Computacionais (*Cloud*, *Fog* e *Edge Computing*) e nos Bancos de Dados, e seus desafios, está crescendo (6). Afinal, esse interesse faz parte cada vez mais da realidade de muitos usuários, sejam eles pessoas que desejam utilizar esses dispositivos ou de profissionais.

Portanto, fica claro que os dispositivos IoT e os conceitos relacionados a eles são através de Smart Cities (e outras classificações associadas), *Scaling* e *Low Latency*, entre outros, são aspectos recorrentes que giram em torno de aplicações que envolvem *Cloud*, *Fog* e *Edge* (25).

Além disso, também tendemos a trabalhar cada vez mais com Bancos de Dados heterogêneos que desempenham um papel essencial no armazenamento, processamento, entrega de consultas, manipulação de dados e análises pertinentes.

Com isso, surge a necessidade de reunir em um só lugar a possibilidade de trabalhar com bancos de dados heterogêneos, que podem variar de acordo com o tipo de cada BD; além da necessidade de armazenar os dados e facilitar o uso nessas bases sem prejudicar o desempenho para que o acesso seja facilitado. As operações são realizadas mais rapidamente nas bordas da rede.

Recapitulando o exposto em **2.1.4**, *Polystore* não é um SGBD, mas possui um mecanismo de armazenamento distinto que promove o acesso aos dados através de seu meio de consulta, ou seja, através de uma consulta (24).

A principal coisa sobre o artigo de (43) é trabalhar em um ambiente onde as fontes de *lead time*, atraso e custo sejam executadas para atender aos requisitos do usuário que minimizem os *lead times*. Por ter um foco sensível ao tempo, já que as aplicações não podem ser processadas em apenas uma *Cloud*, é necessário priorizar o uso do *Fog*.

De acordo com (43), os sistemas baseados em IoT requerem *Fog* e *Cloud Computing*.

Os sensores e ferramentas avançados que aumentaram a eficiência e as vantagens do sistema foram observados em países desenvolvidos e em desenvolvimento. Por exemplo, (43) relata muitos estudos que trabalham com o sistema de computação baseado em *Fog* e *Cloud* e que nesses estudos foi considerado "conveniente para os pacientes devido ao seu custo-benefício, confiabilidade e segurança".

Existem problemas ao trabalhar com dados heterogêneos de fontes múltiplas que foram pesquisados no contexto de sistemas multi-bancos de dados, sistemas *Polystore* e sistemas de integração de dados (51). Alguns conjuntos de dados possuem uma enorme quantidade de dados de diversas fontes de dados, portanto, pode haver vários idiomas para obter acesso a esses diferentes conjuntos de dados.

Por outro lado, (15) traz comparações das principais arquiteturas utilizadas em Sistemas de Big Data e uma dessas arquiteturas é justamente a *Polystore*. Além de explicar como funciona o *Polystore*, os autores demonstram como ele funciona em um cenário básico de um sistema de Big Data, incluindo volume, velocidade, veracidade, variedade e variabilidade.

Em (40) há uma proposta de união de uma *S-Store* com a arquitetura *BigDawg Polystore*. Os autores propuseram o uso do *S-Store* para satisfazer alguns requisitos deste sistema, tais como: fornecer processamento baseado em push de baixa latência e perfeitamente integrado ao gerenciamento de dados ACID. Além de garantir que todos os estados estejam em estado de fluxo, janelas ou tabelas relacionais só podem ser acessadas dentro do contexto de uma transação.

Em (22) há uma ferramenta chamada *Polyflow*, que se baseia no conceito de sistemas *Polystore*, integrando diversas bases de dados de origem heterogênea adotando um esquema *ProvONE* global. *Polyflow* permite que cientistas consultem vários gráficos de proveniência de forma integrada. Com o fim de avaliar a *Polyflow*, especialistas usaram procedência de dados coletados de experimentos reais que geram árvores filogenéticas por meio de fluxos de trabalho, tudo isso para avaliar se o funcionamento da ferramenta condiz com essas diretrizes. Os resultados do experimento sugerem que o *Polyflow* é uma solução viável para interoperar dados de proveniência heterogênea gerados por diferentes WfMSs, tanto do ponto de vista de usabilidade quanto de desempenho.

A possibilidade de se trabalhar com *Polystores* como abordagem futura do *Smart-Bench* (27) é levantada como uma melhoria que pode prover uma integração no middleware permitindo que aplicações possam armazenar diferentes partes de seus dados em diferentes BDs subjacentes. Sendo assim, os autores dão a entender que o projeto de realizar uma integração da ferramenta implementada pode vir a acontecer devido a essa questão ser tratada como um ponto chave de observação feita por eles, afinal não existe um sistema que possa oferecer a melhor escolha, cada um tem suas vantagens.

Sob o contexto de revisar e comparar duas soluções para gerenciar dados multimodais, (33) propôs um estudo em quatro aspectos: fundamento teórico do gerenciamento de dados multimodais, armazenamento e estratégias para esses dados, linguagens de consulta entre modelos e, por fim, avaliação de consulta e otimização. Tudo isso baseando-se em duas soluções que gerenciam diretamente dados multimodais: um único sistema de banco de dados multimodal integrado ou um *middleware* totalmente integrado em vários armazenamentos de dados de modelo único.

Já em (44), é investigada a comunicabilidade de uma API do *Polystore BigDAWG* com o método *SigniFYIng* APIs. Neste contexto, foi analisado o potencial do método na caracterização das estratégias comunicativas do sistema *BigDAWG*. Além disso, o estudo

realizado foi precedido por um mapeamento sistemático simplificado na área de inspeção semiótica de IFAs. Ao final, os resultados deste trabalho indicaram a aplicabilidade do método *SigniFYIng* APIs no quesito de apoiar investigações no contexto da Computação Centrada no Ser Humano e para o estudo da comunicabilidade em sistemas *Polystore*.

Como um dos desafios e tecnologias apontados na seção 1, a integração do *streaming* de dados de baixa latência em arquiteturas de *data warehouse* tornou-se um aprimoramento importante para dar suporte a aplicativos modernos. Nessas arquiteturas, de acordo com (17), existem cargas de trabalho heterogêneas com ingestão de dados e consultas analíticas devem ser executadas com rigorosas garantias de desempenho. Além disso, o*data warehouse* pode consistir em vários tipos diferentes de mecanismos de armazenamento.

Sendo assim, foi proposto por (17) que o problema fundamental deste cenário acima apontado seria a colocação de dados, além dos diferentes cenários de carga de trabalho exigem diferentes designs de posicionamento de dados. Afinal, as condições de carga de trabalho mudam frequentemente. Com isso em mente, foram fornecidas evidências de que seria necessária uma abordagem dinâmica e orientada à carga de trabalho para a colocação de dados em *Polystores* com suporte à ingestão de dados de baixa latência.

Em (1) é levantado o caso de que as estruturas analíticas muitas vezes carecem de uma interface uniforme que permita acessar e aproveitar totalmente os vários modelos oferecidos pelo *Polystore*. Além do mais, é necessário garantir que a digitação das expressões algébricas construídas com operadores de manipulação de dados possa ser verificada e que o *schema* possa ser inferido antes de iniciar a execução dos operadores (*type-safe*). Sendo assim, foi utilizado um *Tensor Data Model* (TDM) que promove a união do aproveitamento da capacidade de modelagem de tensores adicionando-lhes *schemas* com operadores de manipulação de dados. Ao final do experimento foi fornecido aos usuários um mecanismo de inferência de *schema* e *type-safe* que garante a corretude funcional.

A abordagem feita em (1) traz a importância dos sistemas *Polystore* para lidar com a diversidade e o volume de *Big Data*. Alguns pontos são interessantes de serem destacados do trabalho relacionado pelos autores, tais como a tabela 5, dada abaixo, apresenta:

Esses pontos destacados acima mostram a preocupação voltada à segurança de tipos, uniformização de *schemas*, modelagem, uma inovação trazida pelo uso de tensores no contexto de *Polystores*, combinando capacidades de modelagem com operadores de manipulação de dados para melhorar a eficiência e a mantenibilidade das análises de *Big Data*.

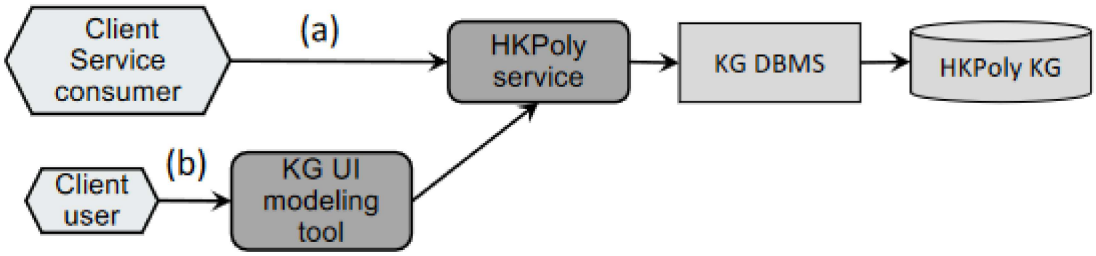
Em (9) o cerne da proposta é trazer uma arquitetura *Polystore* que fornece aos usuários uma única camada para acesso aos dados, encapsulando a heterogeneidade do armazenamento de dados, localização e ligação de dados usando mapeamentos e proveniência de dados. Neste trabalho, têm-se a arquitetura, nomeada de *Hyperknowledge Polystore* (*HKPoly*) que usa o metamodelo *Hyperknowledge* em sua implementação para representar os *schemas* de dados *HKPoly*.

A figura 5 exemplifica a interação do usuário ao acessar a arquitetura proposta de (9), que dá suporte de serviço para gerenciar metadados de domínios e armazenamentos de dados remotos. Ainda na figura 5, o caminho de (a) representa uma aplicação cliente ou opta-se pelo caminho (b) indo para uma interface de usuário. Independente do caminho, os metadados são armazenados em um *HKPoly Knowledge Graph* (ou HKPoly KG), ou seja, em um Grafo de Conhecimento.

Em (5) têm-se a preocupação de trazer os aplicativos de *big data*, pois estes, por sua

Ponto	Descrição
Necessidade de *Polystores*	Muitos *frameworks* analíticos não possuem uma interface uniforme para acessar e tirar proveito dos diferentes modelos de dados oferecidos pelos *Polystores*.
Desafios de Tipagem e Inferência de *Schema*	Importância da garantia de segurança de tipagem, através de expressões algébricas construídas com operadores de manipulação de dados.
Modelo de Dados com Tensores	A proposta dos autores é, através do uso de tensores, trazer um modelo de dados. Tensores são objetos matemáticos abstratos que podem incorporar relações complexas entre entidades.
Implementação em *Scala* e *Spark*	Os autores desenvolveram uma implementação utilizando *Scala* e *Spark*, que oferece um mecanismo de inferência de esquema e segurança de tipagem.

Tabela 5 – Principais pontos do trabalho realizado por (1).



– Figura 5 - Interação do cliente ao utilizar a arquitetura *HKPoly* (9).

vez, geralmente usam diversos conjuntos de dados em conformidade com diferentes modelos de dados. Esses conjuntos de dados são hospedados em armazenamentos heterogêneos, cada um otimizado para tipos específicos de processamento de dados. Sendo assim, os autores viram uma possibilidade de trabalhar nesse contexto utilizando *Polystore*.

Os principais pontos trabalhados em (5) é o que tratar algumas limitações atuais, tais como:

- Aplicativos que utilizam *Polsytore* precisam ter informação de onde e como os dados são armazenados;
- Limitação de flexibilidade e eficiência;
- Utilização de redundância.

Com o objetivo de trazer otimização transparente de consultas, a proposta de (5), traz o ESTOCADA, que visa otimizar consultas utilizando de forma transparente a melhor combinação de dados armazenados e capacidades de processamento em diferentes armazenamentos de dados. Além disso, a proposta aproveita os avanços na reescrita

de consulta baseada em visualização sob restrições. Isso envolve o uso de visualizações predefinidas (resultados de consultas armazenados para uso futuro) para reformular e otimizar novas consultas. Alguns benefícios foram observados, como a melhoria de desempenho, além da facilidade de uso, pois não há especificação manual que os usuários precisem fazer na aplicação.

Através de estudos de caso em cenários reais, a proposta de (63) traz o *ScalarDB* como um gerenciador de transações independente de banco de dados, o que significa que ele pode gerenciar transações em diferentes tipos de BDs sem depender de seus recursos transacionais nativos. Sendo assim, permite que transações abranjam vários BDs distintos sejam executadas eficientemente.

A proposta do *ScalarDB* é de fornecer uma garantia de correção, por meio de serialização estrita, garantindo que as transações pareçam ser executadas em uma ordem sequencial. Ou seja, permite que a otimização de desempenho ocorra pois inclui vários aprimoramentos de desempenho para lidar com transações de maneira eficiente em vários bancos de dados. Os desempenhos alcançados são classificados pelos autores como razoáveis. Sendo assim, pela escalabilidade quase linear, permite que a implantação da proposta seja plausível para aplicações reais.

Por outro lado, a abordagem *TyphonML* de (10), traz uma linguagem e ambiente de modelagem projetados para facilitar o *design* e a implantação de arquiteturas híbridas de persistência de dados que combinam bancos de dados relacionais e NoSQL.

Até aqui observa-se a tendência de desafios intrínsecos à BDs Distribuídos, utilização de *Polystores*, melhorias de desempenho, cenários reais, linguagens de modelagem, etc.

Voltando, em (10) têm-se uma especialização, através de uma linguagem, que permite aos modeladores especificar os dados a serem persistidos em arquiteturas híbridas, abstraindo os detalhes das tecnologias de BDs subjacentes. Sendo assim, também tem como foco trazer um ambiente que ofereça suporte ao processo de modelagem, além de auxiliar no gerenciamento das complexidades associadas à persistência de dados híbridos.

Apesar de *Polsytore* não ser o foco principal trabalhado em (55), temos uma abordagem de um modelo *Fog-to-Cloud* (F2C) que realiza processamento de dados pertos da *Edge*, propiciando melhorias em serviços sensíveis à latência, de forma que, pela necessidade de gerenciamento de dados em tempo real, os sistemas distribuídos proporcionem adequação e segurança.

Sendo assim, em (55) a solução é modelar uma arquitetura para construir um BD distribuído seguro especificamente para sistemas F2C. Esta solução visa responder aos desafios associados à gestão de dados num ambiente F2C distribuído. Os autores realizaram testes e validações de desempenho e constataram que a validação de comparação de modelos tradicionais com a proposta deles entrega uma eficácia, trazendo melhorias no gerenciamento de dados.

Indo para o lado de conectividade temos o trabalho realizado por (50) que foca em ambientes de reconhecimento facial, porém, além disso, que possuem restrições de prazos estritos. O cenário de aplicação da proposta dos autores é beneficiar os setores comercial e policial, permitindo assim que robôs desempenhem funções de assistência, busca, salvamento, etc. Sendo assim, a arquitetura foi projetada para oferecer suporte nesse cenário, isto é feito através de um modelo matemático representando unidades estruturadas de dados em uma série temporal, tudo numa arquitetura focada em *Cloud-Edge*.

No trabalho realizado por (7) são propostas duas versões de uma plataforma de

Cloud IoT chamada ENG-KPS *Cloud*. Essas versão foram desenvolvidas para diversos projetos de pesquisa e serviço seja para universidades como em comunidades colaborativas. Uma das versões, denominada fVSS, melhora um BD tradicional para se tornar um BD distribuído seguro, garantindo privacidade, disponibilidade e integridade dos dados durante o armazenamento, computação e transferência. Os experimentos realizados demonstram a eficiência tanto da fVSS quanto das plataformas ENG-KPS *Cloud*.

Nota-se que muitos trabalhos citados propõem-se a trazer soluções competitivas em relação a desempenho, focando em melhorias do que os serviços tradicionais de *Cloud*, *Fog* e *Edge* proporcionam.

A abordagem utilizada por (21) retrata a recuperação de dados *Ciphertext* em conjuntos de dados heterogêneos no contexto de IoT assistida pela *Cloud*. Trazendo assim um problema crítico que as soluções apontadas pelos autores não conseguiram resolver completamente. Para solucionar essa questão, os autores propõem um sistema de recuperação de dados transparente, que independe da linguagem de programação, das plataformas acessíveis e da heterogeneidade dos conjuntos de dados.

Ao analisar o trabalho acima exposto, apesar de não citar claramente a utilização de *Polystore*, a ideia proposta pelos autores remete ao que um *Polystore* faz, de maneira geral. A proposta de (21) combina um *middleware* de acesso de integração com um sistema de BD em *Cloud*. Esse *middleware* permite consultas entre linguagens e plataformas diferentes, além de gerenciar a integração de dados em BDs heterogêneos para realizar consultas.

Em (53) alguns pontos são relevantes como o contexto de ambientes de computação que utiliza a *Cloud* no sentido de permissão de dados e aplicações. Essa permissão é armazenada e implantada em infraestruturas espalhadas ao redor do mundo. Por conta desse espalhamento geográfico, utilizar esses sistemas distribuídos pode causar problemas relacionados a desempenho. O ponto forte deste trabalho é medir e compreender a performance desses sistemas de computação *Cloud*.

A plataforma utilizada por (39) traz um conjunto de componentes containerizados que são replicados em diferentes níveis *Cloud*, *Fog* e *Edge*. Sendo assim, mecanismos de tolerância a falhas são implementados por meio de replicação. *Apache Kafka* é um *framework* que fora usado na abordagem para distribuir mensagens entre os diferentes níveis da arquitetura, ajudando a gerenciar a comunicação de forma eficiente. Um dos focos dos autores é trazer mais próxima a confiabilidade de aplicações IoT.

Reiterando o assunto de performance, também sobre *Cloud* e *Fog* e sistemas IoT, o trabalho realizado por (62) traz um estudo de comparações de performance em ambientes dinâmicos que utilizam *Fog* a fim de tratar limitações da aplicação. Os autores consideraram armazenar dados em memória, afinal esta prática ofereceu benefícios consideráveis de desempenho devido à natureza transitória dos dados.

Os nodos *Edge* podem vir a falhar ou desconectar, sendo assim a abordagem de BDs Distribuídos é usada para garantir a durabilidade e a disponibilidade dos dados. No final da avaliação de performance, feita por (62), foram comparados o *Redis* e *Apache Ignite* e em alguns pontos o *Redis* foi mais satisfatório em termos de desempenho, no entanto o *Apache Ignite* mostra-se melhor em escalabilidade com o aumento do número de operações, sendo uma boa escolha para cenários com alta carga de trabalho.

No trabalho realizado em (30), foi apresentado um mecanismo de consulta que aborda os desafios do processamento paralelo de consultas *multistore*. Com o intuito de

preservar a expressividade das linguagens de consulta e *script* dos armazenamentos de dados subjacentes, foi criada a abordagem poliglota (*Polystore*) fornecida pela linguagem de consulta *CloudMdsQL*. Para facilitar o processamento de consultas paralelas, foi incorporada à abordagem poliglota o *LeanXcale Distributed Query Engine* (DQE), que fornece um mecanismo de consulta com paralelismo intra-consulta e intra-operador operando em uma interface SQL padrão.

O resultado obtido no trabalho acima pode ser destacado em alguns pontos:

- Abordagem de suporte de plataformas de processamento distribuído, como *Apache Spark*;
- Permitir o uso *ad-hoc* de operadores de mapa/filtro/redução (MFR) definidos pelo usuário como subconsultas;
- Permitir empurrar para baixo predicados da consulta (por exemplo, para condições de junção de ligação) e a recuperação paralela de resultados intermediários;

2.3 CONSIDERAÇÕES DO CAPÍTULO

Ao mapear todos os trabalhos relacionados, juntamente aos resumos, resultados e propostas apresentados por cada um, cria-se a tabela 6 (que pode ser vista na próxima página) onde são apresentados os tópicos principais de suas propostas.

Ao comparar a tabela 6 com a proposta deste trabalho, observa-se que a aplicação do *Polystore* na *Fog* é um tema que ainda não recebeu uma atenção significativa na literatura acadêmica. Embora existam estudos relacionados ao *Polystore* e à *Fog* em pesquisas acadêmicas e científicas, a combinação específica desses conceitos dentro do escopo deste trabalho parece ser pouco explorada. Disso, têm-se uma oportunidade única e desafiadora para propor, investigar e implementar soluções inovadoras nessa área.

Referência	**Principais Aspectos Abordados**
(43)	Uso de *Fog Computing* para atender requisitos sensíveis ao tempo.
(51)	Desafios de lidar com dados heterogêneos e sistemas de integração de dados.
(15)	Comparação de arquiteturas de *Big Data*, incluindo *Polystore*.
(40)	Proposta de união entre *S-Store* e *Polystore* para requisitos específicos.
(22)	Ferramenta *Polyflow* para integrar bases de dados heterogêneas.
(27)	Discussão sobre a integração de *Polystores* em ambientes SmartBench.
(33)	Revisão de soluções para gerenciamento de dados multimodais.
(44)	Análise comunicativa de APIs de *Polystores* usando método SigniFYIng APIs.
(17)	Proposta de posicionamento dinâmico de dados em *Polystores* para baixa latência.
(1)	Uso de *Tensor Data Model* para uniformizar acesso e garantir segurança de tipos em *Polystores*.
(9)	Propõe uma arquitetura *Polystore* (*HKPoly*) que encapsula a heterogeneidade do armazenamento de dados, localização e ligação de dados. Usa o metamodelo *Hyperknowledge* para representar *schemas* de dados.
(5)	Apresenta o ESTOCADA, uma abordagem para otimizar consultas em *big data* usando *Polystore*. Oferece melhorias de desempenho e facilidade de uso.
(63)	Introduz o *ScalarDB*, um gerenciador de transações independente de banco de dados que permite transações eficientes em vários BDs distintos. Realiza estudos de caso em cenários reais.
(10)	*TyphonML*: Linguagem e ambiente de modelagem para arquiteturas híbridas de persistência de dados.
(55)	Modelo *Fog-to-Cloud* para processamento de dados próximos à *Edge*, e arquitetura de BD distribuído seguro.
(50)	Arquitetura para reconhecimento facial em *Cloud-Edge* com restrições de prazos.
(7)	Duas versões de plataforma IoT para garantir privacidade e segurança em BD distribuído.
(21)	Sistema de recuperação de dados transparente para IoT assistida pela *Cloud*.
(53)	Avaliação de desempenho em ambientes de computação *Cloud* distribuídos geograficamente.
(39)	Plataforma para replicação de componentes containerizados em diferentes níveis *Cloud*, *Fog* e *Edge*.
(62)	Estudo de comparações de performance em ambientes dinâmicos usando *Fog*, com foco na armazenagem de dados em memória.
(30)	Abordagem de processamento paralelo de consultas *multistore* usando linguagem *CloudMdsQL* e *LeanXcale Distributed Query Engine*.

Tabela 6 – Comparação de Referências sobre IoT, *Polystores* e Computação Distribuída.

3 PROPOSTA DA ARQUITETURA *FOG-CLOUD POLYSTORE*

No presente capítulo, é demonstrada a proposta de pesquisa e desenvolvimento que serve como a essência desta dissertação.

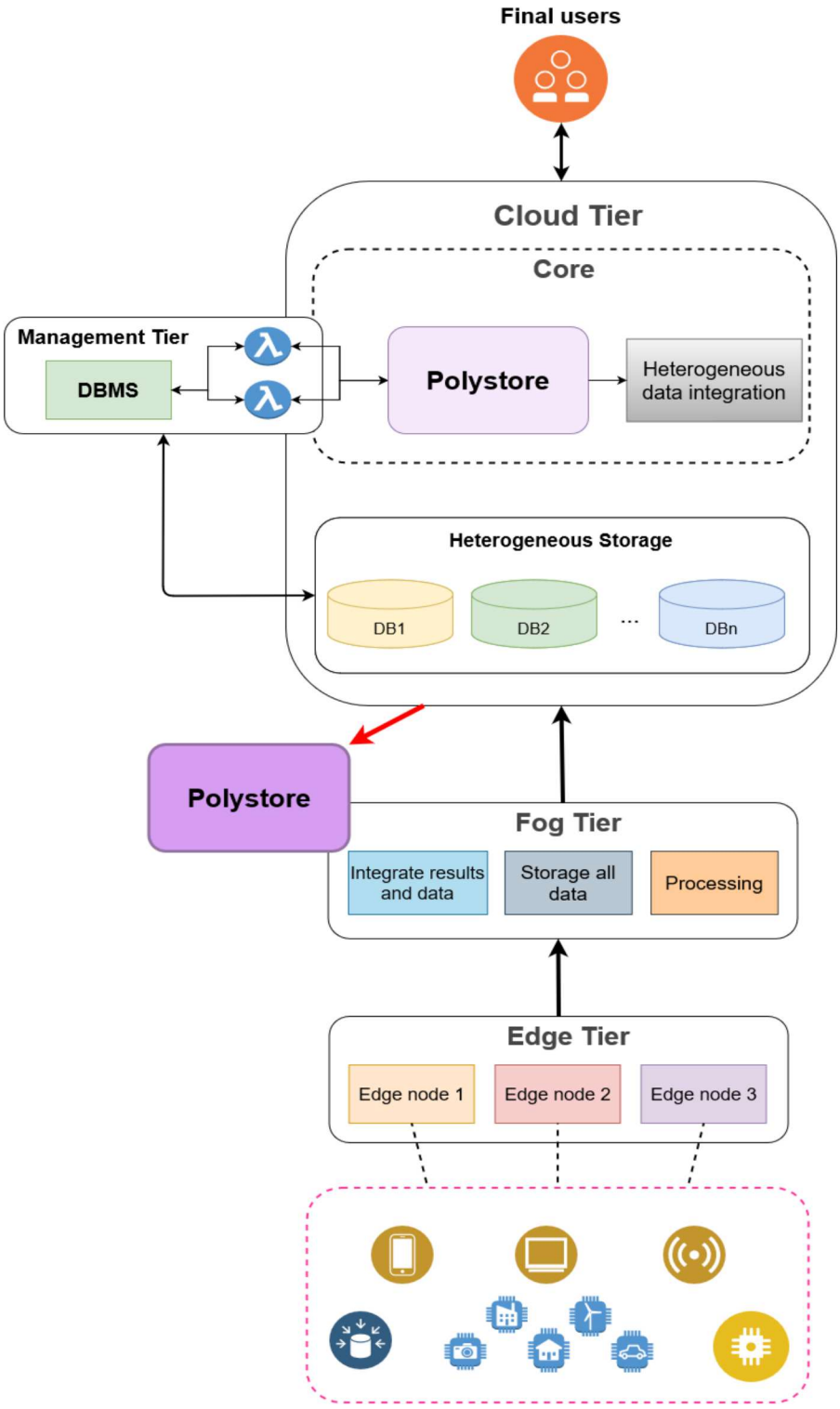
Como mencionado previamente na seção **2.1.4**, um Banco de Dados "Poliglota"ou *Polystore* é concebido para administrar e armazenar dados empregando múltiplos modelos de dados ou SGBDs em um único sistema. Esta abordagem é particularmente útil quando lidamos com dados heterogêneos que requerem diferentes estruturas de armazenamento e gerenciamento. Por exemplo, pode ser necessário armazenar dados estruturados, semi-estruturados e não estruturados simultaneamente, e um banco de dados poliglota permite essa flexibilidade.

Por outro lado, a computação *Fog* amplia a computação em *Cloud* até a periferia da rede, aproximando os recursos computacionais da fonte de dados. Isso significa que, em vez de depender exclusivamente de recursos em *data centers* remotos, a computação em *Fog* leva parte do poder de processamento e armazenamento para dispositivos mais próximos dos usuários ou dos pontos de coleta de dados. Isso pode resultar em uma redução significativa na latência e no tráfego de dados na rede, tornando a computação mais eficiente em cenários onde a latência é crítica, como aplicações de IoT e sistemas de tempo real.

A integração do armazenamento *Polystore* com a *Fog* pode, por conseguinte, proporcionar vantagens em termos de eficiência de processamento de dados e redução de latência. Ao utilizar um banco de dados *Polystore* em conjunto com a computação em *Fog*, podemos distribuir a carga de processamento e armazenamento de forma mais eficaz, otimizando a utilização de recursos e melhorando o desempenho geral do sistema. Isso é especialmente benéfico em ambientes onde a quantidade de dados é grande e a latência é crítica, como em aplicações de análise em tempo real e processamento de grandes volumes de dados.

3.1 ARQUITETURA

A Figura 6 (que pode ser vista na próxima página) representa uma visão geral da proposta na qual o objetivo é facilitar o uso de BDs para usuários experientes e não experientes. Esta etapa será feita através da tecnologia de integração de dados que será realizada na camada Fog. Para chegar à camada Core, os dados passam por processamento dentro da arquitetura.



– Figura 6 - A proposta tem três níveis essenciais: armazenamento de dados e acesso, processamento e nodos *edge*. Figura da autora.

No ecossistema da computação em *Cloud*, o processamento de dados pode demandar um tempo considerável, chegando a durar dias, devido à imensa carga de trabalho que a *Cloud* gerencia, sendo esta a camada mais externa à rede. Por isso, acessar e operar na *Cloud* tende a ser mais dispendioso, especialmente quando se trata de operações típicas de sistemas IoT e demandas por mobilidade, distribuição geográfica, baixa latência e tomada

de decisões rápidas. Sendo assim, em teoria, torna-se mais custoso implementar o conceito de *Polystore* diretamente na *Cloud*. A proposta é apresentada na figura 6.

Os dados são gerados por uma diversidade de dispositivos, desde servidores e computadores até dispositivos inteligentes como pulseiras, celulares e outros eletrônicos habilitados para IoT. Esses dados são inicialmente coletados na camada denominada *Edge Tier*, onde cada nó corresponde a um dispositivo que gerou os dados.

A arquitetura proposta oferece aos usuários finais acesso às bases de dados por meio do *Core*; para acessar a *Cloud*, é necessário utilizar o *Polystore*, que integra os dados provenientes de diferentes tipos de bancos de dados heterogêneos.

Na Figura 6, a seta vermelha destacada representa o uso do *Polystore* na *Fog*. Em teoria, conforme discutido na seção **2.1.4** deste trabalho, a *Cloud* executaria o *Polystore*. Entretanto, a proposta introduz a colaboração da computação *Fog* como uma camada de execução, integração e pré-processamento de dados.

Os dados são transferidos para o *Fog Tier*, onde ocorrem os processos de integração dos resultados dos dados coletados e obtidos do *Edge Tier*. Nesta camada, acontece o refinamento, incluindo a etapa crítica de fusão dos Bancos de Dados, permitindo a integração das bases.

Por fim, o *Cloud Tier* é responsável por entregar os resultados das pesquisas realizadas pelos usuários finais. Ele contém a abstração *Polystore* que analisa os resultados refinados das camadas anteriores e os entrega ao usuário final utilizando uma linguagem unificada. Esse processo aumenta a transparência da aplicação, além de aliviar a carga da *Cloud*, que se concentra em entregar os resultados, enquanto as demais camadas próximas à borda da rede realizam os processamentos mais intensivos.

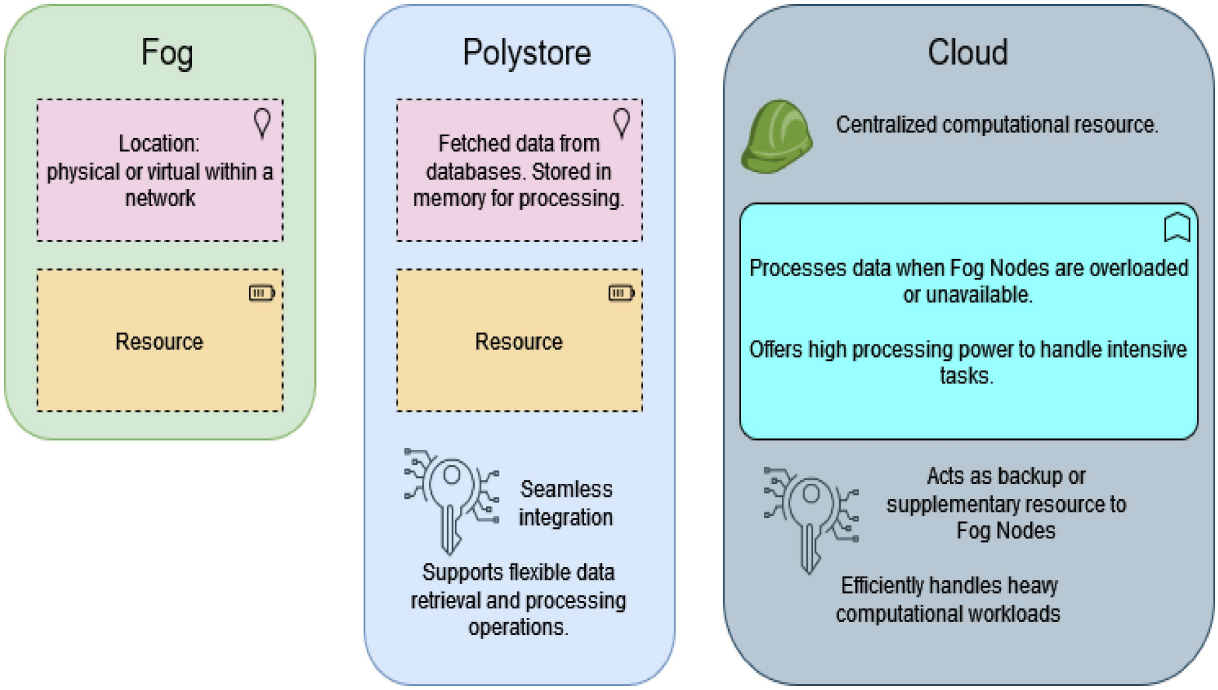
3.1.1 APLICAÇÃO DA PROPOSTA

A execução de um *Polystore* em partes na *Cloud* e em partes na *Fog* representa uma abordagem híbrida para lidar com a complexidade e as demandas de dados distribuídos. Nesta arquitetura proposta, algumas partes do *Polystore* são implantadas em ambientes de *Cloud*, enquanto outras são distribuídas em dispositivos de *Fog*, mais próximos dos pontos de geração e consumo de dados.

Através da aplicação da arquitetura, espera-se que na *Cloud*, as partes do *Polystore* podem se beneficiar da escalabilidade, recursos computacionais robustos e armazenamento confiável oferecidos por provedores de serviços *Cloud*. Isso é especialmente útil para o processamento intensivo de dados e consultas complexas que requerem grandes quantidades de recursos computacionais.

Por outro lado, a implantação de partes do *Polystore* na *Fog* oferece vantagens como baixa latência, redução de sobrecarga de rede e maior privacidade dos dados, uma vez que os dados são processados e armazenados mais próximos dos dispositivos de origem. Isso é essencial para aplicativos de tempo real e sensíveis à latência, como IoT, onde a rápida tomada de decisões é crítica.

A integração entre as partes do *Polystore* na *Cloud* e na *Fog* é realizada por meio de comunicação eficiente e sincronização de dados. Essas informações podem ser vistas na seção 4. Por fim, as consultas podem ser distribuídas e otimizadas entre os componentes, aproveitando as vantagens de cada ambiente para garantir um desempenho ideal e uma experiência do usuário aprimorados. Esse processo pode ser visualizado melhor na figura 7.



– Figura 7 - Características específicas para a concepção da proposta e respectivas ações esperadas de cada camada. Figura da autora.

A figura 7, representa um resumo de passos para a aplicação da proposta. Alguns pontos são importantes de serem destacados. São eles:

- ***Polystore***:
  - **Localização**: Os dados são buscados em bancos de dados e armazenados em memória para processamento.
  - **Recurso**: Oferece integração e suporta operações flexíveis de recuperação e processamento de dados.
  - ***Key Points*** (Pontos-chave):
    - Integração contínua de múltiplos bancos de dados para suportar diversos tipos de dados.
    - Facilita a manipulação eficiente de dados em diferentes formatos e fontes.
- **Computação *Cloud***:
  - **Papel**: Oferece recursos computacionais centralizados.
  - **Função**: Processa dados quando os nodos de Fog estão sobrecarregados ou indisponíveis, fornecendo alta potência de processamento para tarefas intensivas.
  - ***Key points***:
    - Atua como recurso de backup ou suplementar para nodos de Fog.
    - Lida eficientemente com cargas de trabalho computacionais pesadas.
- **Interação entre *Fog* e *Cloud***:

  - **Sinergia**: *Fog* e *Cloud* trabalham juntos para otimizar o processamento de dados.
  - **Complementaridade**: A *Cloud* oferece suporte quando o *Fog* está sobrecarregado, garantindo a continuidade das operações.

- **Benefícios Gerais**:
  - **Latência reduzida**: O processamento de dados mais próximo dos dispositivos de borda reduz a latência e melhora a responsividade do sistema.
  - **Eficiência operacional**: A distribuição inteligente de tarefas entre *Fog* e *Cloud* otimiza o uso de recursos e reduz custos.
  - **Resiliência**: A capacidade de alternar entre *Fog* e *Cloud* oferece redundância e tolerância a falhas.

3.2 DADOS UTILIZADOS

Conforme visto em 1 e 2, a crescente utilização de IoT abrange uma ampla gama de setores na sociedade moderna, desde a saúde, agricultura, até às cidades inteligentes, por exemplo. No contexto deste trabalho optou-se por utilizar dados relativos à saúde, porém, vale ressaltar que poderiam ser escolhidos os mais diversos cenários de aplicação.

Quando se trata de dados reais de informações médicas, é imperativo implementar medidas robustas de segurança para proteger os dados tanto quanto estão em uso e trafegando na rede, quanto quando estão ociosos.

Assegurar que esses dados atendam aos requisitos de privacidade é uma premissa fundamental, especialmente ao lidar com informações confidenciais ou sujeitas a regulamentações específicas. Embora os experimentos conduzidos tenham sido realizados localmente em uma máquina, não houve a necessidade imediata de considerar questões de segurança, como criptografia, entre outras medidas.

No entanto, é crucial ressaltar que, em ambientes reais, a proteção não se limita apenas a informações sensíveis de pacientes, mas também se estende a dados financeiros, registros de execução, senhas e muito mais.

Para esta proposta, foi utilizada a base de dados denominada *MIMIC-III*(**?**)[3] em sua versão *demo 1.4*, que está disponível gratuitamente. Esta base compreende dados desidentificados relacionados à saúde, associados a mais de 40.000 pacientes que estiveram em unidades de cuidados intensivos do *Beth Israel Deaconess Medical Center* entre 2001 e 2012.

Como mencionado anteriormente, embora as informações sensíveis não estejam identificadas, esta base de dados contém detalhes cruciais sobre o atendimento de pacientes reais e, portanto, requer credenciamento prévio para acesso. A versão *demo 1.4* é disponibilizada gratuitamente para fins de testes e análises clínicas por profissionais da saúde, dispensando a necessidade de credenciamento. No âmbito deste estudo, os dados foram tratados de maneira isolada nos bancos de dados selecionados para serem integrados ao *Polystore*, configurados de forma a simular informações provenientes de dispositivos IoT.

[3] Pyshionet

3.3 DESENVOLVIMENTO

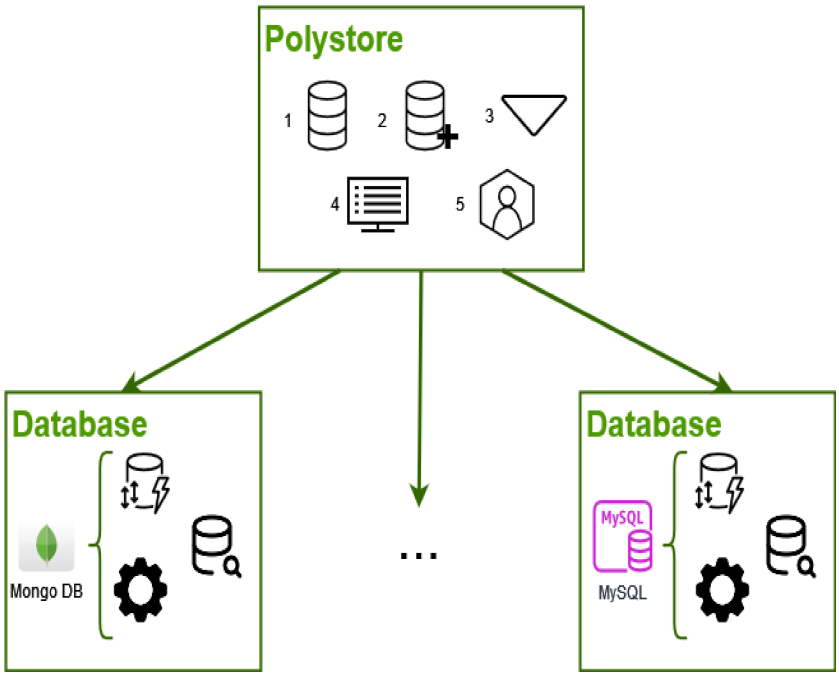
Foi necessário entender quais seriam os requisitos os quais integrariam a arquitetura proposta, ou seja, identificar dentro da aplicação quais seriam os requisitos e os tipos de modelos de dados necessários, tais como, por exemplo, relacional, documental, gráfico, série temporal, entre outros.

Além disso os passos de definição do *Polystore* foram fundamentais para que houvesse suporte aos modelos de dados que foram utilizados, ao final esse sistema precisa permitir integrar diferentes tipos de BDs. O próximo passo a ser levado em consideração seria a definição da arquitetura do ambiente, conforme vista anteriormente em 3.1.

O próximo passo do desenvolvimento foi o de decidir sobre particionamento e replicação de dados. Para a *fog*, alguns dados são interessantes de se manter próximos à *edge* para processamento de baixa latência enquanto que o processo de replicar dados importantes fica para a *cloud* a fim de se ter um *backup* e possibilitar posterior análise.

3.3.1 *MERGE*

Para fins de exemplo, a figura 8 mostra os tipos de bancos de dados os quais foram utilizados para experimento. Vale observar que, ao lidar com BDs de fontes heterogêneas, isso não implica que serão trabalhados com diferentes tipos de BDs. Neste trabalho, porém, optou-se por distribuir os dados em tipos diferentes de BDs, tais como o *MongoDB* (NoSQL)[5], *MySQL* (SQL)[6] e um repositório onde uma parte de dados ficará para comparação com as demais bases de dados.



– Figura 8 - Um *schema* básico da implementação da comunicação do *Polystore* com os respectivos BDs. Figura da autora.

[5] MongoDB
[6] MySQL

A numeração de 1 a 5 que consta na figura 8 representa os passos fundamentais da construção do *Polystore*. Ao todo são realizadas duas etapas no *merge* que são:

- Polystore:
  - 1: Verificação de Banco de Dados que verifica entre os BDs cadastrados que estão disponíveis;
  - 2: Adicionando acesso aos BDs que retorna quais BDs e suas configurações estão disponíveis;
  - 3: Mesclar dados;
  - 4: Exposição;
  - 5: Permissão para acesso a consultas;
- Bancos de dados:
  - Configurações iniciais;
  - Acesso ao BD;
  - Pesquisa no banco de dados;
  - Formatação de saída para Polystore;

O processo de implementação da proposta começou com a configuração do ambiente de desenvolvimento. Foi optado pelo mesmo ser realizado dentro do Sistema Operacional *Windows 11*, processador AMD Ryzen 5 1600, linguagem de programação *Python*[6], além das tecnologias dos BDs mencionados anteriormente que foram usados, conforme a figura 8 mostra.

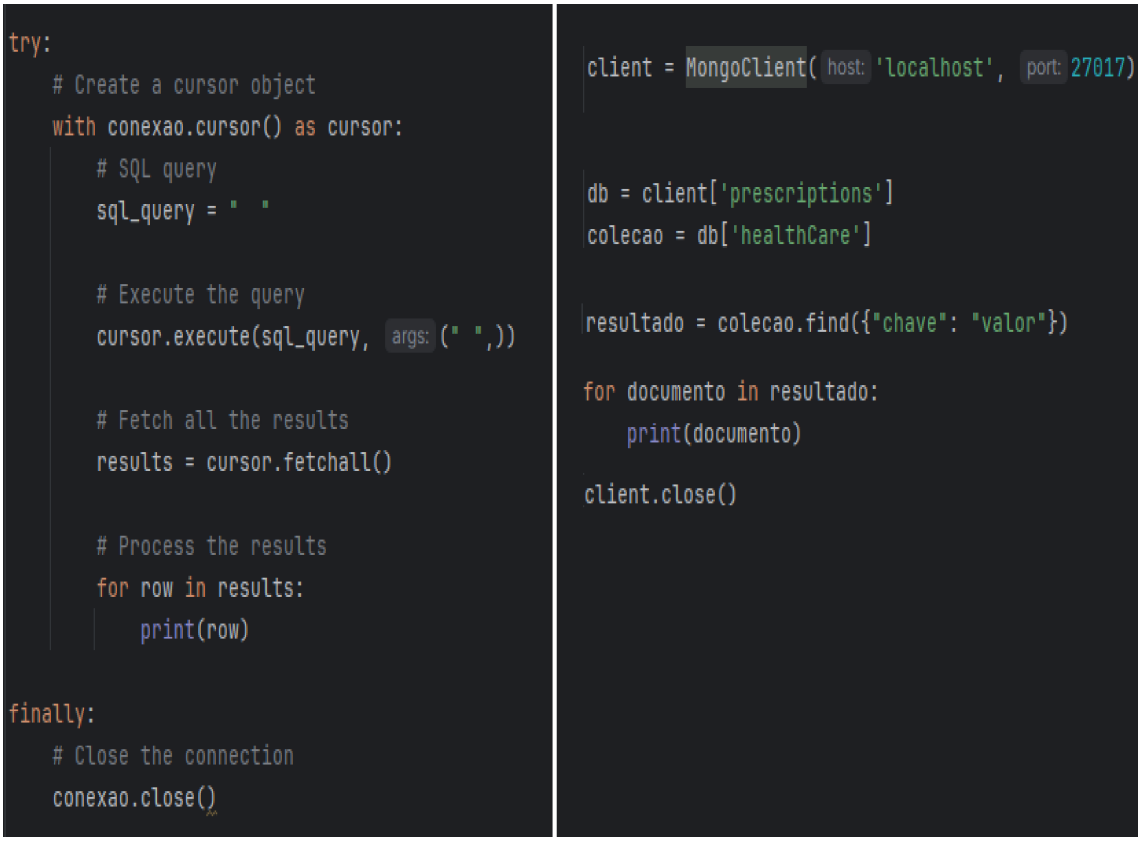
Na figura a seguir foram feitas as conexões necessárias dentro da aplicação, estas podem ser conferidas nos algoritmos abaixo.

A figura 9 mostra, de forma básica, como é realizada a conexão com as bases de dados, passo a passo. Basicamente há a etapa inicial que é fundamental que é a preparação dos dados: baixar o conjunto de dados, extrair e pré-processar os dados conforme necessário e, por fim, preparar os dados para inserção no *MySQL*, *MongoDB* e na base de dados heterogênea que fora utilizada.

Após a preparação dos dados, dá-se início à etapa de criação e configuração dos BDs:

- Criar banco de dados *MySQL* para armazenar dados estruturados.
- Definir tabelas para armazenar dados relevantes do conjunto de dados.
- Insirir os dados nas tabelas *MySQL*.
- Instalar e configurar o *MongoDB*.
- Definir coleções para armazenar dados semiestruturados ou não estruturados do conjunto de dados Mimic-III.
- Insirir os dados nas coleções do *MongoDB*.

[6] Python



```
try:
    # Create a cursor object
    with conexao.cursor() as cursor:
        # SQL query
        sql_query = "  "

        # Execute the query
        cursor.execute(sql_query, args: (" ",))

        # Fetch all the results
        results = cursor.fetchall()

        # Process the results
        for row in results:
            print(row)

finally:
    # Close the connection
    conexao.close()
```

```
client = MongoClient( host: 'localhost',  port: 27017)


db = client['prescriptions']
colecao = db['healthCare']

resultado = colecao.find({"chave": "valor"})

for documento in resultado:
    print(documento)

client.close()
```

– Figura 9 - Como as conexões dos BDs são feitas localmente: a esquerda a conexão com o MySQL e a direita a conexão com o MongoDB. Figura da autora.

- Converter dados relevantes do conjunto de dados em formato CSV (pois, assim estão dispostos os dados de forma originalmente) para inserir dentro da base de dados.

Após as etapas de criação e configuração são definidas quais as *queries* serão utilizadas para as buscas das três bases de dados. Essas consultas são unificadas numa classe dentro da aplicação chamada de *polystore_search*. Ao final da execução de busca, é dada a contagem de tempo da execução em segundos.

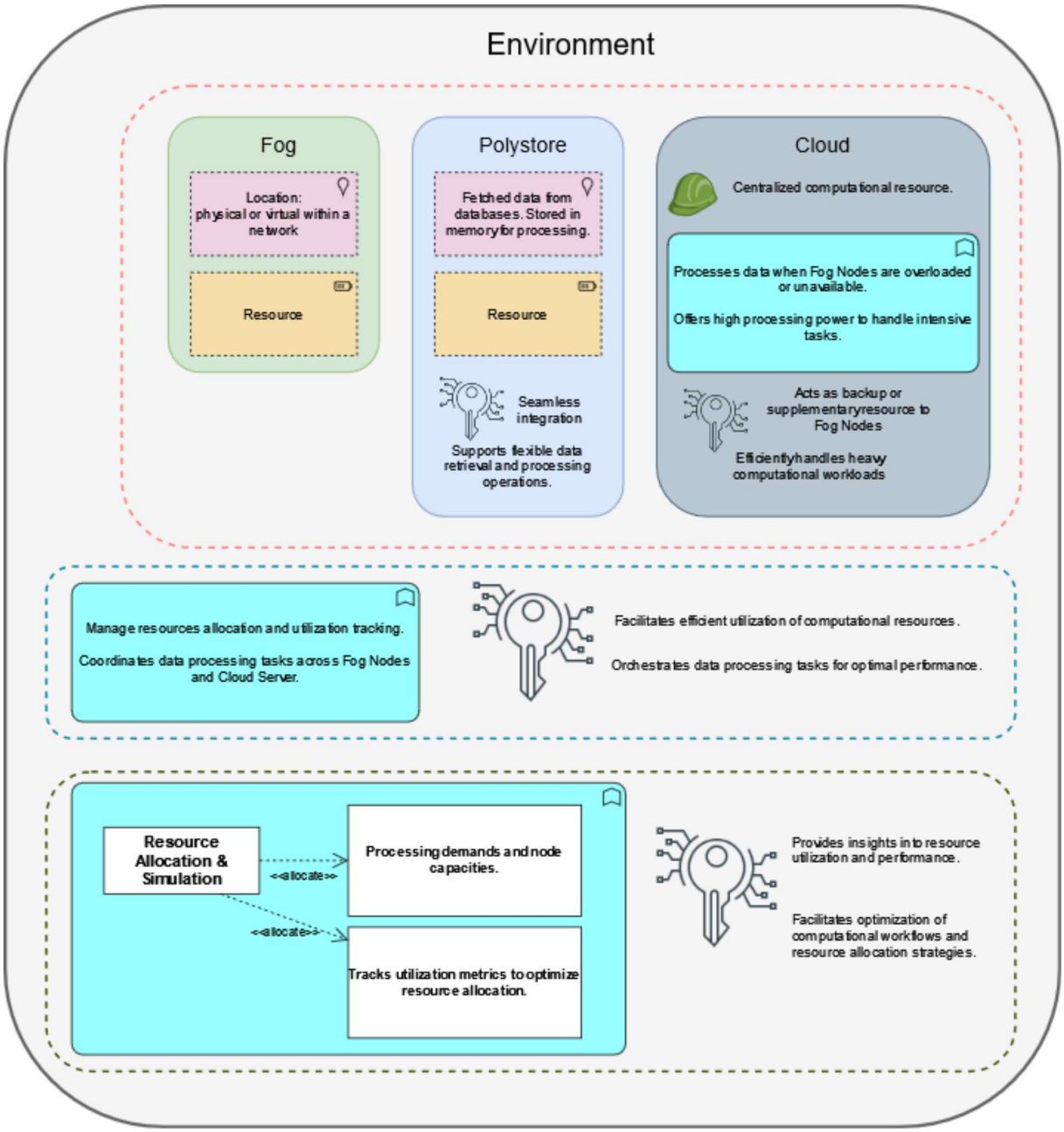
Foi utilizado o conceito de execução de função denominado *time.sleep* para suspender a execução por um período de tempo especificado. Essencialmente, a execução do *thread* atual tem uma pausa, permitindo que outros *threads* ou processos sejam executados durante esse período.

Alguns casos de uso do recurso *time.sleep* foram levados em consideração na proposta, tais como:

- Possíveis atrasos em processos através da simulação num caso de utilização da aplicação em tempo real.
- Controle da taxa de solicitações a APIs ou servidores para evitar sobrecarregá-los.

4 AMBIENTE E RESULTADOS EXPERIMENTAIS

Em cenários de saúde IoT, os requisitos de latência são frequentemente rigorosos, especialmente quando se trata de monitoramento, análise e resposta em tempo real. Para investigar e compreender essas demandas, foi criada uma simulação, que pode ser vista na figura 10, que reflete essa perspectiva em ambientes de *Cloud* e *Fog*.

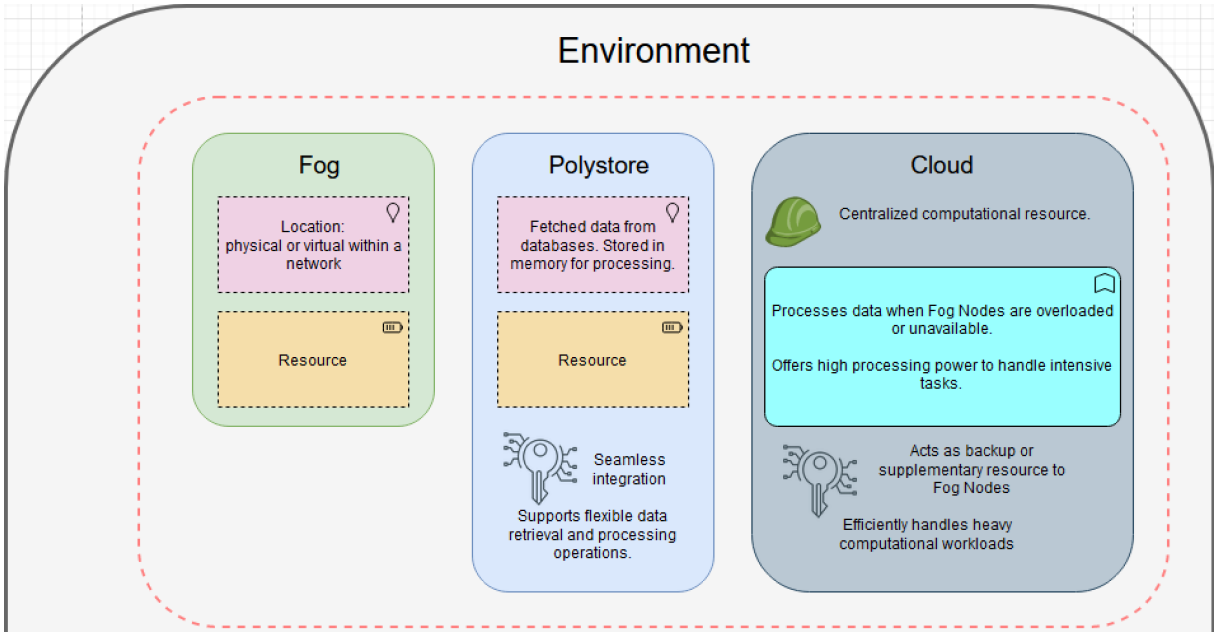


– Figura 10 - Representação gráfica do ambiente de simulação. Figura da autora.

A figura 10 representa todo o mecanismo de funcionamento esperado para o ambiente de simulação. A mesma será dividida em três partes, de modo a facilitar a compreensão de cada uma e melhor visualização.

A primeira parte, concentrada no topo da figura 10, representa a aplicação da

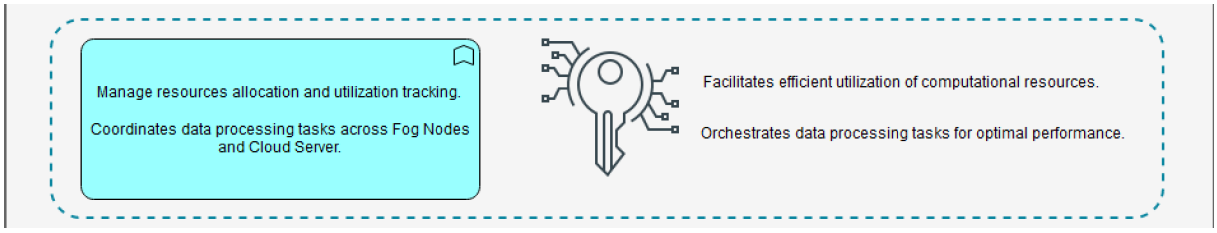
proposta, anteriormente explicada e mencionada em **3.1.1** na figura 7. Esse recorte pode ser melhor visualizado na figura abaixo:



– Figura 11 - Representação gráfica do ambiente de simulação e de sua aplicação. Figura da autora.

Já na segunda parte da figura 10, a parte central (conforme pode ser visualizada na figura 12), alguns pontos são importantes de serem destacados.

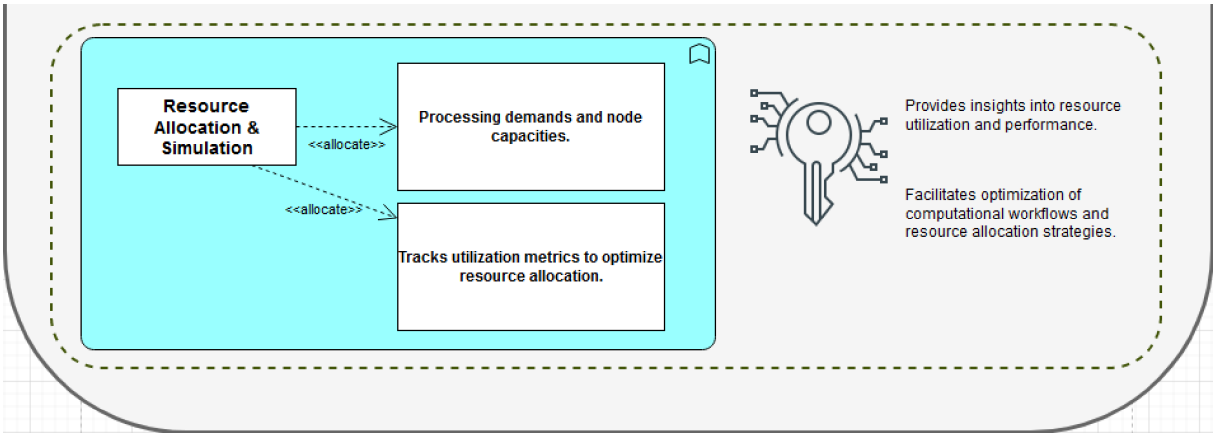
- Função: Gerencia a alocação de recursos e o rastreamento da utilização. Coordena tarefas de processamento de dados entre nodos de Fog e servidor em nuvem.
- *Key Points*:
  - Facilita a utilização eficiente de recursos computacionais.
  - Orquestra tarefas de processamento de dados para desempenho ideal.



– Figura 12 - Representação gráfica das funcionalidades e pontos-chave do ambiente. Figura da autora.

E, por fim, a parte três da figura 10, situada na parte inferior da mesma, representada na figura 13, exemplifica os seguintes pontos:

- Alocação de Recursos e Simulação:
  - Função: Avalia demandas de processamento e capacidades dos nodos. Rastreia métricas de utilização para otimizar alocação de recursos.
  - *Key Points*:
    * Oferece resultados sobre utilização de recursos e desempenho.
    * Facilita a otimização de cargas de trabalho computacionais e estratégias de alocação de recursos.



– Figura 13 - Representação gráfica das funcionalidades e pontos-chave de métodos de alocação. Figura da autora.

A tabela 7 descreve as principais funcionalidades do simulador desenvolvido para gerenciamento e alocação de recursos tanto dos nodos *Fog* como do servidor *Cloud*. O simulador é capaz de executar pesquisas em diferentes bases de dados, calcular a utilização de recursos e alocar recursos de maneira eficiente para atender às demandas de processamento. As funcionalidades do simulador são detalhadas na tabela abaixo, proporcionando uma visão geral das operações que ele pode realizar e seus respectivos papéis no ambiente simulado.

4.1 SIMULAÇÃO

Como mencionado anteriormente em 1.1, a latência representa uma medida crítica que corresponde ao tempo de tráfego da rede e pode indicar possíveis problemas na aplicação, dada a alta demanda por transferência de dados. Em aplicações de saúde, o valor médio de latência pode variar consideravelmente, influenciado por fatores como a distância geográfica entre os dispositivos IoT e os servidores na *Cloud*, o congestionamento da rede e a eficiência dos *pipelines* de processamento de dados. Normalmente, essa medida varia de dezenas a centenas de milissegundos. No entanto, dependendo do contexto, valores elevados de latência podem inviabilizar a resposta em tempo real, tão crucial para várias aplicações na área da saúde.

Nesta etapa de desenvolvimento, o *Polystore* é projetado para decidir de maneira dinâmica onde executar suas operações, seja na *Cloud* ou nos nodos *Fog*, com base em uma análise prévia de eficiência e desempenho. Como classificação, baseada nos conceitos

Funcionalidade	**Descrição**
Mongo Search	Executa uma consulta MongoDB com base na coluna e no valor fornecidos.
MySQL Search	Executa uma consulta MySQL com base na tabela e na chave fornecidas.
CSV Search	Executa uma consulta dentro de uma base de dados em relação a um campo e uma chave fornecidos.
Polystore Search	Executa consultas através de múltiplas bases de dados para uma determinada informação que se deseja obter.
FogNode	Representa um nodo *Fog* com seus respectivos atributos.
CloudServer	Representa um servidor *Cloud* com seus respectivos atributos.
Environment	Representa o ambiente com nodos, servidor, conexões e métricas.
Calculate Utilization	Dentro da funcionalidade *Environment*. Calcula a utilização dos nodos e do servidor com base na carga e no poder de processamento.
Allocate Resources	Aloca recursos para processar os dados e retornar o processamento deles, o local e resultados. Chama a funcionalidade *Polystore Search*.
Allocate and Calculate	Funcionalidade com o objetivo de alocar recursos provenientes do processamento dos dados e realizar o cálculo de utilização.
Initialize Fog Nodes	Inicializa um determinado número de nodos com localizações randômicas, bem como o poder de processamento.
Run Simulation	Roda a simulação de processamento, alocação de recursos e cálculos de utilização para um número determinado de iterações.

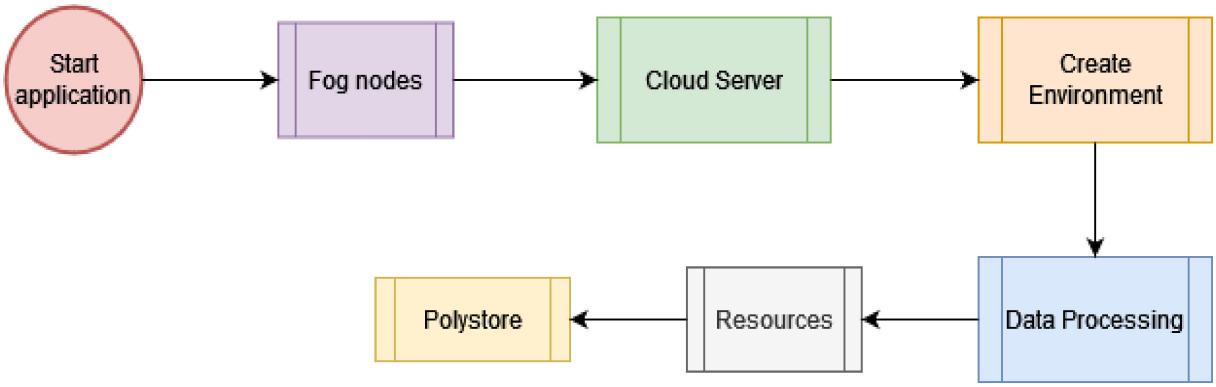
Tabela 7 – Relação das funcionalidades do simulador com suas respectivas descrições.

apresentados em **2.1.4**, a arquitetura desenvolvida atua como um *Polystore* híbrido operando em uma rede de computação distribuída composta por servidores *Cloud* e nodos *Fog*.

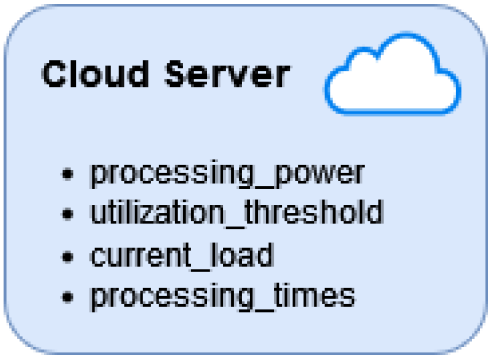
Este processo de decisão baseia-se em parâmetros como latência, largura de banda, carga de trabalho e requisitos de armazenamento. A arquitetura permite essa flexibilidade, caracterizando-se como um *Polystore* híbrido devido à sua capacidade de operar em múltiplas camadas de infraestrutura.

A figura 14 demonstra as etapas do processo de simulação que fora realizado para executar os experimentos.

Na simulação criada, a classe *CloudServer*, que pode ser vista de forma mais detalhada na figura 15, é indicada para tarefas que não são sensíveis à latência da rede. Uma aplicação potencial seria a realização de análises de dados a longo prazo, como a avaliação de tendências históricas do quadro de saúde de pacientes e a modelagem preditiva. Essas tarefas podem ser realizadas de maneira eficiente na *Cloud*, onde a latência não é uma preocupação primordial.



– Figura 14 - Etapas da simulação de uma rede *Cloud*, *Fog* e *Edge* em conjunto com *Polystore*. Figura da autora.



– Figura 15 - Funcionalidades do servidor Cloud. Figura da autora.

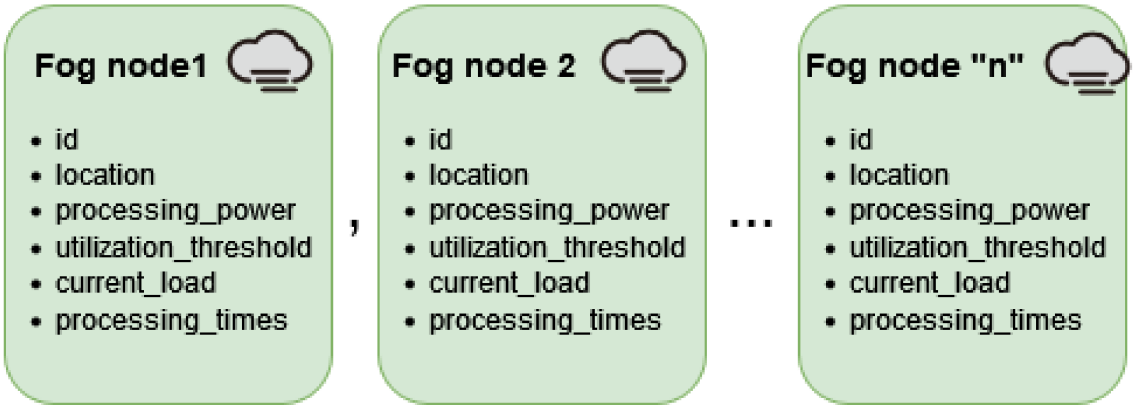
Resumidamente, cada característica do servidor *Cloud*, podem ser descritas abaixo:

- *Processing Power* (Poder de processamento): É a capacidade de um recurso de computação de processar dados. Na simulação, os nodos e o servidor possuem diferentes poderes de processamento.
- *Utilization Threshold* (Limite de Utilização): Este parâmetro determina quanto do poder de processamento de um nodo fog ou servidor pode ser utilizado antes de ser considerado totalmente utilizado.
- *Current Load* (Carga Atual): Se refere à carga atual de processamento que está sendo realizada por um determinado nodo ou pelo servidor. Especificamente, indica a quantidade de demanda de processamento que está sendo tratada pelo nodo ou pelo servidor no momento.
- *Processing Times* (Tempo de Processamento): Um parâmetro para acompanhar o tempo gasto na execução.

Na simulação, os nodos e o servidor possuem limites de utilização que os quais delimitam a capacidade de processamento. Se a carga atual exceder o limite de utilização, a solicitação de processamento poderá ser encaminhada para outro nodo ou servidor que estejam disponíveis. Sendo assim, tendo em vista os conceitos apresentados em **2.1.2**, os

nodos *Fog* normalmente têm poderes de processamento mais baixos em comparação com o servidor *Cloud*, tornando-os adequados para tarefas de processamento com requisitos computacionais mais baixos.

De modo análogo ao servidor *Cloud*, as funcionalidades de um nodo *Fog* são bem semelhantes. Essas funcionalidades podem ser melhor visualizadas na figura 16.



– Figura 16 - Como cada nodo *Fog* é representado dentro da simulação. Figura da autora.

Ao observar os nodos e o servidor, nota-se que ambos possuem limites de utilização que delimitam a capacidade de processamento. Se a carga atual exceder o limite de utilização, a solicitação de processamento poderá ser encaminhada para outro nodo ou servidor.

O parâmentro de carga é atualizado conforme novos dados são processados ou alocados para o nodo ou para o servidor. Ele é usado para acompanhar e controlar a carga de trabalho que está sendo atribuída a cada recurso de processamento. Sendo assim, quando a carga atinge um determinado limite, isso pode indicar que o recurso está sobrecarregado e pode precisar de redistribuição de carga ou otimização do sistema para manter um desempenho eficiente.

É importante ressaltar que, embora a *Cloud* seja adequada para essas tarefas menos sensíveis à latência, ela também desempenha um papel crucial na interação com a borda da rede (*Edge*) e na coordenação das operações entre a *Fog* e os dispositivos IoT. Assim, uma abordagem integrada, considerando tanto as capacidades da *Cloud* quanto as restrições de latência da borda da rede, é essencial para atender às necessidades diversificadas das aplicações de saúde IoT.

Voltando à figura 14, o início do simulador se dá na etapa *Start*, representada de vermelho na figura 14, onde são criados os nodos *fog*. Após a criação, esses nodos são inicializados na etapa *Fog nodes* em locais aleatórios e com poder de processamento também aleatório.

Ao inicializar o *Cloud Server*, é criado e ativado o servidor em *Cloud* com poder previamente estabelecido de processamento. Vale lembrar que esse poder de processamento pode ser alterado e até mesmo randomizado para evitar quaisquer casos repetidos, uma vez que a ideia da proposta é simular um ambiente de solicitação de dados médicos. Após essa etapa, é criado o ambiente em *Create Environment* onde são configurados os nodos *fog* e o servidor *Cloud*.

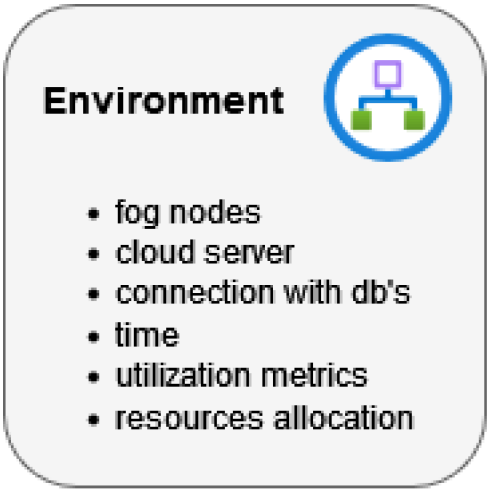
A etapa de *Data Processing* se dá como uma simulação de processamento onde são gerados exemplos de solicitações de processamento de dados.

A etapa *Resources* (em português: recursos) é nomeada como método que está contida dentro da classe *Environment*, onde são passados os dados como parâmetro. Ao ser iniciado, o método cria uma lista *available_fog_nodes* os quais são os que possuem o poder de processamento suficiente para lidar com a solicitação de dados que fora recebida. Além disso, o método *Resources* verifica se o poder de processamento do nodo *fog* é maior ou igual à demanda de processamento dos dados recebidos.

Por fim, temos a etapa de *Polystore*. Inicialmente ela é implementada no começo da aplicação, mas somente é solicitada ao final, após a etapa de simulação de ambiente *Fog* e *Cloud*. A conexão com os BDs permite que sejam buscados os dados sem a intervenção de outras aplicações rodando em segundo plano. Para tal, é necessário que, obviamente, os servidores dos BDs estejam habilitados para a permissão de acesso da aplicação ser aceita. Como existem três bases de dados: *MySQL*, *MongoDB* e uma base heterogênea que foi adicionada para fins de consulta extra, a etapa de *Merge* (vista na subseção **3.3.1**) inclui o caminho dos dados alocados na máquina de testes, além dos servidores conectados previamente ditos.

Caso não se tenha poder de processamento suficiente é avaliado se os nodos *fog* e o servidor *Cloud* conseguiram lidar com a solicitação de dados devido ao poder de processamento insuficiente, sendo assim, o método retornará uma mensagem indicando o mesmo.

Por fim, as funcionalidades do ambiente que engloba todas as características da simulação (vide figura 14), podem ser vistas na figura 17. Observa-se que os primeiros pontos constam os nodos *Fog* e o servidor *Cloud*. Avançando para a conexão com os BDs. Além disso captura-se o tempo em que ocorreu o começo dos experimentos, a fins de observação.



– Figura 17 - Funcionalidades do ambiente de simulação. Figura da autora.

Para que o funcionamento do ambiente inicie todo o processo da simulação, utiliza-se uma abordagem de *Processing Demand* (Demanda de Processamento). Por Demanda, neste caso, refere-se à quantidade de poder de processamento necessária para lidar com uma tarefa ou dado específico. Na simulação, a demanda de processamento é gerada aleatoriamente para cada iteração. Maiores demandas de processamento exigem mais

recursos computacionais para processar, potencialmente excedendo a capacidade dos nodos *Fog* e exigindo a utilização do servidor.

A etapa de inicialização pertence à abordagem de demanda de processamento, uma vez que para executar a simulação sejam necessários alguns passos pré-execução. São eles:

- **Inicialização**:
  - Inicializa os nodos *Fog*, o servidor, a conexão MySQL, o cliente e os dados CSV.
  - Registra o tempo de início e inicializa listas de utilização para nodos *Fog* e servidor em nuvem.
- **Método `calculate_utilization()`:**
  - Calcula a utilização dos nodos *Fog* e do servidor com base no tempo decorrido desde o início.
  - Registra a utilização calculada em listas de utilização correspondentes.
- **Método `allocate_resources(data)`:**
  - Aloca recursos para processar os dados fornecidos.
  - Tenta alocar recursos para os nodos de *Fog* e, em seguida, para o servidor, se necessário.
  - Retorna uma mensagem indicando onde os dados foram processados.
- **Método `allocate_and_calculate(data)`:**
  - Chama os métodos `allocate_resources(data)` e `calculate_utilization()` em sequência para processar os dados e calcular a utilização.
- **Função `polystore_search`:**
  - Esta função simula uma operação de pesquisa em vários armazenamentos de dados, ou seja, em um banco de dados MySQL, um banco de dados MongoDB e um arquivo CSV.
  - Recebe informações sobre a fonte de dados e a chave a ser pesquisada.
  - Retorna um dicionário contendo resultados de pesquisa de diferentes fontes de dados.
  - Na simulação, esta função é chamada para buscar dados após as demandas de processamento serem alocadas aos nodos *Fog* ou ao servidor *Cloud*.

4.2 RESULTADOS

Conforme o exposto na seção 4.1, a simulação modela a alocação de demandas de processamento para nodos *Fog* e um servidor *Cloud* com base em seus poderes de processamento e limites de utilização. A função *Polystore Search* simula operações de recuperação de dados em vários armazenamentos de dados. Os nodos representam recursos computacionais localizados mais próximos da borda, capazes de processar tarefas com menores requisitos computacionais.

Tendo em vista que cada parte do sistema desenvolvimento possui suas características, configurações e conceitos, uma importante função é executada para gerar os resultados do processo de simulação. Esta função se chama *allocate_resources* (função de alocação de recursos). A figura 18 retrata como se dá dentro desta chamada.

```
def allocate_resources(self, data):
    if 'processing_demand' in data and 'subject_id' in data:
        if self.cloud_server.process_data(data['processing_demand']):
            # If cloud server can handle the processing demand, process data there
            polystore_search_result = polystore_search(data['  '], data[  ])
            return f"Data processed at Cloud Server using polystore_search: {polystore_search_result}"


        for node in self.fog_nodes:
            if node.process_data(data['processing_demand']):

                polystore_search_result = polystore_search(data['  '], data[     ])
                return f"Data processed at Fog Node {node.id} using polystore_search: {polystore_search_result}"


        return "Insufficient processing power"
    else:
        return "Data is missing required keys"
```

– Figura 18 - Exemplo de configuração dentro da função de distribuição de recursos. Figura da autora.

No geral, o método *allocate_resources* facilita a alocação dinâmica de recursos para processamento de solicitações de dados, considerando tanto os nodos *fog* quanto o servidor *Cloud*. Ele garante a utilização eficiente do poder de processamento disponível e ajuda no balanceamento de carga na infraestrutura de computação. O que permite que o que esteja disponível seja utilizado, sempre optando pela melhor forma de não ser desperdiçado nenhum recurso.

Em resumo, a função *allocate_resources*, portanto, é responsável por distribuir demandas de processamento de dados entre os nodos e o servidor disponíveis no ambiente de simulação. Conforme pode ser visto na figura 18, a função recebe como entrada um dicionário contendo informações sobre a demanda de processamento, especificamente a quantidade de processamento necessária e uma chave identificadora dos dados.

Inicialmente, a função verifica se a demanda de processamento possui as chaves necessárias, ou seja, se contém informações sobre a quantidade de processamento e a chave de identificação dos dados. Se essas informações estiverem presentes, o processo de alocação de recursos é iniciado.

Após essa etapa inicial, a função verifica se o servidor tem capacidade para processar a demanda de processamento. Se o servidor possuir capacidade disponível, a função executa a operação de busca de dados (*polystore_search*) no servidor e retorna o resultado.

Caso o servidor esteja ocupado ou não tenha capacidade suficiente para processar a demanda, a função tenta distribuir a demanda entre os nodos disponíveis. Para cada

nodo, a função verifica se o o mesmo possui capacidade para processar a demanda. Caso seja possível o processamento e o nodo possuir capacidade disponível, a função executa a operação de busca de dados (*polystore_search*) e retorna o resultado.

Se nenhum dos nodos ou o servidor puderem processar a demanda devido a falta de capacidade, a função retorna uma mensagem indicando que há "Insuficiência de capacidade de processamento". Por fim, se as chaves necessárias não estiverem presentes nos dados de entrada, a função retorna uma mensagem indicando que os dados estão faltando.

Portanto, de forma sintetizada, para obtenção dos resultados, os seguintes passos foram executados:

- Início dos experimentos:
  - Inicializa o conjunto de nodos e o servidor.
  - Os nodos são criados com locais e poderes de processamento aleatórios.
  - O servidor é inicializado com um poder de processamento especificado.
- Configuração do ambiente:
  - Inicializa ambiente *Fog*, juntamente com o servidor e o *Polystore*.
- Processamento de dados:
  - Definição do método de alocamento de recursos para o processamento de dados.
  - Etapa de verificação se o procedimento foi corretamente iniciado.
  - Tentativa de alocação dentro dos nodos *Fog*.
  - Caso não seja possível alocar nos nodos disponíveis, recorre ao servidor.
  - Cálculo de ganho do processamento da *Fog* e da *Cloud*.
  - Retorno de indicação do alocamento dos dados e qual método gerou maior ganho na execução.
- Execução de simulação:
  - Execução da função *run simulation* que usa o ambiente.
  - Utiliza os dados obtidos da busca do *Polystore*.
  - Alocação de recursos e cálculo de utilização.
- Análise de resultados:
  - Retorna a utilização média dos nodos *Fog*.
  - O mesmo ocorre para o servidor (caso tenha sido necessário o servidor executar).

Em determinadas circunstâncias os dados inseridos no simulador podem variar. Sendo assim, tendo em vista todas as características passadas ao simulador e dado o início dos experimentos, espera-se um retorno semelhante ao exposto na figura 19.

Após esse início de simulação consegue-se obter os seguintes resultados expostos na figura 20. Nota-se que, nestes casos a utilização dos nodos *Fog* atingiu um valor de 0.26666, aproximadamente 0.2667, isto significa que atingiu a taxa de $26,67\%$ de seu poder

```
No processing time recorded yet.
Average Fog Node Utilization: 0
Average Cloud Server Utilization: 0
Fog Node Process Data:
Current Load: 800
Processing Demand: 800
Processing Power: 2000
```

– Figura 19 - O começo da execução retorna os recursos iniciais disponíveis.

```
Fog Node Processed Data: Node 2
Start time initialized: 1709234893.5729446
Fog Node Utilization: []
Cloud Server Utilization: []
Fog Node Utilization: 0.2666666666666666
Cloud Server Utilization: 0.0
```

– Figura 20 - Resultados obtidos para um limite de utilização de 80% dos nodos disponíveis.

de processamento total. Foi necessário apenas um nodo para executar o experimento e o mesmo foi suficiente, exigindo menos da *Cloud*. Portanto, há capacidade de processamento adicional para realizar mais tarefas dentro do grupo de nodos disponíveis.

A figura 21 exemplifica os resultados que podem ser obtidos ao serem executados utilizando cinco nodos *fog* e um servidor *Cloud* com um poder de processamento menor do que a demanda apresentada. Sendo assim, a depender do poder de processamento existente, o *Polystore* precisará ser executado no servidor *Cloud*, o que demandaria mais tempo, por exemplo.

Vale ressaltar que a demanda de processamento irá depender de diversos fatores, tais como:

- Urgência de acesso aos dados IoT que chegam toda hora ao sistema;
- Disponibilidade de processamento do servidor;
- Quantidade de nodos *fog* que estão disponíveis;
- Acessos simultâneos de usuários a aplicação;
- Frequência de utilização e requerimento dos mesmos dados por diversos usuários;
- Quais BDs retornam os resultados em quais camadas disponíveis.

Sendo assim, num caso de maior disponibilidade, a aplicação retornará um cenário como o descrito na figura 22, onde, na primeira iteração de utilização, a camada *fog* consegue atender às demandas de uso dos dados. Porém, as demais já são executadas na *Cloud*.

```
Execution time: 15.0992 seconds

Iteration 1:
Data 1: Data processed at Cloud Server
Data 2: Insufficient processing power
Data 3: Insufficient processing power

Iteration 2:
Data 1: Insufficient processing power
Data 2: Insufficient processing power
Data 3: Insufficient processing power

Iteration 3:
Data 1: Insufficient processing power
Data 2: Insufficient processing power
Data 3: Insufficient processing power
```

– Figura 21 - Resultados de exemplo ao serem executadas três iterações do simulador. Figura da autora.

O processamento dos nodos *fog* busca verificar quais deles estão disponíveis na variável *available_fog_nodes* (que opera como uma lista). Ou seja, caso esteja vazia a lista, não é possível que nenhum dado seja processado na *Fog*. Esse método seleciona o primeiro da lista e retorna uma mensagem indicando que os dados serão processados nesse nodo. A mensagem inclui o ID do nó fog (*available_fog_nodes*[0].*id*).

De modo semelhante opera o processamento no *CloudServer*, se não existir nodos disponíveis (*available_fog_nodes* está vazio), o método tenta processar os dados no servidor de *Cloud* (*self.cloud_server*). Ele chama o método *process_data* da instância do servidor. Após isso é passada a demanda de processamento dos dados. Se o servidor tiver poder de processamento suficiente, ele retornará uma mensagem indicando que os dados serão processados no servidor em *Cloud*. Num cenário onde a *Fog* se faz presente é mais interessante averiguar a disponibilidade e alocação de recursos para que existam mais chances de dados serem processados na *Fog*.

Portanto, se nem os nodos nem o servidor puderem lidar com a solicitação de dados devido ao poder de processamento insuficiente, o método retornará uma mensagem indicando o mesmo, conforme previamente demonstrado na figura 21.

Para melhor entendimento de tudo que fora explicado anteriormente, foram elaboradas classes dentro da aplicação as quais possuem elementos dentro de sua estrutura:

- As classes *FogNode* e *CloudServer* que representam os nodos *fog* e o servidor *cloud*, respectivamente, com poder de processamento, assim como a carga disponível;
- A classe *Environment* trata da alocação de recursos com base no balanceamento de carga.

Outra característica da aplicação se dá pelo fato de que os BDs são executados em

```
Execution time: 15.0698 seconds

Iteration 1:
Data 1: Data processed at Fog Node 1
Data 2: Data processed at Fog Node 5
Data 3: Data processed at Fog Node 1

Iteration 2:
Data 1: Data processed at Cloud Server
Data 2: Data processed at Cloud Server
Data 3: Data processed at Cloud Server

Iteration 3:
Data 1: Data processed at Cloud Server
Data 2: Data processed at Cloud Server
Data 3: Data processed at Cloud Server
```

– Figura 22 - Resultados de exemplo sob mudança de cenário de poder de processamento e disponibilidade. Figura da autora.

paralelo, usando recurso de paralelismo, a fim de que seja um "mecanismo de bloqueio que não permite que apenas um *thread* execute o código por vez, mesmo em servidores com vários núcleos de CPU"[7].

Além dos resultados previamente apresentados, têm-se a adição de algumas métricas interessantes dentro dos conceitos de *Cloud* e *Fog* que são as taxas de utilização. Essas taxas permitem medir a eficiência e a eficácia do uso de recursos em ambientes *Cloud* e *Fog*, respectivamente. E, no caso da proposta deste trabalho, levou-se em consideração tais métricas devido ao fato de serem inerentes ao escopo e podendo trazer mais inferências relacionadas ao uso da proposta.

Um dos objetivos da utilização da *Fog* é o de descarregar tarefas de computação e armazenamento de dados presentes na *Cloud* que estão distantes em relação aos nodos *fog*. Ademais, esse uso, conforme falado anteriormente em **2.1.2**, permite reduzir a latência e melhorar a eficiência geral do sistema. Abaixo consta uma simples fórmula que exemplica a taxa de utilização da *Fog*.

$$U_{fog} = \frac{A}{B} \tag{4.1}$$

Onde U_{fog} simboliza a taxa de utilização da *Fog*, A a total de processamento em todos os nodos e B a carga total de processamento em todos os nodos. Esta fórmula permite visualizar mais sobre os nodos, se estão sendo utilizados adequadamente ou se há capacidade de processamento não utilizada.

Analisando os valores que podem aparecer nesse resultado:

[7] Revelo Community

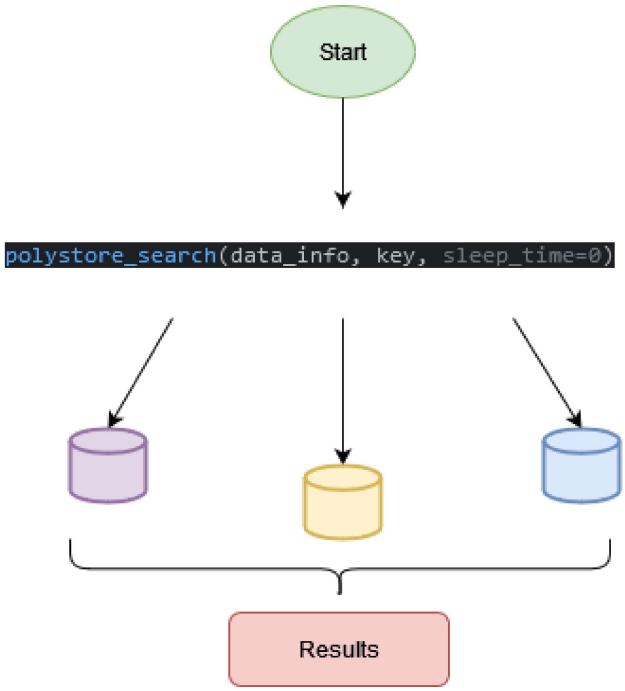
- Se $U_{fog} = 1$, indica que todos os nodos estão sendo utilizados, ou seja, eles estão processando tarefas em sua capacidade máxima.
- Se $U_{fog} < 1$ indica que há capacidade de processamento não utilizada nos nodos.

Já a métrica de utilização da *Cloud* segue a linha demonstrada na equação abaixo:

$$N_u = \frac{M}{N} \tag{4.2}$$

Onde N_u é a utilização da rede, M representa os dados transferidos e N a taxa máxima de dados transferidos. E, dentro da simulação, representa a carga atual dividida pelo poder de processamento.

A saída final de toda a execução do *Polystore* dentro da aplicação pode ser vista na figura 23.



– Figura 23 - ormato do resultado de uma busca dentro do *Polystore*. Figura da autora.

Observa-se que, ao dar início a uma execução, busca-se pela classe *polystore_search* que tem por parâmetros a informação de localização da chave (*data_info*), a chave que se deseja procurar (*key*) e a variável *sleep_time* que é usada como parâmetro de paralelismo na execução, conforme explicado anteriormente na subseção **3.3.1**.

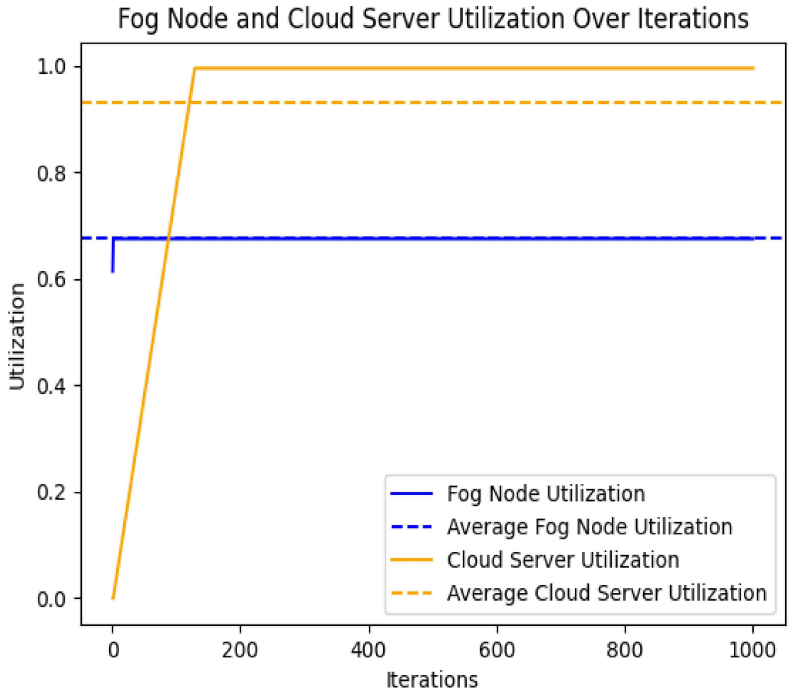
Após essa etapa, são atingidas as bases de dados e são buscados os resultados referêntes à pesquisa inicial, ocasionando na mostra deles na tela do usuário. Vale lembrar que essa implementação do *Polystore* foi realizada de forma manual dentro da aplicação, sem utilização de arquitetura externa para tal. A ideia é sempre buscar pelos termos comuns e unificá-los numa só busca. Para fins de visualização do experimento, foi introduzido um tempo de execução (em segundos) que é mostrado ao final da execução. Esse tempo leva em consideração o tempo de busca dentro das bases de dados e o *time.sleep* que é variável.

Na seção 3.2 foi explicada melhor a escolha e o conteúdo da base de dados. Relembrando que neste trabalho o foco não se trata apenas do volume de dados, apesar de que

aplicações IoT possuem essa característica. Dentro da aplicação a base de dados é dividida entre três vertentes que foram escolhidas para os experimentos: pacientes, prescrições para pacientes e eventos dentro do hospital. A saber:

- Pacientes: dados sensíveis das pessoas atendidas. Engloba o número *id* de cada paciente, gênero, datas importantes (nascimento, entrada e saída do hospital, por exemplo), entre outros.
- Prescrições: dados relacionados aos pacientes com relação à medicamentos aplicados, receitados, doses, etc.
- Eventos: gráficos de relação dos pacientes, resultados, hora do armazenamento das informações no sistema, etc.

Quando a aplicação é acionada e os resultados são exibidos, se abre um gráfico, em janela a parte, referente aos valores colocados dentro do simulador. Abaixo, na figura 24, segue um gráfico de exemplo de um resultado com as seguintes características: 10 nodos *fog* e 1000 iterações. Os valores de exemplo podem variar os gráficos de resultado, porém, aqui o foco não é relacionar comportamentos diferentes diante de variações, mas sim o comportamento geral da aplicação tendo em vista todas as características dela.



– Figura 24 - Exemplo de resultado de utilização de nodos *fog* e do *Cloud Server*. Figura da autora.

Verifica-se uma estabilidade no gráfico representado na figura 24 no que tange a utilização das camadas *Fog* e *Cloud*. A partir de certo momento da execução, o poder de processamento da *Fog* torna-se insuficiente para os nodos *fog* serem utilizados e, portanto, as execuções são realizadas na *Cloud*. Isso irá variar de acordo com a disponibilidade

previamente escolhida no simulador. Além disso, o poder de processamento da *Cloud* é variável também, tendo em vista a diversidade de sua utilização nos mais diversos tipos de cenários.

A variabilidade de resultados irá mudar conforme a disponibilidade de recursos de cada tipo de cenário de uso da aplicação. Vamos supor que, num determinado posto de saúde que é menor que um hospital disponha de recursos menores de poder de processamento dos nodos e do servidor, além da carga, quantidade de nodos e variação de alocação. Em contrapartida, um hospital maior que tem muitas admissões, modificações nos dados, etc. irá requerer um maior sistema para lidar com as demandas.

Sendo assim, foi proposto um simples cálculo que busca verificar o ganho que se tem ao utilizar a *Fog* e ao utilizar a *Cloud*, que é demonstrado a seguir:

$$G_{TP} = M_{TPC} - M_{TPF} \tag{4.3}$$

Onde G_{TP} é o ganho de processamento de tempo, M_{TPC} a média de processamento da *Cloud* e M_{TPF} a média de processamento da *Fog*.

A depender dos valores dispostos, visto nos parágrafos anteriores, esse resultado irá mudar e pode ser visualizado na figura 25.

```
Average Processing Time for Cloud Server: 0.05112338066101074
Gain in Average Utilization (Fog vs Cloud): 0.0
Gain in Average Processing Time (Cloud vs Fog): 0.05112338066101074
```

– Figura 25 - Exemplo de demonstração de cenário onde a *Cloud* possui um maior poder de processamento que a *Fog* e, portanto, é escolhida para execução das buscas. Figura da autora.

4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

A proposta deste trabalho foi utilizar de mecanismos de simulação, paralelismo e *Polystore*. Em resumo, os resultados obtidos demonstram que os dados buscados podem ser processados dentro da aplicação com simulação de uma rede que possui as camadas *Cloud* e *Fog*. Aqui, a camada *Edge* é representadas pelos dados e não foi levada tanto em consideração, uma vez que não possui poder de processamento igual às demais.

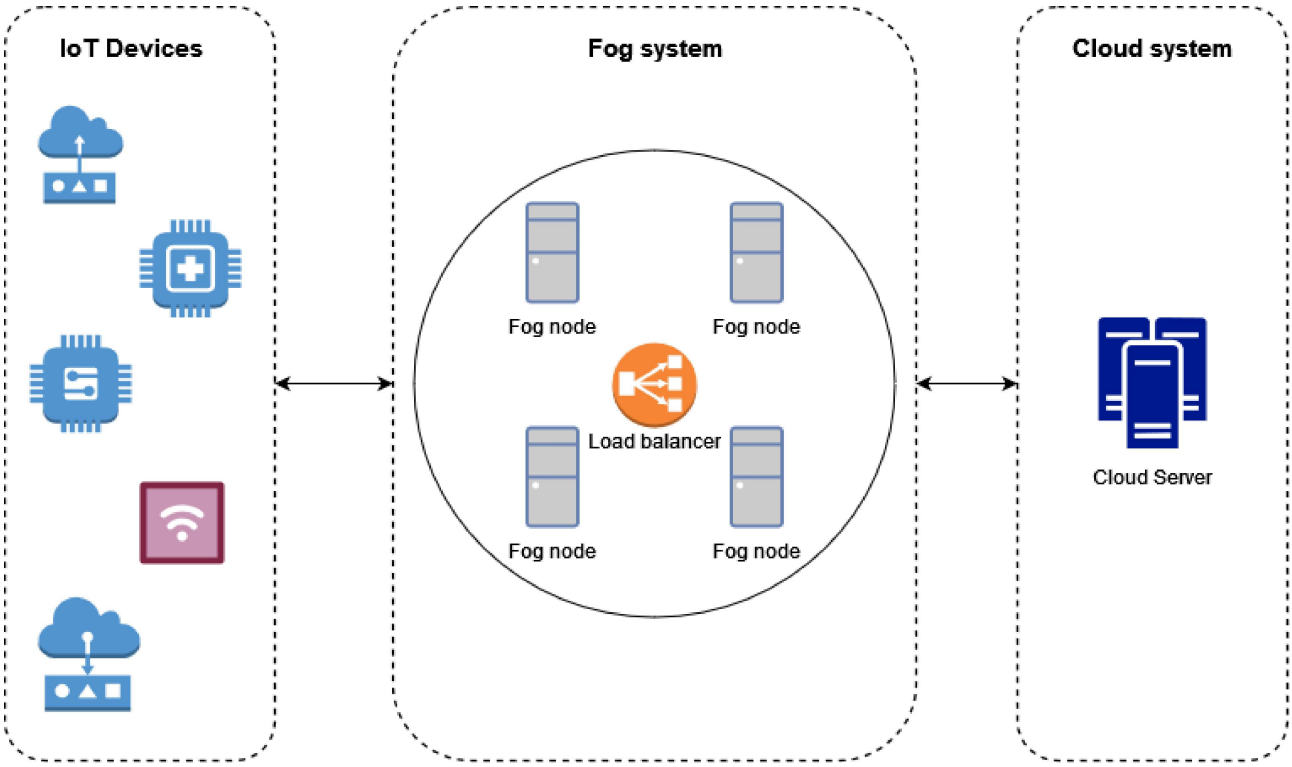
A função *initialize_fog_nodes* inicializa nodos *fog* com localizações aleatórias e poder de processamento. Sendo assim, o conjunto dessas classes forma um *framework* básico que facilita o processo de simulação de *fog* e computação *cloud* com alocação dinâmica de recursos em um ambiente *Polystore* usando *MySQL*, *MongoDB* e uma base heterogênea de dados.

O que se espera é que, ao final dos resultados que foram obtidos, têm-se um sistema que permite ao usuário realizar buscas em bases heterogêneas e verificar o quanto foi utilizado de recursos computacionais. Conforme falado ao longo da subseção 4, nesta proposta foi tratada a parte de latência da rede, obtendo-se informações de quais bases foram executadas na *Fog* e quais precisaram serem executadas na *Cloud*.

O parágrafo acima pode ser melhor verificado visualmente na figura 26. Onde, da esquerda para a direita, temos os *IoT Devices* que são os dispositivos que geram os dados

e que os mandam para a aplicação. Nota-se que esses dispositivos foram tratados como sensores, atuadores, etc.

A próxima comunicação dos dispositivos se dá com o *Fog system* onde é representado por um círculo contendo nodos *fog* os quais formam uma espécie de *cluster*. Ao centro do círculo encontra-se o *Load balancer* que é o balanceador de carga. Uma vez que, dentro da função *allocate_resources*, vista na figura 18, possui um processador de demanda e que é ajustado conforme disponibilidade dos nodos, o *Load balancer* é o responsável por efetuar a procura pelos recursos e os distribuir de forma balanceada, de modo que as consultas realizadas sejam exibidas. E, por fim, têm-se o *Cloud system* onde está alocado o servidor.



– Figura 26 - Demonstração visual da aplicação de execução da proposta. Figura da autora.

Após a análise dos resultados obtidos na execução da simulação, é importante considerar diversas observações e implicações adicionais que podem influenciar os cenários e os resultados dos experimentos. Estas considerações ajudam a entender melhor as variações possíveis e os fatores que podem afetar o desempenho e a utilização dos recursos no ambiente simulado. A seguir, são listados alguns pontos relevantes a serem levados em conta:

- O cenário dos experimentos pode mudar conforme a capacidade de processamento disponível.
- Existem cenários em que é possível observar que a execução exige que o servidor faça as operações, sendo assim, que a *Fog* não teve o poder de processamento ideal para ser utilizada.

- Diversidade de cenários faz com que os valores sejam mudados. Seja da utilização do conjunto de dados, sejam dos valores disponíveis para cada um dos recursos e camadas.
- Possibilidade de alterações conforme o cenário real dos experimentos. Por exemplo: um hospital com maiores recursos tende a ter uma melhoria nos equipamentos e na utilização de tecnologia em relação a uma unidade básica de saúde.

5 CONCLUSÕES E TRABALHOS FUTUROS

Sistemas IoT têm sido cada vez mais utilizados como objetivos de pesquisa, melhorando a qualidade de vida da população (saúde, bem-estar, ambientes inteligentes, etc.) e proporcionando a facilidade de captura de dados através de sensores.

A arquitetura *Polystore* traz como benefício uma unicidade no processo de buscas no escopo de bases heterogêneas de dados. Trazendo consigo, aliado à IoT, o desafio de trazer processamento, velocidade de execução, captura de dados e integração de múltiplos bancos de dados heterogêneos, entre outros, para esse ambiente IoT. Sistemas como este exigem muitas tarefas e respostas rápidas.

Pensando nisso, foi proposto um modelo que executaria a função do *Polystore* sendo processado dentro de uma aplicação com características similares às de um ambiente real de IoT. A simulação retorna algumas características de uma rede previamente configurada e os resultados obtidos das buscas de dados. Esse modelo visou demonstrar benefícios de separar o processamento do *Polystore* sendo feito exclusivamente na *Cloud* e buscou simular os possíveis ganhos de serem executados testes dentro da *Fog*.

A concepção da proposta do trabalho fora publicada no evento internacional AINA 2023 (64). Este evento, entitulado "Conferência Internacional sobre Redes e Aplicações de Informação Avançada", têm por objetivos abordar teoria, projeto e aplicação de redes de computadores e sistemas distribuídos de computação e informação. A submissão da proposta do trabalho foi uma versão inicial deste atual, na versão publicada foram abordadas as questões relacionadas ao processo de criação, início do desenvolvimento, motivação e obtenção de resultados iniciais.

5.1 CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

- Trabalhos relacionados relativos ao tema de pesquisa e ilustrado na tabela 6.
- Proposta do modelo de integração contemporâneo de um ambiente *Polystore*, considerando uma infraestrutura computacional *Fog-Cloud*.
- Simulação em um ambiente experimental, com visualização de execução dos testes.
- Publicação da proposta da dissertação no evento internacional AINA 2023 sob o título de *A Polystore Proposed Environment Supported by an Edge-Fog Infrastructure* (64), de *qualis* A3.

5.2 LIMITAÇÕES

Existem diversas métricas, além das utilizadas, para comparar o ganho de se priorizar o uso de aplicações dentro da *Fog*, tais como:

- Maior disponibilidade de acordo com a capacidade de processamento em caso de desconexão, por exemplo;
- Economia de banda de rede;

- Segurança e privacidade;
- Escalabilidade limitada: embora a *Fog* tenha vantagens, ela pode ter limitações se comparada à *Cloud*;
- Gerenciamento de dados distribuídos: alta complexidade de implementação que garentem consistência, integridade e segurança dos dados.

Essas métricas não foram levadas em consideração neste trabalho, tendo em vista a alta complexidade de implementação e necessidade de ampliar o ambiente de testes, não só em termos de processamento de máquina, mas em tecnologias, *frameworks*, APIs, etc.

A dificuldade de realizar conexões com arquiteturas preexistentes foi algo que estimulou a criação manual do *Polystore*. A configuração da famosa arquitetura *BigDAWG* não foi difícil de fazer, assim como rodar exemplos sugeridos dentro da documentação. A interface do *BigDAWG*, ainda que dentro de um terminal, é amigável e fácil de ser interpretada. Porém, ao deparar com o desafio proposto, a conexão externa do *BigDAWG* tornou-se difícil de realizar, tendo em vista a documentação oficial.

Houve um empecilho relacionado a testes utilizando *softwares open-source* de simulação de redes. Alguns desses *softwares* são bem visuais e trabalham muito a questão de envio de pacotes, adição de *routers*, *switches*, entre outros elementos que estão presentes em ambientes de redes, porém, houve a dificuldade de junção destes programas com o escopo deste trabalho, não só no quesito de incorporação da implementação, como também a falta de documentação mais explicada com exemplos mais concretos.

5.3 TRABALHOS FUTUROS

Na seção acima foram levantados alguns pontos que foram limitantes durante o desenvolvimento desta dissertação. Como pontos fortes a serem trabalhados futuramente vale descatar a possibilidade de utilização de uma arquitetura *Polystore* mais robusta, como o *BigDAWG*, por exemplo, e verificar se essa junção ao simulador criado pode gerar resultados mais expressivos e passíveis de comparação entre propostas.

Outro ponto a ser considerado seria o de tornar a proposta mais visual e amigável para o usuário, pois, por hora, a recomendação para usar a aplicação atual é que o usuário entenda o básico de linguagem de programação, lógica de programação e conceitos computacionais. Em um ambiente IoT, têm-se os mais diversos tipos de usuários, desde os que possuem estes conhecimentos até leigos. Isso dependeria de tópicos externos aos principais deste trabalho, tais como: *Quality Assurance*, *User Experience*, implementação *front-end*, entre outras possibilidades.

REFERÊNCIAS

1 A., G.; LECLERCQ, ; SAVONNET, M.; CULLOT, N. Empowering big data analytics with polystore and strongly typed functional queries. In: SYMPOSIUM ON INTERNATIONAL DATABASE ENGINEERING APPLICATIONS (IDEAS '20), 24., 2020. **Proceedings...** [S.l.: s.n.], 2020. p.13:1–13:10.

2 AKELBEIN, J. et al. BASE MoVE-A Basis for a Future-proof IoT Sensor. In: INFORMATIK 2020, 2021. **Anais...** [S.l.: s.n.], 2021.

3 AKELBEIN, J. P.; BECKMANN, K.; HOSS, M.; SCHNEIDER, S.; SEYFARTH, S.; THOSS, M. BASE MoVE - A Basis for a Future-proof IoT Sensor. In: LECTURE NOTES IN INFORMATICS (LNI), PROCEEDINGS - SERIES OF THE GESELLSCHAFT FUR INFORMATIK (GI), 2020. **Anais...** [S.l.: s.n.], 2020. v.P-307.

4 ALLI, A. A.; ALAM, M. M. The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications. **Internet of Things**, [S.l.], v.9, p.100177, 2020.

5 ALOTAIBI, R.; CAUTIS, B.; DEUTSCH, A.; LATRACHE, M.; MANOLESCU, I.; YANG, Y. ESTOCADA: towards scalable polystore systems. **Proc. VLDB Endow.**, [S.l.], v.13, n.12, p.2949–2952, aug 2020.

6 AMADEO, M.; MOLINARO, A.; PARATORE, S. Y.; ALTOMARE, A.; GIORDANO, A.; MASTROIANNI, C. A Cloud of Things framework for smart home services based on Information Centric Networking. In: IEEE 14TH INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.245–250.

7 ATTASENA, V.; SATIRAMATEKUL, T.; SUEBSAIPROM, P.; ENGKANINAN, A.; PHADERMROD, B. fVSS-Based Secure Database for IoT Cloud Platform. In: JOINT INTERNATIONAL CONFERENCE ON DIGITAL ARTS, MEDIA AND TECHNOLOGY WITH ECTI NORTHERN SECTION CONFERENCE ON ELECTRICAL, ELECTRONICS, COMPUTER AND TELECOMMUNICATIONS ENGINEERING (ECTI DAMT NCON), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.261–266.

8 AZEVEDO, L. G.; SOUZA, R. F. S.; SOARES, E. F. d. S.; THIAGO, R. M.; TESOLIN, J. C. C.; OLIVEIRA, A. C.; MORENO, M. F. A Polystore Architecture Using Knowledge Graphs to Support Queries on Heterogeneous Data Stores. **arXiv preprint arXiv:2308.03584**, [S.l.], 2023.

9 AZEVEDO, L. G.; SOUZA, R.; SOARES, E.; THIAGO, R. M.; TESOLIN, J. C. C.; OLIVEIRA, A. C. C. M.; MORENO, M. F. HKPoly: A Polystore Architecture to Support Data Linkage and Queries on Distributed and Heterogeneous Data. In: 2022 , 2024, New York, NY, USA. **Anais...** Association for Computing Machinery, 2024. (SBSI '24).

10 BASCIANI, F.; DI ROCCO, J.; DI RUSCIO, D.; PIERANTONIO, A.; IOVINO, L. TyphonML: a modeling environment to develop hybrid polystores. In: ACM/IEEE INTERNATIONAL CONFERENCE ON MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS: COMPANION PROCEEDINGS, 23., 2020, New York,

NY, USA. **Proceedings...** Association for Computing Machinery, 2020. (MODELS '20).

11 BOHLI, J. et al. SMARTIE project: Secure IoT data management for smart cities. In: INTERNATIONAL CONFERENCE ON RECENT ADVANCES IN INTERNET OF THINGS (RIOT), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.1–6.

12 BONDIOMBOUY, C. Query processing in cloud multistore systems. **BDA: Bases de Données Avancées**, [S.l.], 2015.

13 BONDIOMBOUY, C.; VALDURIEZ, P. Query processing in multistore systems: an overview. **International Journal of Cloud Computing**, [S.l.], v.5, n.4, p.309–346, 2016.

14 CATTELL, R. Scalable SQL and NoSQL data stores. **SIGMOD Rec.**, New York, NY, USA, v.39, n.4, p.12–27, may 2011.

15 DAVOUDIAN; MENGCHI. Big data systems: A software engineering perspective. **ACM Computing Surveys**, [S.l.], v.53, n.5, p.1–39, 2020.

16 DE DONNO, M.; TANGE, K.; DRAGONI, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. **IEEE Access**, [S.l.], v.7, 2019.

17 DU, J.; MEEHAN, J.; TATBUL, N.; ZDONIK, S. Towards Dynamic Data Placement for Polystore Ingestion. In: INTERNATIONAL WORKSHOP ON REAL-TIME BUSINESS INTELLIGENCE AND ANALYTICS (BIRTE '17), 2017. **Proceedings...** [S.l.: s.n.], 2017. p.2:1–2:8.

18 ERGÜZEN, A.; ÜNVER, M. Developing a file system structure to solve healthy big data storage and archiving problems using a distributed file system. **Applied Sciences**, [S.l.], v.8, n.6, p.913, 2018.

19 FADIYA, S. O.; SAYDAM, S.; ZIRA, V. V. Advancing big data for humanitarian needs. **Procedia Engineering**, [S.l.], v.78, p.88–95, 2014.

20 FATIMA, H.; WASNIK, K. Comparison of SQL, NoSQL and NewSQL databases for internet of things. In: IEEE BOMBAY SECTION SYMPOSIUM (IBSS), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.

21 FENG, X.; MA, J.; LIU, S.; MIAO, Y.; LIU, X.; CHOO, K.-K. R. Transparent Ciphertext Retrieval System Supporting Integration of Encrypted Heterogeneous Database in Cloud-Assisted IoT. **IEEE Internet of Things Journal**, [S.l.], v.9, n.5, p.3784–3798, 2022.

22 FERREIRA, Y. M. **Polyflow**: a Polystore-compliant mechanism to provide interoperability to heterogeneous provenance graphs. 2020. Dissertação (Mestrado em Ciência da Computação) — Mestrado em Ciência da Computação (Dissertações). `https://repositorio.ufjf.br/jspui/handle/ufjf/12302`. 2020.

23 FISCHER, J. E.; CRABTREE, A.; RODDEN, T.; COLLEY, J. A.; COSTANZA, E.; JEWELL, M. O.; RAMCHURN, S. D. "Just whack it on until it gets hot": Working with IoT data in the home. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS - PROCEEDINGS, 2016. **Anais...** [S.l.: s.n.], 2016.

24 GADEPALLY, V. et al. The BigDAWG polystore system and architecture. In: IEEE HIGH PERFORMANCE EXTREME COMPUTING CONFERENCE (HPEC), 2016., 2016. **Anais...** [S.l.: s.n.], 2016.

25 GOMES, E.; COSTA, F.; DE ROLT, C.; PLENTZ, P.; DANTAS, M. A Survey from Real-Time to Near Real-Time Applications in Fog Computing Environments. **Telecom**, [S.l.], v.2021, n.2, p.489–517.

26 GUO, T. et al. Hybrid concurrency control protocol for data sharing among heterogeneous blockchains. **Frontiers of Computer Science**, [S.l.], v.18, n.3, p.183104, 2024.

27 GUPTA, P. et al. Smartbench: A benchmark for data management in smart spaces. In: VLDB ENDOWMENT, 2020. **Proceedings...** [S.l.: s.n.], 2020. v.13, n.12, p.1807–1820.

28 KEPNER, J. et al. Associative array model of SQL, NoSQL, and NewSQL Databases. In: IEEE HIGH PERFORMANCE EXTREME COMPUTING CONFERENCE (HPEC), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–9.

29 KIM, A. **Survey and Evaluation of Database Management System Extensibility**. 2024. Tese (Doutorado em Ciência da Computação) — Carnegie Mellon University Pittsburgh, PA.

30 KRANAS, P. et al. Parallel query processing in a polystore. **Distributed and Parallel Databases**, [S.l.], p.1–39, 2021.

31 KVET, M.; PAPAN, J.; DURNEKOVÁ, M. H. Treating temporal function references in relational database management system. **IEEE Access**, [S.l.], 2024.

32 LOPEZ PENA, M. A.; MUNOZ FERNANDEZ, I. Sat-iot: An architectural model for a high-performance fog/edge/cloud iot platform. In: IEEE 5TH WORLD FORUM ON INTERNET OF THINGS, WF-IOT 2019 - CONFERENCE PROCEEDINGS, 2019. **Anais...** [S.l.: s.n.], 2019.

33 LU, J.; I., H.; CAUTIS, B. Multi-model Databases and Tightly Integrated Polystores: Current Practices, Comparisons, and Open Challenges. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT (CIKM '18), 27., 2018. **Proceedings...** [S.l.: s.n.], 2018. p.2301–2302.

34 LUCERO, S. IoT platforms: enabling the Internet of Things. **IHS Technology**, [S.l.], v.Whitepaper, 2016.

35 LUCERO, S. et al. **IoT platforms**: enabling the Internet of Things. White paper.

36 MANCINI, R. et al. iGateLink: A Gateway Library for Linking IoT, Edge, Fog, and Cloud Computing Environments. In: **Intelligent and Cloud Computing**. [S.l.]: Springer, Singapore, 2021. p.11–19.

37 MANCINI, R.; TULI, S.; CUCINOTTA, T.; BUYYA, R. iGateLink: A Gateway Library for Linking IoT, Edge, Fog, and Cloud Computing Environments. In: SMART INNOVATION, SYSTEMS AND TECHNOLOGIES, 2021. **Anais...** [S.l.: s.n.], 2021. v.194.

38 MANIHAR, S.; PATEL, R.; AGRAWAL, S. A survey on mission critical task placement and resource utilization methods in the IoT fog-cloud environment. In: RECENT TRENDS IN COMPUTATIONAL SCIENCES, 2024. **Anais...** [S.l.: s.n.], 2024. p.284–290.

39 MARTíN, C.; GARRIDO, D.; DíAZ, M.; RUBIO, B. From the Edge to the Cloud: Enabling Reliable IoT Applications. In: INTERNATIONAL CONFERENCE ON FUTURE INTERNET OF THINGS AND CLOUD (FICLOUD), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.17–22.

40 MEEHAN, Z. et al. Integrating real-time and batch processing in a polystore. In: IEEE HIGH PERFORMANCE EXTREME COMPUTING CONFERENCE, 2016. **Anais...** [S.l.: s.n.], 2016. p.1–7.

41 MIJUSKOVIC, A.; CHIUMENTO, A.; BEMTHUIS, R.; ALDEA, A.; HAVINGA, P. Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification. **Sensors**, [S.l.], v.21, n.5, p.1832, 2021.

42 MU, X.; ANTWI-AFARI, M. F. The applications of Internet of Things (IoT) in industrial management: a science mapping review. **International Journal of Production Research**, [S.l.], v.62, n.5, p.1928–1952, 2024. `https://doi.org/10.1080/00207543.2023.2290229`.

43 NAHA, R. K. et al. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. **Future Generation Computer Systems**, [S.l.], v.104, p.131–141, 2020.

44 NETO, C. G.; SALGADO, L.; STRÖELE, V.; DE OLIVEIRA, D. SigniFYIng APIs in the context of polystore systems: a case study with BigDAWG. In: BRAZILIAN SYMPOSIUM ON HUMAN FACTORS IN COMPUTING SYSTEMS (IHC '20), 19., 2020. **Proceedings...** [S.l.: s.n.], 2020. p.57:1–57:6.

45 PHILIP, J. M.; DURGA, S.; ESTHER, D. Deep learning application in iot health care: A survey. In: ADVANCES IN INTELLIGENT SYSTEMS AND COMPUTING, 2021. **Anais...** [S.l.: s.n.], 2021. v.1167.

46 PHILIP, J. M.; DURGA, S.; ESTHER, D. Deep learning application in iot health care: A survey. In: **Intelligence in Big Data Technologies—Beyond the Hype**. [S.l.]: Springer, Singapore, 2021. p.199–208.

47 PINCHEIRA, M. et al. Cost-effective IoT devices as trustworthy data sources for a blockchain-based water management system in precision agriculture. **Computers and Electronics in Agriculture**, [S.l.], v.180, p.105889, 2021.

48 PINCHEIRA, M.; VECCHIO, M.; GIAFFREDA, R.; KANHERE, S. S. Cost-effective IoT devices as trustworthy data sources for a blockchain-based water management system in precision agriculture. **Computers and Electronics in Agriculture**, [S.l.], v.180, 2021.

49 PISZCZEK, M. **Enhancing Cloud Database Performance**: General-Purpose Compression and Workload-Driven Layout. 2024. Tese (Doutorado em Ciência da Computação) — Massachusetts Institute of Technology.

50 POSTOACA, A.-V.; NEGRU, C.; POP, F. Deadline-aware Scheduling in Cloud-Fog-Edge Systems. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER, CLOUD AND INTERNET COMPUTING (CCGRID), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.691–698.

51 POUDEL, M. et al. Processing Analytical Queries over Polystore System for a Large Astronomy Data Repository. **Applied Sciences**, [S.l.], v.12, n.5, p.2663, 2022.

52 POUDEL, M.; SHRESTHA, S.; YILANG, W.; WANMING, C.; BHALLA, S. Polystore database systems for managing large scientific dataset archives. In: INTERNATIONAL CONFERENCE ON RELIABILITY, INFOCOM TECHNOLOGIES AND OPTIMIZATION (TRENDS AND FUTURE DIRECTIONS)(ICRITO), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–6.

53 PRASETIO, N.; FERDINAND, D.; WIJAYA, B. C.; ANGGREAINY, M. S.; KUMIAWAN, A. Performance Analysis of Distributed Database System in Cloud Computing Environment. In: IEEE 9TH INTERNATIONAL CONFERENCE ON COMPUTING, ENGINEERING AND DESIGN (ICCED), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–6.

54 RATH, K. C.; KHANG, A.; ROY, D. The Role of Internet of Things (IoT) Technology in Industry 4.0 Economy. In: **Advanced IoT Technologies and Applications in the Industry 4.0 Digital Economy**. [S.l.]: CRC Press, 2024. p.1–28.

55 SENGUPTA, S.; KAHVAZADEH, S.; MASIP-BRUIN, X.; GARCIA, J. SFDDM: A Secure Distributed Database Management in Combined Fog-to-Cloud Systems. In: IEEE 24TH INTERNATIONAL WORKSHOP ON COMPUTER AIDED MODELING AND DESIGN OF COMMUNICATION LINKS AND NETWORKS (CAMAD), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–7.

56 SHVACHKO, K.; KUANG, H.; RADIA, S.; CHANSLER, R. The Hadoop Distributed File System. In: IEEE 26TH SYMPOSIUM ON MASS STORAGE SYSTEMS AND TECHNOLOGIES (MSST), 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.1–10.

57 SINGH, S.; HOSEN, A. S.; YOON, B. Blockchain Security Attacks, Challenges, and Solutions for the Future Distributed IoT Network. **IEEE Access**, [S.l.], v.9, p.13938–13959, 2021.

58 SINGH, S.; SANWAR HOSEN, A. S.; YOON, B. Blockchain Security Attacks, Challenges, and Solutions for the Future Distributed IoT Network. **IEEE Access**, [S.l.], v.9, 2021.

59 SINGHAL, R.; ZHANG, N.; NARDI, L.; SHAHBAZ, M.; OLUKOTUN, K. Polystore++: accelerated polystore system for heterogeneous workloads. In: IEEE 39TH INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS (ICDCS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1641–1651.

60 TRIPATHY, H. K.; MISHRA, S.; DASH, K. Significance of IoT in Education Domain. In: . [S.l.: s.n.], 2021.

61 TRIPATHY, H. K.; MISHRA, S.; DASH, K. Significance of IoT in Education Domain. In: **Internet of Things**: Enabling Technologies, Security and Social Implications. [S.l.]: Springer, Singapore, 2021. p.59–83.

62 VLAHAVAS, G.; MICHAILIDOU, A.-V.; TOLIOPOULOS, T.; PSOMIADIS, V. G.; GOUNARIS, A.; VAKALI, A. Performance comparison of distributed in memory databases in dynamic fog environments. In: IEEE INTL CONF ON PARALLEL DISTRIBUTED PROCESSING WITH APPLICATIONS, BIG DATA CLOUD COMPUTING, SUSTAINABLE COMPUTING COMMUNICATIONS, SOCIAL COMPUTING NETWORKING (ISPA/BDCLOUD/SOCIALCOM/SUSTAINCOM), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.418–425.

63 YAMADA, H.; SUZUKI, T.; ITO, Y.; NEMOTO, J. ScalarDB: Universal Transaction Manager for Polystores. **Proc. VLDB Endow.**, [S.l.], v.16, n.12, p.3768–3780, aug 2023.

64 YUNG, L. R. B.; STRÖELE, V.; DANTAS, M. A. R. A Polystore Proposed Environment Supported by an Edge-Fog Infrastructure. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 2023. **Anais...** Springer International Publishing, 2023. p.292–302.

APÊNDICE A – Informações extras

APÊNDICE A – Projeto da proposta no GitHub: https://github.com/LudmilaY/dissertation

APÊNDICE A – Artigo publicado: YUNG, Ludmila Ribeiro Bôscaro; STRÖELE, Victor; DANTAS, Mario Antônio Ribeiro. "A Polystore Proposed Environment Supported by an Edge-Fog Infrastructure". In: International Conference on Advanced Information Networking and Applications. Cham: Springer International Publishing, 2023. p. 292-302. DOI: https://doi.org/10.1007/978-3-031-28451-9$_2$6. Qualis A3.