UNIVERSIDADE FEDERAL DO CEARÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

JOANA RANIKELLY DE ARAÚJO SILVA

EVOLUÇÃO E TENDÊNCIAS EM PLATAFORMAS DE PROCESSAMENTO DE GRAFOS EM LARGA ESCALA: UMA REVISÃO SISTEMÁTICA

RUSSAS

2025

JOANA RANIKELLY DE ARAÚJO SILVA

EVOLUÇÃO E TENDÊNCIAS EM PLATAFORMAS DE PROCESSAMENTO DE GRAFOS EM LARGA ESCALA: UMA REVISÃO SISTEMÁTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Cenez Araújo de Rezende

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58e Silva, Joana Ranikelly de Araujo.
Evolução e tendências em plataformas de processamento de grafos em larga escala: uma revisão sistemática / Joana Ranikelly de Araujo Silva. – 2025.
57 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2025.
Orientação: Prof. Dr. Cenez Araújo de Rezende.

1. Plataformas de processamento de grafos em larga escala. 2. Grafos. I. Título.

CDD 005.1

JOANA RANIKELLY DE ARAÚJO SILVA

EVOLUÇÃO E TENDÊNCIAS EM PLATAFORMAS DE PROCESSAMENTO DE GRAFOS EM LARGA ESCALA: UMA REVISÃO SISTEMÁTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: 28/08/2025

BANCA EXAMINADORA

Prof. Dr. Cenez Araújo de Rezende (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Tatiane Fernandes Figueiredo
Universidade Federal do Ceará (UFC)

Profa. Ms. Valéria Maria da Silva Pinheiro
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que tornaram este trabalho possível. Primeiramente, agradeço ao meu orientador, Prof. Dr. Cenez Araújo de Rezende, pela orientação e pela paciência ao longo de todo o processo. À minha família, sou profundamente grata pelo amor e pela confiança constantes. À minha mãe, Maria Rilene de Araújo Silva, cuja sua incansável dedicação e seu incentivo foram essenciais para que eu continuasse minha graduação. E, de maneira especial, ao meu namorado, Jorge Gabriel de Oliveira Medeiros, que sempre esteve ao meu lado com apoio e compreensão. Sua paciência e incentivo foram fundamentais para que eu superasse os desafios e seguisse em frente com confiança.

"Porque o Senhor é o que dá a sabedoria; da sua boca é que sai o conhecimento e o entendimento."

(Provérbios 2:6)

RESUMO

Este trabalho realizou uma revisão sistemática da literatura sobre plataformas de processamento de grafos em larga escala, com foco na evolução de soluções ao longo de 2013–2023 e nas tendências emergentes para eficiência, escalabilidade, segurança e interoperabilidade. Selecionamos 13 estudos primários de alto impacto e organizamos suas contribuições em quatro ondas temáticas: (i) consolidação do modelo vertex-centric e extensões híbridas CPU+GPU; (ii) otimizações de I/O e novas arquiteturas de particionamento; (iii) observabilidade fina e elasticidade custo-sensível; e (iv) sustentabilidade energética, propriedade temporal e orquestração serverless. Nossos achados mostram ganhos de até 120× em tempo de execução, aceleração de 1.9× em TEPS/W, speed-ups de 14× em SSD prefetch e reduções de custo de 34 %, mas também identificam gargalos de comunicação, ausência de mecanismos de segurança e falta de benchmarks unificados. Propusemos um roadmap para pesquisas futuras que inclui threat models para grafos distribuídos, particionamento adaptativo em streaming, APIs cross-framework, protótipos FaaS federado e digital twins de grafos em tempo real. Por meio desta síntese, oferecemos subsídios teóricos e práticos para orientar o desenvolvimento de novas plataformas que equilibrem desempenho, resiliência e confiança em ambientes heterogêneos de nuvem e edge.

Palavras-chave: processamento de grafos; plataformas de grafos; revisão sistemática; escalabilidade; eficiência energética; interoperabilidade; segurança em grafos

ABSTRACT

We present a systematic literature review of large-scale graph processing platforms spanning 2013–2023, examining how solutions have evolved to address efficiency, scalability, security, and interoperability. Thirteen high-impact primary studies were selected and classified into four thematic "waves": (1) consolidation of vertex-centric models with CPU+GPU extensions; (2) I/O optimizations and novel partitioning architectures; (3) fine-grained observability and cost-aware elasticity; and (4) energy sustainability, temporal property graphs, and serverless orchestration. Our synthesis reveals performance gains up to 120× in execution time, 1.9× improvements in TEPS/W, 14× speed-ups via SSD prefetch, and 34 % cost reductions, alongside persistent communication bottlenecks, a near-complete absence of integrated security mechanisms, and a lack of unified benchmarking frameworks. We propose a future research roadmap including distributed graph threat modeling, adaptive streaming partitioning, cross-platform APIs, federated FaaS prototypes, and real-time graph digital twins. This work provides both theoretical insights and practical guidelines for designing next-generation graph processing platforms that balance high performance, resilience, and trustworthiness in heterogeneous cloud-edge environments.

Keywords: graph processing; graph platforms; systematic literature review; scalability; energy efficiency; interoperability; graph security

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	Association for Computing Machinery
API	Application Programming Interface
BFS	Breadth-First Search
BSP	Bulk Synchronous Parallel
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DVFS	Dynamic Voltage and Frequency Scaling
E/S	Entrada e Saída
FaaS	Function-as-a-Service
GAS	Gather-Apply-Scatter
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HDFS	Hadoop Distributed File System
HPC	High-Performance Computing
I/O	Input/Output
IoT	Internet of Things
ML	Machine Learning
ONNX	Open Neural Network Exchange
PICOC	Population, Intervention, Comparison, Outcomes, Context
PIM	Processing-In-Memory
PUE	Power Usage Effectiveness
RAM	Random Access Memory
RDD	Resilient Distributed Dataset
SCC	Strongly Connected Components
SLA	Service-Level Agreement
SmartNIC	Smart Network Interface Card
SQL	Structured Query Language
SSD	Solid State Drive
SSSP	Single-Source Shortest Path
TEE	Trusted Execution Environment
TEPS	Traversed Edges Per Second

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Motivação

Nos últimos anos, a quantidade e a complexidade dos dados gerados em aplicações distribuídas, como redes sociais, Internet das Coisas e sistemas de recomendação, cresceram de forma exponencial. Em muitas dessas aplicações, as relações entre entidades são tão importantes quanto os próprios dados, por exemplo, conexões de amizade em redes sociais, dependências entre serviços em micro-serviços ou ligações entre moléculas em biologia computacional. Grafos são estruturas matemáticas naturalmente adequadas para modelar essas conexões: vértices representam entidades e arestas representam relacionamentos.

No entanto, quando a escala desses grafos chega a bilhões ou até trilhões de arestas, implementações tradicionais tornam-se impraticáveis: custo de Input/Output (I/O), balanceamento de carga, comunicação excessiva e limitações de memória são alguns dos desafios centrais. Para superar essas barreiras, surgiram novas plataformas e modelos de programação — como Pregel, PowerGraph, GraphX e soluções serverless — que repartem o trabalho entre múltiplos nós, aproveitam processamento heterogêneo (Central Processing Unit (CPU) + Graphics Processing Unit (GPU)) e exploram técnicas de pré-busca e caching.

1.2 Problema de Pesquisa

Apesar da quantidade de soluções, ainda faltam estudos que:

- Sintetizem a evolução histórica dessas plataformas, apontando avanços e limitações centrais;
- Identifiquem como eficiência, escalabilidade, segurança e interoperabilidade têm sido tratados ao longo do tempo;
- Destaquem tendências futuras capazes de orientar novas pesquisas e adoções em ambientes de produção.

Sem essa visão integrada, pesquisadores e engenheiros de software correm o risco de reinventar soluções já conhecidas ou de subestimar desafios emergentes em grandes ambientes de nuvem, High-Performance Computing (HPC) e edge computing.

1.3 Objetivos

1.3.1 Objetivo Geral

Realizar uma revisão sistemática da literatura para traçar a evolução das plataformas de processamento de grafos em larga escala, avaliando como as dimensões de eficiência, escalabilidade, segurança e interoperabilidade vêm sendo abordadas e apontando tendências futuras na área.

1.3.2 Objetivos Específicos

1. Mapear e categorizar os principais modelos computacionais (vertex-centric, edge-centric, GAS, dataflow) e suas implementações em plataformas maduras e emergentes;
2. Analisar empiricamente como cada plataforma otimiza recursos (CPU, I/O, energia) e escala sobre diferentes topologias de cluster;
3. Avaliar o nível de atenção dado a aspectos de segurança e interoperabilidade em cada estudo primário;
4. Identificar lacunas e propor direções de pesquisa para aumentar a resiliência, a compatibilidade entre ferramentas e a eficiência energética de soluções futuras.

1.4 Metodologia

Para cumprir esses objetivos, adotamos a abordagem de revisão sistemática da literatura conforme Kitchenham e Charters (2007):

- Definição de perguntas de pesquisa (Q1, Q2) e critérios Population, Intervention, Comparison, Outcomes, Context (PICOC);
- Formulação de strings de busca em bases como Scopus e Association for Computing Machinery (ACM) Digital Library (2013–2023);
- Aplicação de critérios de inclusão/exclusão, triagem de títulos, resumos e leitura completa;
- Extração padronizada de dados de 13 estudos primários (tabela de atributos);
- Síntese quantitativa (distribuição temporal, tipologia) e qualitativa (análise em “ondas”, mapeamento de desafios e tendências).

1.5 Estrutura do Trabalho

O restante deste trabalho está organizado da seguinte forma:

Capítulo 2 — Fundamentação Teórica: conceitos de sistemas distribuídos, teoria dos grafos e modelos de programação paralela.

Capítulo 3 — Metodologia: detalhamento do protocolo de revisão sistemática, bases consultadas e critérios de seleção.

Capítulo 4 — Resultados: apresentação quantitativa dos estudos, análise da evolução em ondas temáticas e mapeamento de desafios e tendências.

Capítulo 5 — Conclusões e Trabalhos Futuros: síntese dos principais achados, limitações do estudo e recomendações para pesquisas subsequentes.

Com isso, este trabalho se propõe a oferecer um panorama atualizado das plataformas de processamento de grafos em larga escala, auxiliando tanto acadêmicos quanto profissionais na escolha de soluções e no direcionamento de futuras investigações.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, apresentamos uma visão geral dos conceitos fundamentais que sustentam esta pesquisa, destacando sua relação com o processamento de grafos em larga escala.

2.1 Sistemas Distribuídos no Contexto de Processamento de Grafos

Na literatura, existem diferentes definições para sistemas distribuídos, cada uma delas enfatiza um aspecto importante do que é, de fato, um sistema distribuído.

Segundo Tanenbaum e Steen (2007), “um sistema distribuído é um conjunto de computadores independentes que se apresenta aos seus usuários como um sistema único e coerente”. Esta definição destaca a transparência para o usuário, indicando que a coleção de componentes interligados em rede deve parecer como um único sistema coeso. Em uma perspectiva complementar, Coulouris *et al.* (2007) definem que "um sistema distribuído é aquele no qual componentes localizados em computadores interligados comunicam-se e coordenam suas ações apenas por meio de troca de mensagens". Esta definição enfatiza a comunicação entre os componentes (nós) do sistema por meio de mensagens, sendo esta a base para a coordenação e funcionamento do sistema como um todo. De forma similar, Özsu e Valduriez (2010) afirmam que "um sistema de computação distribuída afirma que ele é composto por vários elementos de processamento autônomos (não necessariamente homogêneos) que estão interconectados por uma rede de computadores e que cooperam na execução de suas tarefas designadas". Essa definição destaca a autonomia entre os nós e a cooperação entre eles para realizar tarefas, o que é essencial em sistemas distribuídos, particularmente em contextos como o de bancos de dados distribuídos.

De maneira geral, sistemas distribuídos são caracterizados por componentes que funcionam de forma colaborativa para realizar tarefas complexas, geralmente em locais geograficamente distintos, para melhorar a eficiência, a escalabilidade e a tolerância a falhas. Esses sistemas são fundamentais para aplicações modernas como computação em nuvem, *big data*, e especificamente, para o processamento de grafos em larga escala que é o foco deste trabalho. Em tais sistemas, a capacidade de distribuir a carga de trabalho entre vários nós permite lidar com volumes massivos de dados e operações simultâneas de maneira eficiente.

Para exemplificar esse fato podemos citar o Facebook, onde grafos com mais de 1 trilhão de arestas foram processados por um *cluster* de 200 máquinas, usando o Apache Giraph,

executado sobre a plataforma Hadoop. Como escreveu Ching *et al.* (2015), a implementação de um sistema distribuído foi essencial para viabilizar a análise de grafos sociais, como recomendação de conteúdo, propagação de informações e cálculos de centralidade. O estudo também destacou a necessidade de integração com outras ferramentas de dados distribuídos, como Hive, Hadoop Distributed File System (HDFS) e MapReduce para fornecer dados de fontes internas da empresa aos algoritmos.

2.1.1 Características de Sistemas distribuídos

2.1.1.1 Transparência

Uma das principais características dos sistemas distribuídos é a transparência, um conceito que pode ser estendido para diversos aspectos desse tipo de sistema. A transparência pode ser definida como a ocultação dos componentes que formam um sistema distribuído, o que permite aos usuários interagirem com o sistema como se ele fosse único e coeso, independente da sua distribuição física. Tanenbaum e Steen (2007) introduzem sete tipos de transparência:

Tabela 1 – Tipos de Transparência em Sistemas Distribuídos

Transparência	**Descrição**
Acesso	Trata de ocultar diferenças na representação dos dados e nos métodos de acesso aos recursos.
Localização	Permite que recursos sejam acessados sem conhecimento da sua localização física ou na rede.
Migração	Oculta que um recurso pode ser movido para outra localização.
Relocação	Oculta que um recurso pode ser movido para outra localização enquanto em uso.
Replicação	Oculta que um recurso é replicado.
Concorrência	Oculta que um recurso esteja sendo usado por mais de um usuário ao mesmo tempo.
Falha	Oculta a falha e a recuperação de um recurso.

Fonte: Adaptado de Tanenbaum (2007).

A transparência é fundamental para a experiência do usuário em sistemas distribuídos, pois ela permite que a complexidade subjacente da distribuição dos recursos seja invisível para quem utiliza o sistema. Isso assegura uma interação fluida e eficiente, mesmo em ambientes altamente distribuídos.

2.1.1.2 Comunicação

Coulouris *et al.* (2007) atribuem à necessidade de compartilhar recursos o motivo de existirem sistemas distribuídos. A comunicação entre os nós que compõem esses sistemas é fundamental para viabilizar esse compartilhamento e envolve a troca de informações entre processos que residem em computadores distintos mas interligados por uma rede. A comunicação em sistemas distribuídos pode ser realizada de várias formas, com os mecanismos mais comuns sendo a troca de mensagens e o uso de protocolos de comunicação. A troca de mensagens é o método mais direto, onde processos enviam e recebem dados através de canais de comunicação (ANDREWS, 2000). Este método é a base para muitos sistemas distribuídos, permitindo que diferentes componentes cooperem de maneira eficiente. Um exemplo prático da comunicação em sistemas distribuídos é o uso de mensageria assíncrona em sistemas de microsserviços. Aqui, diferentes serviços podem comunicar-se entre si utilizando filas de mensagens, como o RabbitMQ ou Apache Kafka (KREPS *et al.*, 2011), onde as mensagens são enviadas de forma assíncrona, permitindo uma maior escalabilidade e resiliência do sistema.

2.1.1.3 Coordenação e sincronização

Coordenação e sincronização são elementos cruciais em sistemas distribuídos, pois garantem que os diferentes componentes do sistema possam operar em harmonia, mesmo quando estão geograficamente dispersos. A coordenação envolve a organização das atividades entre nós, enquanto a sincronização se refere à manutenção de uma ordem temporal ou de estado entre eles (LAMPORT, 1978).

Um dos maiores desafios em sistemas distribuídos é a sincronização de relógios. Devido à natureza distribuída, cada nó pode ter seu próprio relógio, levando a potenciais discrepâncias temporais entre as operações. Lamport (1978) propôs o conceito de relógios lógicos, que, em vez de sincronizar fisicamente todos os relógios, permite que os eventos sejam ordenados logicamente com base nas dependências causais. Esse método de sincronização lógica é fundamental para sistemas como bancos de dados distribuídos, onde a ordem das transações precisa ser mantida para garantir a consistência.

Portanto, a coordenação e a sincronização em sistemas distribuídos são fundamentais para garantir que os componentes distribuídos possam trabalhar de forma coesa e eficiente, mantendo a integridade e a consistência dos dados e das operações.

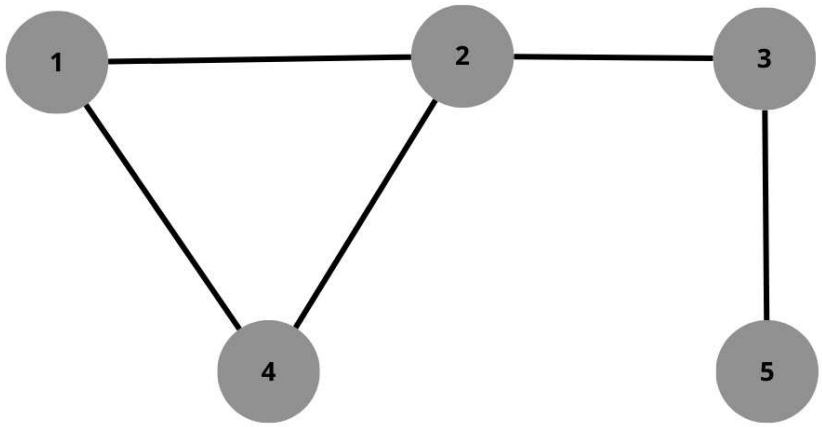
2.2 Fundamentos da Teoria dos Grafos

A Teoria dos Grafos engloba o estudo dos elementos que formam um grafo (vértices e arestas). Essa estrutura que aparenta ser simples permite a modelagem e solução de diversos problemas como, por exemplo, rotas em mapas, análise de redes sociais e otimização de fluxos de entregas. Nos subtópicos abaixo são explorados alguns conceitos e propriedades do grafos.

2.2.1 Definições Básicas

Como definido por Diestel (2017), um grafo é um par $G = (V, E)$, onde V é o conjunto de *vértices* e $E \subseteq [V]^2$ é o conjunto de *arestas*, ou seja, subconjuntos de dois elementos de V.

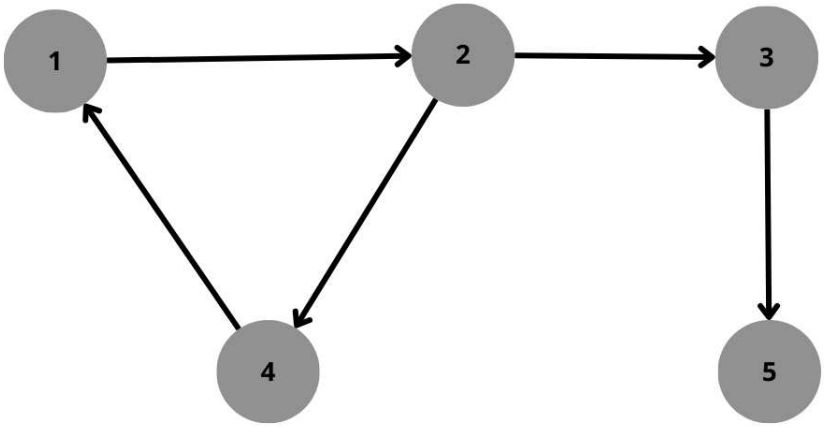
Figura 1 – Grafo simples não direcionado.



Fonte: elaborado pelo autor (2025).

A Figura 1 ilustra um grafo simples, ou seja, não possui laços, onde o conjunto de *vértices* $V = \{1,2,3,4,5\}$ e o conjunto de *arestas* $E = \{\{1,2\},\{1,4\},\{2,3\},\{2,4\},\{3,5\}\}$ (GROSS; YELLEN, 2004). A natureza das arestas define a classificação dos grafos, que podem ser *direcionados* quando as arestas possuem orientação, assim como na Figura 2, ou *não direcionados* quando as arestas não possuem orientação, como na Figura 1 (CHARTRAND; ZHANG, 2012).

Figura 2 – Grafo simples direcionado.

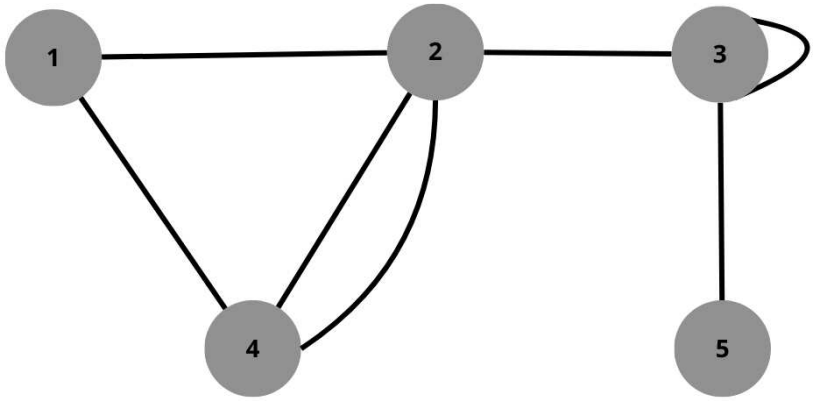


Fonte: elaborado pelo autor (2025).

No caso da Figura 2, o conjunto de vértices é $V = \{1,2,3,4,5\}$ e o conjunto de arestas é $E = \{(1,2),(2,3),(2,4),(3,5),(4,1)\}$.

Já os grafos do tipo *multígrafos* permitem a existência de laços, ou seja, uma aresta que conecta um vértice a ele mesmo. A Figura 3 exemplifica esse conceito no vértice 3.

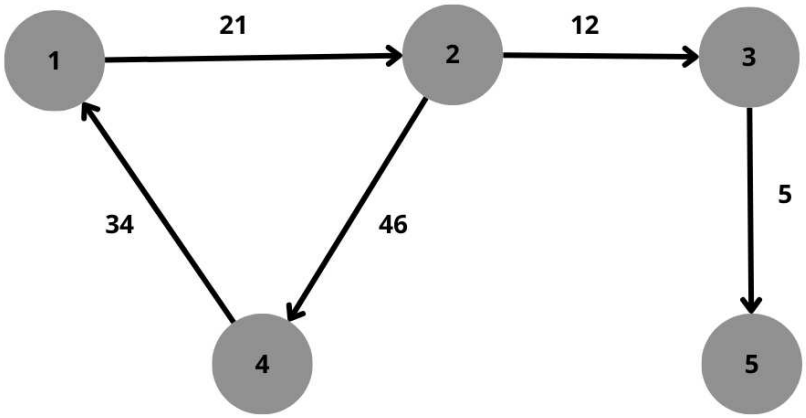
Figura 3 – Grafo multígrafo não direcionado.



Fonte: elaborado pelo autor (2025).

Os grafos também podem ser classificados como *ponderados*, onde cada aresta possui um valor atribuído, como ilustra a Figura 4 (WEST, 2001; CORMEN *et al.*, 2009).

Figura 4 – Grafo simples direcionado e ponderado.



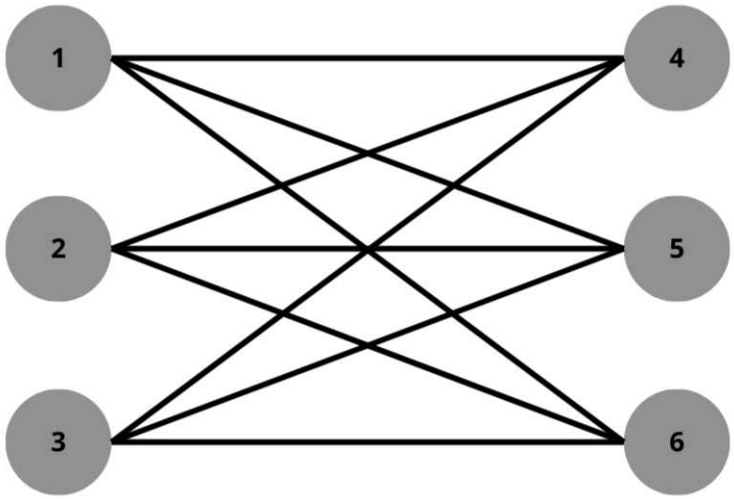
Fonte: elaborado pelo autor (2025).

Um caso clássico que envolve grafos ponderados é o problema do caminho mínimo, como o algoritmo de Dijkstra (CORMEN *et al.*, 2009).

Além dessas classificações citadas acima, existem outras também importantes, como os grafos *completos* onde todos os vértices são conectados por uma aresta, ou seja, há uma aresta para cada par de vértices (WEST, 2001).

Outra classficação importante são os grafos *bipartidos*, os que possuem um conjunto de vértices divididos em subconjuntos U e W sem que haja arestas entre os vértices de um mesmo subconjunto. A Figura 5 ilustra um grafo bipartido e completo.

Figura 5 – Grafo bipartido completo.

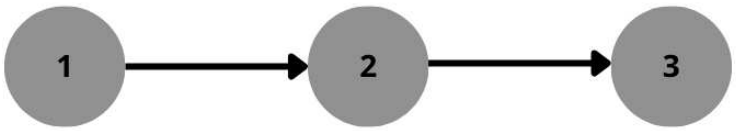


Fonte: elaborado pelo autor (2025).

Há também os grafos *cíclicos*, como o da Figura 1, que contém um ciclo entre os

vértices 1, 2 e 4, e *acíclicos*, sendo que os acíclicos direcionados (Figura 6) (Directed Acyclic Graph (DAG)'s – Directed Acyclic Graphs) são indispensáveis para *pipelines* de dados, onde um *step* depende do outro para que seja executado, como exemplo prático onde esse conceito é aplicado podemos citar o Apache Airflow.

Figura 6 – Grafo acíclico direcionado.



Fonte: elaborado pelo autor (2025).

2.2.2 Propriedades do Grafos

As propriedades dos grafos definem aspectos de sua estrutura que interferem diretamento no comportamento de algorítmos e na complexidade do seu processamento. Entre as principais propriedades estão:

- **Grau de um vértice**: número de arestas incidentes a um vértice. Em grafos direcionados, distinguem-se grau de entrada (*in-degree*) e grau de saída (*out-degree*).
- **Densidade**: razão entre o número de arestas presentes e o número máximo possível de arestas. Essa medida permite avaliar o quão "cheio" ou "conectado" está o grafo.
- **Diâmetro**: maior distância entre quaisquer dois vértices, medida em número de arestas, sendo um indicador da "largura" da rede.
- **Componentes conexas**: subconjuntos de vértices em que há caminhos entre quaisquer dois elementos.
- **Caminhos e ciclos**: sequências de vértices conectados que permitem a análise de rotas, dependências e repetições dentro do grafo.

Essas propriedades são fundamentais tanto na análise teórica quanto na aplicação prática de algoritmos sobre grafos, especialmente em contextos de larga escala (WEST, 2001).

2.3 Modelos Computacionais para Processamento de Grafos

A implementação de algorítmos sobre grafos massivos necessita de modelos computacionais eficientes para processsar estruturas complexas e distribuir o processammento entre

múltiplos nós de forma equilibrada. Entre esse modelos destacam-se o *vertex-centric*, que apresenta uma abordagem descentralizada baseada nos vértices individuais; o modelo Bulk Synchronous Parallel (BSP) (Bulk Synchronous Parallel), que organiza o processamento em *supersteps* com sincronização global; e outras variações como e outras variações como os modelos *edge-centric*, Gather-Apply-Scatter (GAS) (Gather-Apply-Scatter), *dataflow*, entre outros. Esta seção apresenta os principais modelos usados em plataformas de grafos massivos.

2.3.1 Modelo Vertex-Centric

O modelo *vertex-centric*, também conhecido como *think like a vertex*, é uma abordagem computacional para o processamento de grafos que foca a execução a partir da perspectiva de cada vértice individual. Em vez de operar sobre o grafo como um todo, cada vértice executa logicamente de forma independente, comunicando-se com seus vizinhos por meio de mensagens e atualizando seu próprio estado local com base nessas mensagens.

Este paradigma foi formalizado com o lançamento do modelo Pregel, proposto por Malewicz *et al.* (2010), o qual foi inspirado no modelo BSP (*Bulk Synchronous Parallel*) de Valiant (1990). A ideia central do Pregel é dividir o processamento em *superpassos* (*supersteps*), onde todos os vértices ativos executam uma função definida pelo usuário, trocam mensagens com outros vértices e, ao final do superpasso, sincronizam-se globalmente.

Cada vértice mantém:

- um estado local (por exemplo, valor do vértice);
- uma lista de vizinhos (arestas incidentes);
- e uma função computacional associada que é chamada a cada superpasso.

Durante o processamento, os vértices enviam mensagens para seus vizinhos que serão processadas no superpasso seguinte. A computação prossegue até que todos os vértices estejam inativos (ou seja, não enviem mensagens), caracterizando um ponto de convergência do algoritmo.

Esse modelo oferece diversas vantagens:

- escalabilidade: permite o particionamento do grafo em múltiplas máquinas;
- simplicidade de programação: o desenvolvedor escreve a lógica de um único vértice;
- paralelismo natural: os vértices ativos podem ser processados em paralelo.

Por outro lado, o modelo *vertex-centric* também possui limitações importantes. A sincronização global entre superpassos pode introduzir sobrecargas significativas, especialmente

em casos de baixa densidade de comunicação ou quando há alto desequilíbrio de carga entre partições (GONZALEZ *et al.*, 2012).

Frameworks como Apache Giraph e Hama adotam esse modelo diretamente, enquanto plataformas como GraphLab e PowerGraph o estendem com variações assíncronas (ex.: o modelo GAS — Gather, Apply, Scatter) para mitigar as limitações da sincronização global (GONZALEZ *et al.*, 2012).

O modelo *vertex-centric* tornou-se a base conceitual dominante para diversas plataformas de grafos em larga escala, e sua simplicidade continua a influenciar novos modelos, inclusive em ambientes modernos como computação *serverless* e *dataflow* distribuído.

2.3.2 Modelo BSP (Bulk Synchronous Parallel)

O modelo *Bulk Synchronous Parallel* (BSP) foi introduzido por Valiant (1990) como uma proposta para preencher a lacuna entre o modelo teórico de paralelismo e a computação paralela prática. Seu objetivo principal é oferecer um modelo simples e eficiente que possa ser implementado sobre diferentes arquiteturas de hardware paralelas.

O BSP organiza o processamento paralelo em uma sequência de etapas chamadas *superpassos* (*supersteps*), cada um composto por três fases distintas:

1. **Cálculo local**: cada unidade de processamento executa operações sobre seus dados locais.
2. **Comunicação**: as unidades trocam mensagens entre si de forma assíncrona.
3. **Barreira de sincronização**: todas as unidades aguardam até que todas tenham concluído o superpasso antes de avançar para o próximo.

A abstração BSP provê um modelo de paralelismo determinístico e estruturado, favorecendo o design e a análise de algoritmos distribuídos. Essa abordagem é particularmente eficaz em aplicações em que as dependências entre os dados são claramente definidas por mensagens — como é o caso do processamento de grafos.

O modelo BSP foi adotado como base para o sistema Pregel (MALEWICZ *et al.*, 2010), no qual cada vértice do grafo atua como uma unidade de computação independente, que se comunica com seus vizinhos e sincroniza com os demais vértices ao final de cada superpasso. Essa estrutura permite paralelizar algoritmos como Breadth-First Search (BFS), PageRank, Single-Source Shortest Path (SSSP), entre outros, com controle explícito sobre o fluxo de dados e a sincronização.

Apesar de suas vantagens, o BSP apresenta limitações práticas, especialmente em

ambientes heterogêneos ou de alta latência. A barreira de sincronização pode causar gargalos significativos quando o tempo de processamento entre as unidades é desequilibrado, levando à subutilização de recursos. Modelos mais recentes, como o GAS (*Gather-Apply-Scatter*) e arquiteturas assíncronas, surgem em parte como resposta a essas limitações (GONZALEZ *et al.*, 2012).

Ainda assim, o BSP permanece como um modelo fundamental na computação paralela, sendo a base de diversos frameworks modernos de grafos em larga escala.

O uso do BSP em grafos, apesar de eficiente para certos padrões de comunicação e paralelismo, apresenta limitações importantes em contextos reais de larga escala. Conforme destacado por McColl *et al.* (2014), a principal desvantagem do BSP em ambientes distribuídos heterogêneos é o impacto da sincronização global no tempo total de execução. Nesses casos, o tempo de espera nos pontos de barreira (superpasso) pode ser significativo, principalmente quando há desequilíbrio de carga entre os *workers* ou variação na latência de rede.

Além disso, alguns algoritmos não se adaptam de maneira natural à estrutura em superpassos. Por exemplo, Yan *et al.* (2014) demonstram que algoritmos como Dijkstra e componentes fortemente conexas (Strongly Connected Components (SCC)) exigem adaptações complexas quando implementados sob o paradigma BSP, resultando em sobrecarga computacional e aumento no número de mensagens trocadas.

2.3.3 Modelo Edge-Centric

O modelo *edge-centric* surge como uma alternativa ao paradigma *vertex-centric*, especialmente em contextos onde o volume de arestas é significativamente maior do que o de vértices, como ocorre em grafos esparsos mas densamente conectados localmente. Nesse modelo, a computação ocorre a partir da perspectiva das arestas, ou seja, cada aresta é a unidade de computação que acessa os dados de seus vértices de origem e destino, e executa uma função sobre esses dados.

O *edge-centric* busca otimizar o acesso sequencial ao grafo armazenado em disco ou memória secundária, reduzindo o custo de Entrada e Saída (E/S) (entrada e saída) e melhorando o desempenho em ambientes com restrição de memória. Em vez de manter todo o grafo na memória — o que pode ser inviável em grafos massivos —, esse modelo permite o particionamento das arestas em blocos sequenciais, que são processados como fluxos (*streams*) de dados.

Um dos sistemas que popularizou esse modelo foi o *X-Stream*, proposto por Roy

et al. (2013), que introduziu o conceito de *streaming partitions*. Nesse sistema, as arestas são armazenadas em disco em formato linear e lidas sequencialmente para minimizar o custo de acesso aleatório. O X-Stream alterna entre duas fases: a fase de *gather*, onde os dados são coletados de cada aresta; e a fase de *apply*, onde os dados coletados são aplicados de volta aos vértices.

As principais vantagens do modelo *edge-centric* incluem:

- maior eficiência em acesso a disco, aproveitando leitura sequencial;
- melhor uso de memória, permitindo processar grafos que excedem a capacidade de Random Access Memory (RAM);
- modelo naturalmente alinhado a arquiteturas baseadas em fluxo de dados.

Em comparação ao *vertex-centric*, o *edge-centric* favorece *workloads* com alta densidade de arestas e algoritmos baseados em agregações ou difusões amplas (como PageRank). Sua aplicação é particularmente vantajosa em ambientes de armazenamento limitado, sendo uma alternativa promissora para execução em sistemas de arquivos distribuídos ou dispositivos com restrições de RAM.

2.3.4 Modelo GAS (Gather-Apply-Scatter)

O modelo *GAS* (Gather-Apply-Scatter) foi introduzido como uma evolução do paradigma *vertex-centric*, com o objetivo de oferecer maior flexibilidade e desempenho em ambientes distribuídos, especialmente em grafos naturais, caracterizados por alto grau de irregularidade na distribuição das conexões entre vértices. Essa abordagem foi popularizada pela plataforma PowerGraph, proposta por Gonzalez *et al.* (2012).

O GAS divide o processamento de cada vértice em três fases distintas:

1. **Gather**: o vértice coleta informações de seus vizinhos a partir das arestas incidentes, acumulando um valor parcial (ex.: soma, máximo, média).
2. **Apply**: o vértice atualiza seu valor com base no resultado da agregação realizada na fase anterior.
3. **Scatter**: o vértice propaga o novo valor às arestas conectadas, atualizando o estado das conexões ou informando os vizinhos.

Essa decomposição favorece maior controle sobre o fluxo de dados e permite otimizações adicionais, como a execução assíncrona e o balanceamento mais eficiente da carga de trabalho entre os nós. A abordagem GAS é particularly útil em grafos com distribuição de

grau altamente desigual (como redes sociais ou da Web), onde a estratégia de espelhamento de vértices (*vertex-cut*) pode ser mais eficiente que a tradicional partição por arestas ou vértices.

Diferentemente do BSP tradicional, o modelo GAS permite que diferentes partes do grafo sejam processadas em ritmos distintos, sem exigir sincronizações globais a cada iteração. Essa característica reduz o tempo de espera entre partições e melhora o aproveitamento dos recursos computacionais, especialmente em clusters com hardware heterogêneo ou tarefas irregulares.

Conforme demonstrado por Gonzalez *et al.* (2012), essa abordagem melhora significativamente a escalabilidade em comparação com sistemas baseados exclusivamente no modelo Pregel. Em experimentos realizados com grafos reais contendo bilhões de arestas, o PowerGraph apresentou ganhos expressivos de desempenho ao reduzir a sobrecarga de comunicação e evitar o congestionamento em vértices de alto grau.

2.3.5 Modelo baseado em Fluxo de Dados (Dataflow)

O modelo baseado em *fluxo de dados* (*dataflow*) é uma abordagem computacional amplamente utilizada em plataformas de processamento distribuído orientadas a dados, como Apache Spark, Flink e Dryad. Diferentemente dos modelos centrados em vértices ou arestas, o *dataflow* organiza a computação como um grafo acíclico direcionado (DAG), em que os nós representam operações e as arestas representam o fluxo de dados entre essas operações.

No contexto do processamento de grafos em larga escala, o modelo *dataflow* foi incorporado por sistemas como o *GraphX*, que estende o Apache Spark para suportar computações sobre grafos (XIN *et al.*, 2013). Nesse modelo, o grafo é representado por estruturas imutáveis de dados — como Resilient Distributed Datasets (RDDs) (*Resilient Distributed Datasets*) —, e os algoritmos são expressos como sequências de transformações funcionais (ex.: `map`, `reduce`, `join`) sobre essas estruturas.

Uma das principais vantagens desse modelo é a capacidade de integrar o processamento de grafos com outros tipos de *workloads* de dados, como consultas Structured Query Language (SQL), aprendizado de máquina ou transformações em dados tabulares. Isso proporciona grande interoperabilidade e flexibilidade, especialmente em *pipelines* de dados heterogêneos.

Além disso, o modelo *dataflow* facilita a otimização global do plano de execução, pois o sistema pode reordenar, combinar ou distribuir as operações com base na análise do DAG. Essa característica é fundamental para alcançar eficiência em *clusters* de larga escala.

Contudo, o modelo também apresenta limitações. Como apontado por Gonzalez *et al.* (2014), o uso de estruturas imutáveis implica na reconstrução parcial do grafo a cada iteração, o que pode gerar sobrecarga significativa em algoritmos iterativos. Essa limitação é particularmente evidente em algoritmos de propagação de rótulo, como PageRank, que requerem múltiplas iterações.

Apesar disso, o modelo *dataflow* tem se mostrado altamente eficaz em ambientes de *Big Data* e *cloud computing*, sendo uma escolha natural para aplicações que exigem integração entre processamento de grafos e outras formas de análise de dados.

Com base nos modelos apresentados, é possível observar que cada abordagem oferece vantagens específicas diante dos desafios impostos pelo processamento de grafos em larga escala. Enquanto modelos como o *vertex-centric* e o BSP destacam-se por sua simplicidade e estrutura formal, abordagens como GAS e *dataflow* demonstram maior adaptabilidade a cargas de trabalho irregulares e integração com ambientes analíticos modernos. A Tabela 2 apresenta uma comparação entre os cinco principais modelos computacionais discutidos, destacando suas características, benefícios e limitações. Essa análise será fundamental para a avaliação das plataformas encontradas nos estudos analisados na revisão sistemática, especialmente no que se refere à eficiência, escalabilidade, interoperabilidade e desafios ainda abertos.

Tabela 2 – Comparação entre Modelos Computacionais para Processamento de Grafos

Modelo	Características	Vantagens	Limitações
Vertex-Centric	Cada vértice executa logicamente, enviando mensagens a seus vizinhos a cada superpasso.	Simplicidade de programação, paralelismo natural, base do modelo Pregel.	Exige sincronização global, sensível a balanceamento de carga.
BSP (Bulk Synchronous Parallel)	Computação estruturada em superpassos com fases de cálculo, comunicação e sincronização.	Determinismo, fácil depuração, base formal sólida.	Sincronização pode causar gargalos; baixa adaptabilidade a workloads irregulares.
Edge-Centric	A computação é feita pelas arestas, permitindo leitura sequencial e uso eficiente de disco.	Excelente para E/S sequencial, útil em sistemas com pouca RAM.	Dificuldade em algoritmos com forte dependência entre vértices; particionamento complexo.
GAS (Gather-Apply-Scatter)	Divide a execução em três fases; permite execução assíncrona e balanceamento refinado.	Escalabilidade superior em grafos irregulares; evita sincronizações desnecessárias.	Requer estruturação mais complexa do grafo; menos intuitivo para iniciantes.
Dataflow	Grafo de operações sobre dados imutáveis (DAG); integra-se com outras cargas analíticas.	Alta interoperabilidade; ideal para pipelines de Big Data.	Estruturas imutáveis geram overhead em algoritmos iterativos.

Fonte: Elaborado pela autora com base em Malewicz *et al.* (2010), Valiant (1990), Roy *et al.* (2013), Gonzalez *et al.* (2012), Xin *et al.* (2013).

2.4 Plataformas de Processamento de Grafos em Larga Escala

Diversas plataformas computacionais foram desenvolvidas ao longo da última década com o objetivo de viabilizar o processamento eficiente de grafos massivos em ambientes distribuídos. Essas plataformas combinam diferentes modelos computacionais, estratégias de particionamento e mecanismos de tolerância a falhas para lidar com o crescimento exponencial de dados interconectados em aplicações como redes sociais, bioinformática, cibersegurança e sistemas de recomendação.

A seguir, são apresentadas algumas das principais plataformas, com destaque para suas arquiteturas, modelos subjacentes e propostas técnicas.

Pregel

Proposta por Malewicz *et al.* (2010), a Pregel foi uma das primeiras plataformas desenhadas especificamente para o processamento de grafos em larga escala. Baseada no modelo *vertex-centric* e no paradigma BSP, a Pregel permite que cada vértice atue de forma autônoma, enviando mensagens a seus vizinhos em superpassos sincronizados. Essa plataforma foi pioneira na definição de uma Application Programming Interface (API) de alto nível para algoritmos sobre grafos distribuídos, influenciando diversos sistemas posteriores.

Apache Giraph

Inspirado diretamente na Pregel, o Apache Giraph é uma implementação *open-source* escrita em Java que roda sobre o ecossistema Hadoop. Ele preserva o modelo *vertex-centric* e a estrutura BSP, mas introduz melhorias como suporte a *multithreading* e compressão de mensagens. Giraph foi utilizado pelo Facebook para processar grafos com mais de um trilhão de arestas (CHING *et al.*, 2015).

GraphLab e PowerGraph

As plataformas GraphLab e sua evolução, PowerGraph, propõem um modelo de execução assíncrona baseado na decomposição GAS (Gather-Apply-Scatter). Essa abordagem busca reduzir os gargalos de sincronização do BSP, permitindo uma execução mais eficiente em grafos irregulares, com vértices de alto grau. O PowerGraph também introduz o particionamento por vértice (*vertex-cut*), o que melhora o balanceamento de carga em determinadas topologias (GONZALEZ *et al.*, 2012).

X-Stream

X-Stream adota uma abordagem *edge-centric*, otimizando o acesso sequencial aos dados do grafo. Com foco em eficiência de E/S e uso limitado de memória principal, essa plataforma permite o processamento de grafos muito maiores que a RAM disponível, sendo especialmente útil em ambientes de recursos restritos (ROY *et al.*, 2013).

GraphX

GraphX é um framework integrado ao Apache Spark, baseado em *dataflow*, que permite unificar o processamento de grafos com tarefas analíticas tradicionais. Utilizando estruturas imutáveis como RDDs, ele oferece interoperabilidade com outras bibliotecas do ecossistema Spark, permitindo que operações sobre grafos façam parte de *pipelines* maiores de processamento de dados (XIN *et al.*, 2013).

Plataformas Emergentes

Mais recentemente, surgiram plataformas que combinam novas arquiteturas e paradigmas, como PrefEdge, que explora estratégias de pré-busca em Solid State Drives (SSDs) para acelerar a travessia de grafos (NILAKANT *et al.*, 2014), Granula, que introduz análise de desempenho em nível granular (NGAI *et al.*, 2017), e soluções baseadas em Function-as-a-Service (FaaS) (*Function-as-a-Service*) e *workflows serverless* para execução elástica e sob demanda de algoritmos distribuídos (RISTOV *et al.*, 2023). Tais abordagens ampliam o escopo de aplicação das plataformas e apontam para tendências futuras em ambientes de *cloud computing*.

Essas plataformas serão analisadas em maior profundidade no Capítulo 4, onde serão discutidas suas contribuições, limitações e potencial evolutivo em relação aos desafios centrais deste trabalho: eficiência, escalabilidade, segurança e interoperabilidade.

Diante do exposto, observa-se que o processamento de grafos em larga escala demanda uma base técnica sólida que envolve desde conceitos fundamentais da teoria dos grafos até arquiteturas computacionais distribuídas. Os modelos computacionais apresentados — *vertex-centric*, BSP, *edge-centric*, GAS e *dataflow* — constituem os alicerces sobre os quais as plataformas modernas foram construídas. Compreender essas estruturas é essencial para analisar criticamente as propostas desenvolvidas ao longo da última década e suas contribuições frente aos desafios de eficiência, escalabilidade, segurança e interoperabilidade. A próxima etapa deste trabalho consiste na apresentação da metodologia utilizada para conduzir a revisão sistemática da literatura, a qual fundamenta a análise dos estudos selecionados e a resposta às questões de pesquisa definidas.

3 METODOLOGIA

Este capítulo apresenta a metodologia aplicada na execução da revisão sistemática da literatura, cujo objetivo foi identificar e analisar a evolução das plataformas de processamento de grafos em larga escala, bem como as tendências emergentes para enfrentar os desafios de eficiência, escalabilidade, segurança e interoperabilidade. Esta revisão seguiu as diretrizes metodológicas propostas por Kitchenham e Charters (2007). A escolha pelas diretrizes metodológicas propostas por Kitchenham e Charters (2007) justifica-se pelo fato de serem amplamente reconhecidas e adotadas na área de Engenharia de Software e Ciência da Computação como referência para a condução de revisões sistemáticas da literatura. Essas diretrizes fornecem um conjunto estruturado de etapas que asseguram a reprodutibilidade, a rastreabilidade e a transparência do processo de revisão, desde a formulação das perguntas de pesquisa até a síntese dos resultados. Além disso, permitem a elaboração de protocolos formais, a aplicação criteriosa de critérios de inclusão e exclusão, bem como a documentação rigorosa de todo o processo, aspectos fundamentais para garantir a confiabilidade dos achados obtidos neste estudo.

A condução da revisão ocorreu entre os meses de abril e maio de 2025, contemplando as seguintes etapas: definição das questões de pesquisa, desenvolvimento da estratégia de busca, aplicação dos critérios de seleção, triagem e leitura dos estudos, classificação dos estudos, extração estruturada dos dados e análise qualitativa dos resultados.

3.1 Questões de Pesquisa

As questões de pesquisa definidas foram:

- **Q1:** Como as plataformas de processamento de grafos em larga escala evoluíram ao longo do tempo, abordando desafios de eficiência, escalabilidade, segurança e interoperabilidade?
- **Q2:** Quais são as tendências futuras previstas para enfrentar os desafios emergentes de eficiência, escalabilidade, segurança e interoperabilidade nas plataformas de processamento de grafos em larga escala?

A revisão irá focar em plataformas de processamento de grafos em larga escala, abrangendo técnicas, algoritmos, plataformas, otimizações e métricas de desempenho.

3.2 Desenvolvimento da Estratégia de Busca

Para determinar a estratégia de busca, primeiramente foram estabelecidos os parâmetros PICOC (População, Intervenção, Comparação, Resultados e Contexto), como sugerido por Kitchenham e Charters (2007).

- **(P) População:** Processamento de grafos em larga escala.
- **(I) Intervenção:** Plataformas de processamento de grafos em larga escala.
- **(C) Comparação:** A comparação envolve a evolução, melhorias, desafios e tendências das plataformas de processamento de grafos em larga escala.
- **(O) Resultado (*Output*):** Os resultados esperados incluem aspectos de desempenho como eficiência, escalabilidade, segurança e interoperabilidade.
- **(C) Contexto:** Futuro e inovações emergentes no campo.

Abaixo a tabela 3 associa os parâmetros apresentados com os termos da string de busca utilizada.

Tabela 3 – Associação dos Parâmetros PICOC com Termos de Busca

Parâmetro	**Termos da Busca**
População	"big graph processing"OR "large-scale graph processing"
Intervenção	"graph processing platforms"OR "graph analysis platforms"
Comparação	"evolution"OR "improvements"OR "challenges"OR "trends"
Resultado	"efficiency"OR "scalability"OR "security"OR "interoperability"
Contexto	"future"OR "emerging innovations"

Fonte: Elaborado pela autora (2024).

A string de busca obtida pela associação feita acima é a seguinte:

("big graph processing"OR "large-scale graph processing") AND ("graph processing platforms"OR "graph analysis platforms") AND ("evolution"OR "improvements"OR "challenges"OR "trends") AND ("efficiency"OR "scalability"OR "security"OR "interoperability") AND ("future"OR "emerging innovations")

A busca foi limitada aos anos de 2013 a 2023 e foi realizada em duas bases de dados: Scopus e ScienceDirect.

3.3 Seleção dos Estudos

A seleção dos estudos seguiu um processo rigoroso para garantir a relevância e qualidade dos artigos incluídos, conforme descrito a seguir.

3.3.1 Critérios de Inclusão

- Estudos que abordam diretamente o processamento de grafos em larga escala, incluindo técnicas, algoritmos, plataformas, otimizações, métricas de desempenho e aspectos relacionados.
- Trabalhos escritos em inglês ou português, com texto completo disponível.
- Artigos publicados em conferências ou periódicos revisados por pares.
- Publicações no intervalo temporal de 2013 a 2023.

3.3.2 Critérios de Exclusão

- Estudos em idiomas diferentes do inglês ou português.
- Publicações anteriores a 2013.
- Trabalhos que não tratam de grafos em larga escala.
- *Surveys*, revisões de literatura ou meta-análises (estudos secundários) não extraídos como primários.

3.3.3 Processo de Seleção

O processo de seleção foi dividido em três fases:

1. **Triagem Inicial:** remoção de duplicatas e aplicação preliminar dos critérios.
2. **Revisão de Títulos e Resumos:** avaliação de aderência aos critérios.
3. **Leitura Completa:** confirmação da relevância com leitura integral dos artigos.

3.3.4 Classificação dos Estudos: Primários x Secundários

Nesta revisão, os estudos foram categorizados em *primários* e *secundários*. Estudos primários são aqueles que apresentam contribuições originais com propostas de novos algoritmos, técnicas ou frameworks, avaliação empírica ou *benchmarks* em ambientes reais ou simulados e/ou dados e métricas de desempenho extraídos diretamente de experimentos com o objetivo garantir

rigor empírico permitindo extração de métricas de primeira mão para a Tabela 4. Enquando que estudos secundários são *surveys*, revisão sistemáticas e meta-análises que compilam resultados de múltiplos estudos primários. Nesta revisão, eles serão utilizados apenas para contextualização do estado da arte, sem extração direta de dados padronizados para evitar viéses e preservar a consitência na coleta de dados empíricos.

3.3.5 Extração dos Dados

Os dados dos estudos primários selecionados foram extraídos utilizando a tabela padronizada (Tabela 4), contendo campos como Título, Autores, Ano de Publicação, Objetivo, Metodologia, Métricas, Aspectos de Eficiência, Escalabilidade, Segurança e Interoperabilidade.

Tabela 4 – Informações dos Estudos Selecionados

Campo	**Descrição**
Título	Título completo do artigo.
Autores	Lista dos autores do estudo.
Ano de Publicação	Ano em que o artigo foi publicado.
Fonte/Biblioteca	Base de dados onde o artigo foi encontrado (ex.: Scopus, ScienceDirect, ACM).
Tipo de Publicação	Tipo de artigo (ex.: Conferência, Revista).
Objetivo do Estudo	Objetivo principal do estudo.
Metodologia	Descrição da metodologia utilizada no estudo.
Técnica/Algoritmos	Técnica ou algoritmos abordados no estudo.
Plataformas	Plataformas de processamento de grafos discutidas no estudo.
Principais Resultados	Resultados principais encontrados no estudo.
Conclusões	Conclusões apresentadas pelos autores.
Limitações	Limitações do estudo mencionadas pelos autores.
Tendências Futuras	Tendências futuras mencionadas pelos autores.
Desafios	Desafios abordados no estudo.
Métricas de Desempenho	Métricas de desempenho usadas na avaliação.
Aspectos de Eficiência	Aspectos de eficiência discutidos no estudo.
Aspectos de Escalabilidade	Aspectos de escalabilidade discutidos no estudo.
Aspectos de Segurança	Aspectos de segurança discutidos no estudo.
Aspectos de Interoperabilidade	Aspectos de interoperabilidade discutidos no estudo.

Fonte: Elaborado pela autora (2025).

3.3.6 Síntese e Análise dos Resultados

A análise dos dados extraídos será conduzida para identificar padrões e tendências, com:

- **Análise qualitativa:** identificação de temas emergentes.
- **Síntese dos resultados:** resumo das descobertas principais.
- **Discussão crítica:** comparação com as questões de pesquisa e identificação de lacunas.

4 RESULTADOS

4.1 Resumo Quantitativo dos Estudos Primários

Nesta seção apresentamos um panorama numérico dos 13 estudos primários extraídos, conforme o protocolo descrito.

4.1.1 Triagem e Seleção

A aplicação das estratégias de busca e dos critérios de inclusão/exclusão resultou na seleção de 13 estudos primários, a partir de um total de 32 publicações inicialmente identificadas. A Tabela 5 resume cada fase do processo.

Tabela 5 – Resumo quantitativo das fases de seleção

Fase	**Quantidade**
Busca inicial	32
Triagem de títulos e resumos	25
Estudos primários completos	13

Fonte: Elaborado pela autora (2025).

4.1.2 Distribuição Temporal

A Figura 7 mostra a quantidade de estudos primários por ano de publicação (2013–2023).

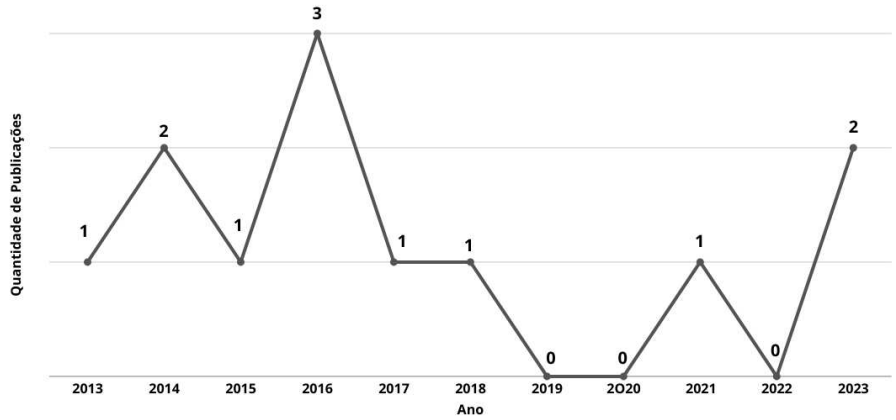


Figura 7 – Número anual de estudos primários (2013–2023).
Fonte: Elaborado pela autora (2025).

4.1.3 Tipologia de Publicação

A Tabela 6 apresenta a distribuição dos 13 estudos por tipo de publicação. Constatou-se predomínio de artigos de conferência (aprox. 69%) em relação aos artigos de periódico (aprox. 31%), reflexo do caráter experimental e de prova de conceito das propostas.

Tabela 6 – Classificação dos estudos primários por tipo de publicação

Tipo de Publicação	**Contagem**	**%**
Artigos de conferência (*peer-reviewed*)	9	69%
Artigos de periódico (*peer-reviewed*)	4	31%

Fonte: Elaborado pela autora (2025).

4.2 Evolução Tecnológica das Plataformas em "Ondas" Temáticas

Para responder à **Q1** — como as plataformas de grafos em larga escala evoluíram — organizamos os 13 estudos primários em quatro *ondas* temáticas, cada uma caracterizada por avanços centrais e desafios associados. A Tebela 7 ilustra os artigos correspondentes de cada onda. Chamamos de onda cada período com caracteísticas específicas.

Tabela 7 – Quatro ondas temáticas na evolução de plataformas de grafos (2013–2023)

Período & Tema	Principais Estudos e Contribuições
Onda 1 (2013–2015) Consolidação *Vertex-Centric*	• *A Yoke of Oxen & a Thousand Chickens* (**??**): *pipeline* híbrido CPU+GPU, até 5× *speed-up*. • *The Energy Case* (GHARAIBEH *et al.*, 2013): eficiência energética TEPS/W 1.9× com DVFS. • *One Trillion Edges* (CHING *et al.*, 2015): 1 trilhão de arestas em 200 nós, 120× vs. Hive.
Onda 2 (2014–2017) I/O e Diversificação de Arquiteturas	• *PrefEdge* (NILAKANT *et al.*, 2014): SSD *prefetch*, 14× *speed-up*, –80% latência E/S. • *FENNEL* (TSOURAKAKIS *et al.*, 2014): particionamento *streaming* para grafos massivos. • *Resource Dependency* (**??**): modelos de dados e *cache* em *frameworks web*. • *Parallel Multi-core* (**??**): otimizações em servidores *multi-core*. • *Intelligent Processes* (WANG *et al.*, 2017): agentes autônomos para balancear carga.
Onda 3 (2017–2018) Observabilidade e Elasticidade	• *Granula* (NGAI *et al.*, 2017): *profiling fine-grained* mostrou 70% de tempo ocioso em I/O e sincronização. • *Auto-Scaling Graph* (HEIDARI; BUYYA, 2019)): reduziu 34% do custo-dólar sem degradar latência.
Onda 4 (2021–2023) Sustentabilidade, Temporalidade e *Serverless*	• *GRADOOP* (ROST *et al.*, 2021): até 18× vs. Neo4j e 5× vs. GraphX. • *Serverless Workflows* (RISTOV *et al.*, 2023): FaaS federado – desafios de *cold-start* e concorrência. • *Graph-Massivizer* (MOLAN *et al.*, 2023): *digital twin* sustentável – AUC 0.86 em anomalias, –10% PUE.

Fonte: Elaborado pela autora (2025).

4.3 Mapeamento de Desafios e Tendências

Nesta seção, atendemos à **Q2**:

Quais são as tendências futuras previstas para enfrentar os desafios emergentes de eficiência, escalabilidade, segurança e interoperabilidade nas plataformas de processamento de grafos em larga escala?

Para tanto, mapeamos quantitativamente os *Desafios* e as *Tendências Futuras* explicitados nos 13 estudos primários, relacionando-os às quatro dimensões de interesse da **Q1**.

4.3.1 Desafios Identificados

A Tabela 8 mostra a frequência em que os desafios abordados nos estudos primários aparecem.

Tabela 8 – Frequência dos principais desafios nos estudos primários

Desafio	**Nº de estudos**
Balanceamento de carga	9
Particionamento dinâmico	2
Comunicação (mensagens e E/S)	2
Tolerância a falhas	1
Sustentabilidade energética	1
Outros (compressão, privacidade etc.)	0

Fonte: Elaborado pela autora (2025), com base nos 13 estudos primários.

4.3.1.1 Balanceamento de carga

A grande maioria (9/13) destaca a distribuição desigual de trabalho entre nós como fator crítico. Desequilíbrios geram *hotspots*, desperdiçam CPU e largura de banda, e elongam a latência global.

4.3.1.2 Particionamento dinâmico

Embora fundamental para grafos mutáveis, apenas dois estudos propõem redistribuição de partições em tempo de execução, evidenciando uma lacuna em cenários de *streaming* e grafos de crescimento contínuo.

4.3.1.3 Comunicação (mensagens e E/S)

Dois trabalhos mediram gargalos de I/O e sincronização:

- *Granula* identificou até 70 % de tempo ocioso em I/O/sincronização (NGAI *et al.*, 2017).

- *PrefEdge* demonstra ganhos com leitura sequencial de SSD, mas mantêm desafios de concorrência de acesso.

4.3.1.4 Tolerância a falhas

Apenas um artigo aborda recuperação de falhas, ressaltando que *checkpoints* minimizam recomputação, porém introduzem *overhead* de armazenamento e sincronização.

4.3.1.5 Sustentabilidade energética

A eficiência (TEPS/W, PUE) surge em apenas um estudo híbrido CPU+GPU, apontando que métricas de consumo ainda não são padrão na avaliação de plataformas de grafos.

4.3.2 Tendências Futuras

A Tabela 9 mostra a frequência das tendências futuras mencionadas nos estudos primários.

Tabela 9 – Frequência das tendências futuras mencionadas nos estudos primários

Tendência	**Nº de estudos**
Grafos dinâmicos / *streaming*	5
Particionamento adaptativo	1
Arquiteturas *serverless* / FaaS federado	1
Graph Neural Networks (GNNs) (Graph Neural Networks)	1
Digital twins de *data centers*	1
Eficiência energética (TEPS/W, PUE)	1
Elasticidade *custo-aware*	0

Fonte: Elaborado pela autora (2025), com base nos 13 estudos primários.

4.3.2.1 Grafos dinâmicos e streaming

Cinco estudos defendem *frameworks* que processem atualizações incrementais em tempo real, críticos para redes sociais, Internet of Things (IoT) e monitoramento contínuo.

4.3.2.2 Particionamento adaptativo

Citado em apenas um trabalho, sugere pesquisa inicial em técnicas que reajustem cargas sem interromper a execução.

4.3.2.3 Serverless / FaaS federado

Aborda orquestração elástica com funções sob demanda, reduzindo custos mas impondo desafios de *cold-start* e interoperabilidade entre provedores.

4.3.2.4 GNNs e Digital Twins

A incorporação de GNNs (Graph Neural Networks) e a criação de gêmeos digitais de *data centers* indicam direções para predição de desempenho e otimização energética.

4.3.2.5 Lacunas em eficiência energética e elasticidade

Surpreendentemente não aparecem como "tendência" apesar dos ganhos empíricos. Há necessidade de:

- Definir *benchmarks* energéticos e Service-Level Agreements (SLAs) de consumo.
- Projetar políticas automáticas de escalonamento custo-desempenho.

4.3.3 Relação com Eficiência, Escalabilidade, Segurança e Interoperabilidade

- **Balanceamento de carga**
  - *Eficiência*: elimina *hotspots*, melhora *throughput*.
  - *Escalabilidade*: permite crescimento quase linear.
  - *Segurança*: nós equilibrados reduzem riscos de falha concentrada.
  - *Interoperabilidade*: *vertex-cut* dinâmico requer APIs padronizadas para coordenação.
- **Particionamento dinâmico**
  - *Eficiência*: ajusta carga em tempo real, reduz *overhead* de comunicação.
  - *Escalabilidade*: mantém desempenho em grafos mutáveis.
  - *Segurança*: redistribuição deve preservar consistência e integridade.
  - *Interoperabilidade*: protocolos de migração precisam atuar sobre múltiplos sistemas de armazenamento.
- **Comunicação (mensagens e E/S)**
  - *Eficiência*: *prefetch* e *streaming* reduzem latência de *supersteps*.
  - *Escalabilidade*: leitura sequencial e *caching* mitigam custos em *clusters* grandes.
  - *Segurança*: necessidade de criptografia e autenticação em canais de mensagem.
  - *Interoperabilidade*: uso de formatos padrão (e.g., Protobuf) facilita integração.

- **Tolerância a falhas**
  - *Eficiência*: *checkpoints* frequentes reduzem recomputação.
  - *Escalabilidade*: recuperações rápidas mantêm disponibilidade.
  - *Segurança*: apenas um estudo menciona confidencialidade indiretamente—lacuna crítica.
  - *Interoperabilidade*: estratégias de *checkpoint* devem funcionar com diversos *back-ends*.
- **Sustentabilidade energética**
  - *Eficiência*: métricas TEPS/W e PUE quantificam consumo por *workload*.
  - *Escalabilidade*: arquiteturas híbridas ajustam recursos para melhor relação energia-desempenho.
  - *Segurança*: menor consumo reduz superfície de ataque em *data centers*.
  - *Interoperabilidade*: exposições via APIs permitem orquestração coordenada de políticas de consumo.
- **Grafos dinâmicos / streaming**
  - *Eficiência*: processa apenas deltas, reduz custo computacional.
  - *Escalabilidade*: suporta crescimento contínuo sem reprocessar todo o grafo.
  - *Segurança*: fluxos em tempo real exigem criptografia e controle de acesso.
  - *Interoperabilidade*: conectores Kafka/Flink padronizam ingestão e saída de dados.
- **Serverless / FaaS federado**
  - *Eficiência*: paga-se-por-uso (*pay-per-use*), mas sofre com *cold-start*.
  - *Escalabilidade*: elástica por *design*, lida com picos sem provisionamento prévio.
  - *Segurança*: funções isoladas em Trusted Execution Environment (TEE) reforçam confinamento.
  - *Interoperabilidade*: padrões CloudEvents e OpenFaaS viabilizam *workflows* portáteis.
- **GNNs e Digital Twins**
  - *Eficiência*: modelos preditivos podem reduzir custo de consultas complexas.
  - *Escalabilidade*: *frameworks* de Machine Learning (ML) distribuído escalam horizontalmente.
  - *Segurança*: detecção proativa de anomalias reforça resiliência.
  - *Interoperabilidade*: APIs Open Neural Network Exchange (ONNX) e conectores

ML-Graph permitem *pipelines* integrados.

Este quadro evidencia como cada desafio e tendência afeta diretamente eficiência, escalabilidade, segurança e interoperabilidade, apontando tanto áreas consolidadas como lacunas críticas para futuras pesquisas.

4.4 Comparação de Métricas-Chave

Nesta seção comparamos quantitativamente as principais métricas de desempenho reportadas nos 13 estudos primários, respondendo parcialmente à **Q1** (eixo "eficiência" e "escalabilidade"). Destacamos, em particular, *speed-up*, TEPS/W e outras medidas de *throughput* e latência.

4.4.1 Tabela de Métricas Reportadas

Para consolidar os achados empíricos dos estudos selecionados, os principais resultados quantitativos de desempenho foram extraídos e organizados. A Tabela 10 apresenta uma compilação dessas métricas, destacando os valores reportados para *speed-up*, eficiência energética (TEPS/W), *throughput*, latência e redução de custos. Cada entrada na tabela especifica o estudo, a plataforma ou técnica avaliada e o contexto da medição, permitindo uma comparação direta dos ganhos de desempenho obtidos por cada abordagem.

Tabela 10 – Principais métricas de desempenho nos estudos primários

Estudo	**Plataforma**	**Métrica**	**Valor Reportado**	**Referência**
Gharaibeh et al. (2013)	TOTEM (CPU+GPU)	*Speed-up*	5×	vs. implementação *single-thread* CPU
Gharaibeh et al. (2013)	TOTEM + DVFS	TEPS/W	1.9×	vs. configuração 2× CPU sem DVFS
Ching et al. (2015)	One Trillion Edges: Graph Processing at Facebook-Scale	*Speed-up*	120×	vs. Hive em Hadoop (mesmo *cluster*)
Nilakant et al. (2014)	PrefEdge (SSD)	*Speed-up*	14×	vs. *baseline single-thread* SSD
Tsourakakis et al. (2014)	FENNEL (*streaming*)	Redução de tempo	15%	vs. particionamento estático uniforme
Lee et al. (2016)	Servidores *multicore*	*Throughput* (ME/s)	350M/s	vs. implementação *multithread* padrão
Wang et al. (2017)	Intelligent Processes	*Speed-up*	2×	vs. modelo *vertex-centric* puro
Ngai et al. (2017)	Granula *profiling*	Ciclos ociosos	70%	medido em Giraph/-PowerGraph
Heidari & Buyya (2018)	Auto-scaling Graph	*Custo-per-performance*	–34%	vs. AWS EC2 sem *auto-scaling*
Rost et al. (2021)	GRADOOP (Flink)	*Speed-up*	18× / 5×	vs. Neo4j / vs. GraphX
Ristov et al. (2023)	*Serverless Workflows*	Latência *cold-start*	1.2s	média em AWS Lambda
Molan et al. (2023)	Graph-Massivizer	PUE	–10%	vs. *data center* sem *digital twin*
Ching et al. (2015)	One Trillion Edges: Graph Processing at Facebook-Scale.	Iteração PageRank	<3min	em 200 nós, 1tri edge graph

Fonte: Elaborado pela autora (2025).

4.4.2 Análise Crítica

- **Ganho máximo de *speed-up*:** 120× (CHING *et al.*, 2015) em ambiente industrial contrasta com ganhos mais modestos (5×–14×) em cenários acadêmicos.
- **Eficiência energética:** apenas um estudo quantifica TEPS/W (1.9×), indicando necessidade de adoção mais ampla dessa métrica.
- ***Throughput* vs. Latência:** alto *throughput* (350M/s) em servidores *multi-core* não garante baixa latência de iteração, e paradigmas *serverless* sofrem com *cold-start* (1.2s).
- **Gargalos de I/O e sincronização:** 70% de ciclos ociosos (NGAI *et al.*, 2017) ressaltam importância de otimizações de E/S e *caching*.
- ***Benchmark* unificado:** comparações "18× vs. Neo4j e 5× vs. GraphX" (ROST *et al.*, 2021) expõem falta de cenários e dados padronizados.

Estes achados reforçam a necessidade de:

- Medir *speed-up*, *throughput*, latência e TEPS/W em um único *framework* experimental.
- Adotar *benchmarks* reproduzíveis (Graph500, GreenGraph500) e divulgar *scripts* de avaliação.
- Investigar *trade-offs* entre desempenho e consumo energético, especialmente em *clouds* e ambientes *serverless*.

4.5 Síntese Narrativa Integrada

Nesta síntese, resumimos os achados quantitativos (Secção 4.1), a evolução por ondas temáticas (Secção 4.2), a comparação de métricas (Secção 4.4) e o mapeamento de desafios e tendências (Secção 4.3), para responder de forma articulada às perguntas de pesquisa:

- **Q1:** Como as plataformas de processamento de grafos em larga escala evoluíram, em termos de eficiência, escalabilidade, segurança e interoperabilidade?
- **Q2:** Quais tendências futuras se desenham para enfrentar os desafios emergentes nessas mesmas dimensões?

4.5.1 Resposta à Q1: Evolução ao Longo das Quatro Ondas

4.5.1.1 Eficiência

Na Onda 1 (2012–2015), os estudos Gharaibeh et al. e Ching et al. focaram em aceleração bruta (*speed-up* de 5× a 120×) estendendo o modelo *vertex-centric* a CPU+GPU e grandes *clusters*. Na Onda 2 (2014–2017), as otimizações de E/S (PrefEdge, FENNEL) reduziram latências até 80%, enquanto arquiteturas *multi-core* e processos inteligentes elevaram *throughput* a centenas de milhões de arestas por segundo. A Onda 3 (2017–2018) introduziu *profiling* fino (Granula) e *auto-scaling* sensível a custo, evidenciando que até 70% do tempo de *superstep* era desperdiçado em I/O e sincronização, e reduzindo custos em 34% sem degradar latência. Já na Onda 4 (2021–2023), plataformas como GRADOOP e Graph-Massivizer começaram a considerar métricas energéticas (TEPS/W, PUE), mas ainda de forma experimental.

4.5.1.2 Escalabilidade

O modelo *vertex-centric* original se mostrou capaz de escalar linearmente até trilhões de arestas (One Trillion Edges), mas sofreu com comunicação intensa. Nas ondas seguintes, particionamento *streaming* (FENNEL) e *vertex-cut* (PowerGraph) melhoraram a distribuição de carga; a elasticidade *serverless* e *auto-scaling* (Heidari & Buyya) permitiram adequar dinamicamente recursos conforme a carga; e *digital twins* de *data centers* (Graph-Massivizer) demonstraram escalabilidade sustentável em nuvens federadas.

4.5.1.3 Segurança

Até o momento, a maior parte dos estudos primários negligenciou a segurança de dados e comunicação. Apenas um trabalho menciona confidencialidade em cenários de falha, e nenhum explora autenticação, criptografia ou *threat models*. Essa lacuna persiste ao longo das quatro ondas, indicando que segurança ainda não é considerada critério de *design* nessas plataformas.

4.5.1.4 Interoperabilidade

A integração com ecossistemas Hadoop/Spark (Giraph, GraphX) foi primordial na Onda 2; a adoção de APIs padronizadas (Gremlin, Cypher) e conectores *serverless* surge na

Onda 4; mas, em geral, falta avaliação prática de compatibilidade entre *frameworks*, ausência de *benchmarks* de interoperabilidade e raras iniciativas de *pipelines* híbridos (e.g., ML + grafos).

4.5.2 Resposta à Q2: Tendências Emergentes

4.5.2.1 Grafos Dinâmicos e Streaming

Cinco estudos apontam para processamento incremental em tempo real, foco em aplicações de IoT e redes sociais, e necessidade de reconstruir apenas deltas.

4.5.2.2 Particionamento e Balanceamento Adaptativos

Embora identificados como desafio majoritário (9/13), poucos trabalhos propõem soluções dinâmicas. Espera-se pesquisas em redistribuição *online* de partições, sem interrupção de execução.

4.5.2.3 Plataformas Serverless e FaaS Federado

A orquestração de funções sob demanda promete elasticidade extrema e *pay-per-use*, mas impõe desafios de *cold-starts*, latência imprevisível e falta de padrões federados.

4.5.2.4 Eficiência Energética e Sustentabilidade

Apesar de ganhos pontuais (TEPS/W, PUE), não há *benchmarks* consolidados. Tendência futura: adoção sistemática de métricas verdes (GreenGraph500), SLAs de consumo e políticas automatizadas de redução de energia.

4.5.2.5 Graph Neural Networks e Digital Twins

Incorporar GNNs a *pipelines* de grafos permitirá predição de padrões de uso e otimização proativa. *Digital twins* industriais de *data centers* facilitarão experimentação em escala realista antes da implantação.

4.5.2.6 Segurança e Privacidade

A ausência quase total de trabalhos nessa dimensão torna evidente a urgência de integrar *threat models*, autenticação distribuída e criptografia leve, especialmente em ambientes

multi-locatários e *serverless*.

4.5.3 Conexão entre Q1 e Q2

A evolução técnica demonstrou progressos claros em eficiência e escalabilidade, mas manteve segurança e interoperabilidade em segundo plano. As tendências mapeadas preveem cessões de *gaps* em *streaming* e elasticidade, mas exigem encarar lacunas em métricas verdes e proteção de dados. Assim, a agenda de pesquisa recomendada deve:

- Integrar segurança como *first-class citizen* em arquiteturas *vertex-*, *edge-* e *dataflow-centric*.
- Padronizar *benchmarks* que avaliem simultaneamente performance, consumo energético, latência de *cold-start* e interoperabilidade.
- Desenvolver mecanismos de particionamento e *auto-scaling* totalmente adaptativos, com monitoramento em tempo real.
- Explorar o uso de GNNs e *digital twins* para predição e otimização contínua de recursos.

Com isso, espera-se que esta revisão cumpra seu popósito de oferecer não apenas o panorama histórico das plataformas de grafos em larga escala, mas também um *roadmap* claro para responder aos desafios emergentes de eficiência, escalabilidade, segurança e interoperabilidade.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo central traçar a evolução das plataformas de processamento de grafos em larga escala, analisando como os desafios de eficiência, escalabilidade, segurança e interoperabilidade foram abordados ao longo da última década e quais são as tendências emergentes para o futuro. Através de uma revisão sistemática da literatura, foram selecionados e analisados 13 estudos primários que representam os avanços mais significativos na área.

Os resultados revelaram que a evolução dessas plataformas pode ser compreendida em quatro "ondas" temáticas. A primeira (2013-2015) consolidou o paradigma *vertex-centric* em arquiteturas híbridas CPU+GPU, alcançando ganhos expressivos de aceleração. A segunda onda (2014-2017) focou em otimizar a entrada e saída de dados com técnicas de *prefetch* em SSDs e particionamento *streaming*, mitigando gargalos de latência. A terceira (2017-2018) introduziu a observabilidade e a elasticidade, com *profiling fine-grained* que expôs ineficiências de I/O e mecanismos de *auto-scaling* que reduziram custos operacionais. Finalmente, a quarta onda (2021-2023) trouxe à tona a sustentabilidade, com métricas energéticas e o uso de *digital twins*, e a computação *serverless* com *workflows* em FaaS federado. Conclui-se que, enquanto a eficiência e a escalabilidade evoluíram notavelmente, com sistemas capazes de processar trilhões de arestas, a segurança e a interoperabilidade permaneceram como lacunas críticas, sendo raramente abordadas nos estudos.

As principais contribuições desta pesquisa residem na organização dessa trajetória evolutiva no modelo de ondas, que facilita a compreensão dos avanços e desafios; no mapeamento quantitativo que evidencia a predominância de desafios como balanceamento de carga e a escassez de pesquisas em segurança; e na síntese integrada que oferece um *roadmap* claro para a área. Contudo, o trabalho possui limitações, como o número restrito de estudos analisados e a ausência de replicação experimental, baseando-se nos resultados reportados pelos autores.

Com base nas lacunas identificadas, os trabalhos futuros devem direcionar esforços para áreas cruciais. Primeiramente, a segurança precisa ser tratada como um requisito fundamental, desenvolvendo *threat models* para ambientes distribuídos e avaliando o *overhead* de criptografia. Em segundo lugar, é essencial a criação de protocolos de particionamento dinâmico e adaptativo, capazes de reajustar cargas em tempo real. Adicionalmente, a comunidade necessita de *benchmarks* unificados e APIs padronizadas para permitir comparações justas entre plataformas (*cross-platform*). Por fim, tendências como a integração com FaaS federado e o

desenvolvimento de *digital twins* em *streaming*, combinando GNNs e hardware especializado como Processing-In-Memory (PIM)/Smart Network Interface Cards (SmartNICs), representam fronteiras promissoras para investigações futuras. O alinhamento entre essas frentes de pesquisa será determinante para que a próxima geração de plataformas de grafos alcance, de forma equilibrada, os objetivos de desempenho, escalabilidade, segurança e interoperabilidade.

REFERÊNCIAS

ANDREWS, G. R. **Foundations of Multithreaded, Parallel, and Distributed Programming**. 1. ed. Reading: Addison-Wesley, 2000.

CHARTRAND, G.; ZHANG, P. **A First Course in Graph Theory**. 1. ed. Mineola: Dover Publications, 2012.

CHING, A.; EDUNOV, S.; KABILJO, M.; LOGOTHETIS, D.; TAWARMALANI, S. One trillion edges: Graph processing at facebook-scale. In: **Proceedings of the VLDB Endowment**. [S.l.]: VLDB Endowment, 2015. v. 8, n. 12, p. 1804–1815. Disponível em: <https://dl.acm.org/doi/10.14778/2824032.2824077>. Acesso em: 16 abr. 2025.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 3. ed. Cambridge: MIT Press, 2009.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. **Distributed Systems: Concepts and Design**. 4. ed. Boston: Addison-Wesley, 2007.

DIESTEL, R. **Graph Theory**. 5. ed. Berlin: Springer, 2017.

GHARAIBEH, A.; COSTA, L. B.; RIPEANU, M. The energy case for graph processing on hybrid cpu and gpu systems. In: **IA3 '13: Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms**. Denver, CO: ACM, 2013. p. 1–8. Disponível em: <https://dl.acm.org/doi/10.1145/2535753.2535755>. Acesso em: 28 jul. 2025.

GONZALEZ, J. E.; LOW, Y.; GU, H.; BICKSON, D.; GUESTRIN, C. Powergraph: Distributed graph-parallel computation on natural graphs. In: **Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)**. Hollywood, CA: USENIX Association, 2012. Disponível em: <https://dl.acm.org/doi/10.5555/2387880.2387883>. Acesso em: 12 mar. 2025.

GONZALEZ, J. E.; XIN, R. S.; DAVE, A.; CRANKSHAW, D.; FRANKLIN, M. J.; STOICA, I. Graphx: Unifying data-parallel and graph-parallel analytics. In: **Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)**. Broomfield, CO: USENIX Association, 2014. p. 599–613. Disponível em: <https://arxiv.org/abs/1402.2394>. Acesso em: 17 jun. 2025.

GROSS, J. L.; YELLEN, J. **Handbook of Graph Theory**. Boca Raton: CRC Press, 2004.

HEIDARI, S.; BUYYA, R. A cost-efficient auto-scaling algorithm for large-scale graph processing in cloud environments with heterogeneous resources. **IEEE Transactions on Software Engineering**, v. 46, n. 11, p. 1194–1208, 2019. Disponível em: <https://ieeexplore.ieee.org/document/8798698>. Acesso em: 29 jul. 2025.

KITCHENHAM, B. A.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. Durham, UK, 2007. Disponível em: <https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf>. Acesso em: 2 abr. 2025.

KREPS, J.; NARKHEDE, N.; RAO, J. Kafka: A distributed messaging system for log processing. In: **Proceedings of the NetDB conference**. [S.l.: s.n.], 2011. Disponível em: <https://www.

semanticscholar.org/paper/Kafka-%3A-a-Distributed-Messaging-System-for-Log-Kreps/ea97f112c165e4da1062c30812a41afca4dab628>. Acesso em: 15 jan. 2025.

LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. **Communications of the ACM**, v. 21, n. 7, p. 558–565, 1978. Disponível em: <https://dl.acm.org/doi/10.1145/359545.359563>. Acesso em: 22 jan. 2025.

MALEWICZ, G.; AUSTERN, M. H.; BIK, A. J. C.; DEHNERT, J. C.; HORN, I.; LEISER, N.; CZAJKOWSKI, G. Pregel: A system for large-scale graph processing. In: **Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data**. Indianapolis, IN: ACM, 2010. Disponível em: <https://dl.acm.org/doi/10.1145/1807167.1807184>. Acesso em: 5 fev. 2025.

MCCOLL, R.; EDIGER, D.; POOVEY, J.; CAMPBELL, D.; BADER, D. A. A performance evaluation of open source graph databases. In: **PPAA '14: Proceedings of the first workshop on Parallel programming for analytics applications**. New Orleans, LA: ACM, 2014. p. 11–18. Disponível em: <https://dl.acm.org/doi/10.1145/2567634.2567638>. Acesso em: 16 abr. 2025.

MOLAN, J.; PETROVA, E.; RODRIGUEZ, C.; STEIN, M. The graph-massivizer approach toward a european sustainable data center digital twin. In: **Proceedings of the 2023 ACM/IEEE International Conference on High Performance Computing, Networking, Storage and Analysis (SC)**. Denver, CO: ACM/IEEE, 2023. p. 1–14. Disponível em: <https://ieeexplore.ieee.org/document/10196815>. Acesso em: 22 jul. 2025.

NGAI, E. C. H.; DING, C.; MA, J. Granula: Toward fine-grained performance analysis of large-scale graph processing platforms. In: **Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)**. Boston, MA: IEEE, 2017. p. 201–210. Disponível em: <https://dl.acm.org/doi/10.1145/3078447.3078455>. Acesso em: 28 jun. 2025.

NILAKANT, N.; SAAD, Y.; ZOU, Z.; BALAJI, P.; SANKARAN, R.; SCHWAN, K. Prefedge: Ssd prefetcher for large-scale graph traversal. In: **Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS)**. Phoenix, AZ: IEEE, 2014. p. 846–855. Disponível em: <https://dl.acm.org/doi/10.1145/2611354.2611365>. Acesso em: 23 jun. 2025.

RISTOV, S.; JOKANOVIC, M.; MAKINEN, J.; BRANDIC, I. Large-scale graph processing and simulation with serverless workflows in federated faas. In: **Proceedings of the 23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)**. Bangalore, India: IEEE/ACM, 2023. p. 127–136. Disponível em: <https://dl.acm.org/doi/10.1145/3578245.3585333>. Acesso em: 1 jul. 2025.

ROST, C.; GOMEZ, K.; TäSCHNER, M.; FRITZSCHE, P.; BäR, M.; TRAUB, J.; BREß, S.; SAAKE, G. Distributed temporal graph analytics with gradoop. **The VLDB Journal**, v. 31, n. 2, p. 375–401, 2021. Disponível em: <https://dl.acm.org/doi/10.1007/S00778-021-00667-4>. Acesso em: 31 jul. 2025.

ROY, A.; MIHAILOVIC, I.; ZWAENEPOEL, W. X-stream: Edge-centric graph processing using streaming partitions. In: **Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP)**. Farmington, PA: ACM, 2013. p. 472–488. Disponível em: <https://dl.acm.org/doi/10.1145/2517349.2522740>. Acesso em: 21 maio 2025.

TANENBAUM, A. S.; STEEN, M. van. **Distributed Systems: Principles and Paradigms**. 2. ed. Upper Saddle River: Pearson, 2007.

TSOURAKAKIS, C. E.; BONCHI, F.; GIONIS, A.; BONCHI, F. Fennel: Streaming graph partitioning for massive scale graphs. In: **Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY: ACM, 2014. p. 1–10. Disponível em: <https://dl.acm.org/doi/10.1145/2556195.2556213>. Acesso em: 7 jul. 2025.

VALIANT, L. G. A bridging model for parallel computation. **Communications of the ACM**, v. 33, n. 8, p. 103–111, 1990. Disponível em: <https://dl.acm.org/doi/10.1145/79173.79181>. Acesso em: 11 fev. 2025.

WANG, X.; CHEN, J.; ZHAO, L. Intelligent and independent processes for overcoming big graphs. In: **Proceedings of the 2017 IEEE International Conference on Distributed Computing Systems (ICDCS)**. Atlanta, GA: IEEE, 2017. p. 345–354. Disponível em: <https://dl.acm.org/doi/abs/10.1007/s11227-016-1834-4>. Acesso em: 18 jul. 2025.

WEST, D. B. **Introduction to Graph Theory**. 2. ed. Upper Saddle River: Prentice Hall, 2001.

XIN, R. S.; GONZALEZ, J. E.; FRANKLIN, M. J.; STOICA, I. Graphx: A resilient distributed graph system on spark. In: **First International Workshop on Graph Data Management Experiences and Systems (GRADES)**. New York, NY: ACM, 2013. Disponível em: <https://dl.acm.org/doi/10.1145/2484425.2484427>. Acesso em: 9 jun. 2025.

YAN, D.; CHENG, J.; LU, Y.; NG, W. Pregel algorithms for graph connectivity problems with performance guarantees. In: **Proceedings of the VLDB Endowment**. [S.l.]: VLDB Endowment, 2014. v. 7, n. 13, p. 1542–1553. Disponível em: <http://www.vldb.org/pvldb/vol7/p1542-yan.pdf>. Acesso em: 5 maio 2025.

ÖZSU, M. T.; VALDURIEZ, P. **Principles of Distributed Database Systems**. 3. ed. New York: Springer, 2010.

APÊNDICE A – PLANILHA DE EXTRAÇÃO DE DADOS

A planilha completa utilizada para a extração e consolidação dos dados dos 13 estudos primários selecionados nesta revisão sistemática da literatura está disponível publicamente no seguinte endereço eletrônico:

<https://docs.google.com/spreadsheets/d/
18ap3--6YiZqUq7Dr5CGxhMDvopXkai6dNUNo0rmK8AE/edit?usp=sharing>