Graph Similarity Computations on Large Graph Databases

Puya Memarzia, Virendra Bhavsar, and Suprio Ray

Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada

Introduction

Graph theory has been around since the 18th century. However, in recent times, the range of problems that require graph processing has grown rapidly. With the ever evolving needs of users and the rise of big data, understanding the relationships between interconnected data has grown in importance.

Graphs are ideal for a wide variety of applications, including natural language processing, geospatial analysis, medical records, social networks, ecommerce, and cybersecurity.

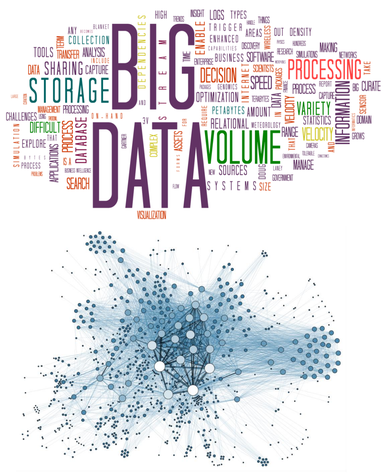

Figure 1. Graphs and Big data

Graph Similarity

Graph similarity is a way of measuring how similar two graphs are. This is done using intricate algorithms that analyze the graph's structure in addition to its properties. These algorithms generally return a number as a measure of the similarity between a pair of graphs. This similarity metric can then be used to group similar entities, such as webpages, products, malware, or medical patients.

Motivation

- Processing large numbers of complex graphs is a challenging task, and an ongoing problem.
- Data size and complexity continuously evolves over time. Graph databases are well-suited for this.
- Prior work designed for small-scale graph processing, and is not intuitive to use for large volumes of data.
- Performance and usability are major challenges.
- We need a comprehensive solution that can handle large-scale data, as well as data from different sources.

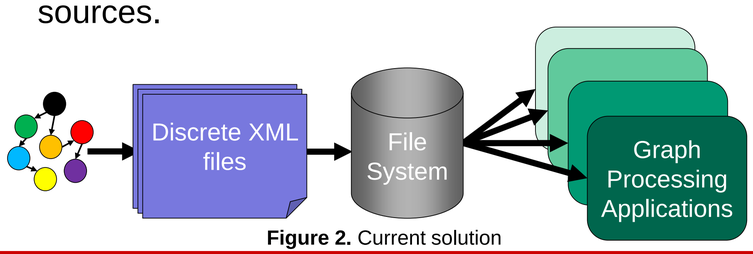

Figure 2. Current solution

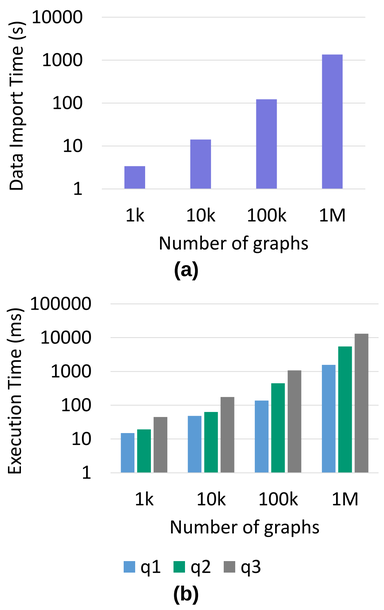

Figure 3. Neo4j microbenchmarks (a) importing data (b) basic pattern matching

Data Characteristics

- **Large**: Upwards of millions of discrete graphs
- **Complex**: Nodes are labeled and have additional properties. Edges are directed, and have weights and labels.
- **Dynamic**: Need to be able to add, remove, or modify nodes and relationships, at any time. Some graph types have additional rules that must be satisfied.
- Index-free adjacency: find adjacent nodes without the need for indexes or database scans. Performance doesn't degrade like relational databases.
- Labels are more important than properties, and are given preferential treatment in the database. Properties are stored separately, but cached.

Graph Processing Components

- **Graph database: Neo4j**
  - Meets our requirements to store and process the datasets.
  - Is an open source project, written in Java.
- **Random graph generator**
  - We are developing a flexible tool that will allows us to generate graphs or trees, with the characteristics that we need.
  - Supports weighted and labeled property graphs.
  - Outputs results in the GraphML format.
- **Parallel computing framework: Nvidia CUDA, JCuda**
  - GPGPU framework will be used to accelerate complex operations.
- **Data Loader**
  - Implement tool to filter and convert graph data to a format import tools can understand.
- **Graph Visualization Tools: Gephi, yEd**

Datasets

- Randomly generated graphs
- Electronic Medical Records (EMR)
- E-business
- E-learning data

Need uniform schema for each dataset, and data loaders for real world data.

Queries

- Nodes
- Relationships
- Paths
- Neighborhoods
- Distances

A combination of metrics

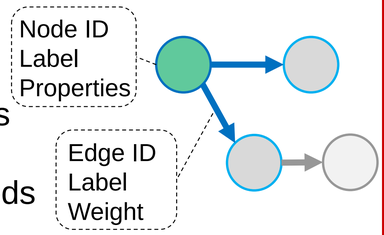

Figure 4. Graph components

Proposed Framework

- Powerful graph processing framework based around a graph database.
- Handle importing/exporting data in various formats, and visualizing the graphs.
- Some similarity algorithms will be implemented based on related work in [1][2][3].
- Cypher query language can satisfy some tasks, and computationally intensive workloads can be offloaded to GPU.

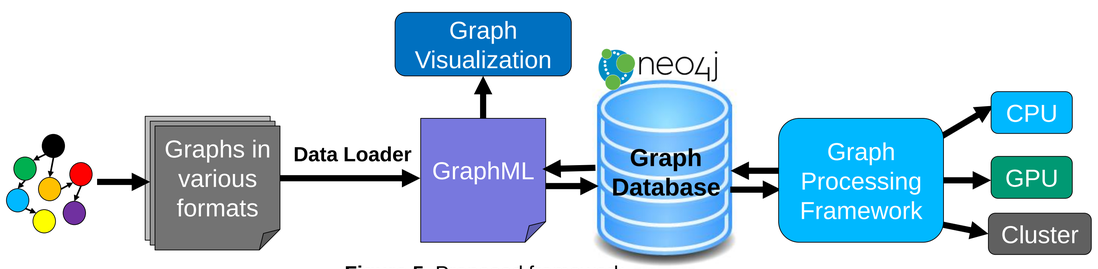

Figure 5. Proposed framework

Future Work

- Implement data loaders to import non-graph data.
- Improve performance by utilizing high performance computing (GPUs, clusters, cloud computing, etc.).
- Explore new techniques and data structures to reduce graph memory usage in parallel applications.
- Explore indexing techniques for graph databases.

References: [1] Bhavsar, Virenda, Harold Boley, and Lu Yang. "A weighted-tree similarity algorithm for multi-agent systems in e-business environments." (2004). [2] Kiani, Mahsa, Virendrakumar C. Bhavsar, and Harold Boley. "Combined Structure-Weight Graph Similarity and its Application in E-Health." CSWS. 2013. [3] Kiani, Mahsa, Virendrakumar C. Bhavsar, and Harold Boley. "Similarity of Attributed Generalized Tree Structures: A Comparative Study." Similarity Search and Applications. Springer International Publishing, 2015. 150-161.