Topic List

Topic： Graph Analytics Systems

Seed Papers

- **Parallelizing Sequential Graph Computations** *ACM Transactions on Database Systems (TODS) 43(4): 18:1-18:39* (*invited*). Wenfei Fan, Jingbo Xu , Wenyuan Yu , Jingren Zhou, Xiaojian Luo, Ping Lu, Qiang Yin, Yang Cao, and Ruiqi Xu
- **Adaptive Asynchronous Parallelization of Graph Algorithms** *ACM Transactions on Database Systems (TODS) 45(2): 6:1-6:45* (*invited*), 2020. Wenfei Fan, Ping Lu, Wenyuan Yu, Jingbo Xu, Qiang Yin, Xiaojian Luo, Jingren Zhou, and Ruochun Jin.
- **Application Driven Graph Partitioning** *ACM SIGMOD Conference on Management of Data (SIGMOD),* 2020. Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Xiaojian Luo, Ruiqi Xu, Qiang Yin, Wenyuan Yu, and Jingren Zhou
- **MiniGraph: Querying Big Graphs with a Single Machine** *The 49th International Conference on Very Large Data Bases (VLDB)*, 2023. Xiaoke Zhu, Yang Liu, Shuhao Liu, and Wenfei Fan
- Graph Computation with Adaptive Granularity. 2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024. Xu, Ruiqi, Yue Wang, and Xiaokui Xiao.

Project: Graph Computation under Heterogeneous Hardware Environment

key ideas

1. graph processing systems usually assume the underlying cluster consists of machines with same CPUs, memory sizes and peer-to-peer network bandwidth/latency. However, such is usually not the case. How to efficiently deal with heterogeneous hardware environment is rarely studied. One may consider the following aspects:
    - how to partition the graph tailored for a heterogeneous cluster?
        - specifically, one can start with ADP (paper: **Application Driven Graph Partitioning**), which evenly distributes the computational and communicational workload of a given graph application across a homogeneous cluster. One need to figure out how to skewly distributes such workloads catering for the different machines on a heterogeneous cluster. The factors that one needs to consider may include: 1) the computational power, 2) the network bandwidth and latency and 3) the memory capacity of the machines in the cluster. To simplify the problem, one may assume two of these three factors are the same for all the machines in the cluster, and study how the difference of the other factor may affect the distributed computation computation
    - similarly, one can consider how to adjust the granularity of parallel computation for a heterogeneous cluster.
        - this extends the paper **Graph Computation with Adaptive Granularity**. Specifically, how to adjust the granularity on a heterogeneous cluster? Consider the justifications that one need to make to the previous system Argan.
2. when there is not enough memory space, it is necessary to offload data to disks (hard drive or SSDs). Such computation, also known as out-of-core computation, needs to focus on how to fetch (resp. flush back) data to (resp. from) disks efficiently, in addition to carrying out the top-level graph applications.
    - on the one hand, paper **MiniGraph: Querying Big Graphs with a Single Machine** develops the first out-of-core system under graph-centric paradigm. It directly migrates the distributed programming model of PIE (see paper: **Parallelizing Sequential Graph Computations**) . However, there are still much room for improvements, for instance:
        - an asynchronous model would be much preferable for out-of-core computation, in contrast to the synchronous execution model adapted by MiniGraph
        - how to load and write-back data from disks? are pre-computed partitions really necessary? under the settings of a single machine, the whole graph is available on disk, so it should be possible to compute without pre-partition the graph?
    - on the other hand, many vertex-centric based out-of-core graph systems have been developed (see the papers provided), and they all comes with their own I/O optimizations. Think about which of these optimizations can be adapted under a graph-centric scenario.

Other Materials

Julian Shuan's List

Topic： Parallelization and Incrementalization of Graph Neural Networks

- Study how to parallelize graph neural networks for better efficiency and scalability:
    - One project is to study how to integrate different parallel execution models into a single, adaptive and uniform one, with better efficiency and scalability while relieving users from the work of finding the optimal model to use. One can think of the previous work including *Adaptive Asynchronous Parallelization of Graph Algorithms* and *Graph Computation with Adaptive Granularity* (see seed papers from 1st topic) and try to migrate them from the context of conventional graph analytics to that of graph learning.
        - Seed paper:
            - Shao Y, Li H, Gu X, et al. Distributed graph neural network training: A survey[J]. ACM Computing Surveys, 2024, 56(8): 1-39.
            - Besta M, Hoefler T. Parallel and distributed graph neural networks: An in-depth concurrency analysis[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024.
    - Another project is to study how to boost the accuracy of incremental / temporal GNN training while ensuring its efficiency. For dynamic / temporal graphs, maintaining consistency over time is challenging because the model needs to ensure that updates do not invalidate previously learned patterns or introduce inconsistencies. Typical challenges in terms of consistency include: concept drift, class imblance, bias towards recent data, catestrophic forgetting and etc.. It is well ackowledged that there are gaps in terms of accuracy between graph learning over dynamic / temporal graphs and those over static ones. One possible solution is to migrate what we have achieved on incrementalization of graph analytics to graph learning.

The key is to establish theoretical results on both the quality and efficiency of the results: for quality, can we ensure the incremental computation is as good as retrain the model from scratch? For efficiency, can we ensure incremental boundedness or relative boundedness?

- Seed paper:
  - Longa A, Lachi V, Santin G, et al. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities[J]. arXiv preprint arXiv:2302.01018, 2023.
  - Wenfei Fan, Muyang Liu, Chao Tian, Ruiqi Xu, and Jingren Zhou, Incrementalization of Graph Partitioning Algorithms The 46th International Conference on Very Large Data Bases (VLDB), 2020.
  - Wenfei Fan, Chao Tian, Qiang Yin, Ruiqi Xu, Wenyuan Yu, and Jingren Zhou., Incrementalizing Graph Algorithms, ACM SIGMOD Conference on Management of Data (SIGMOD), 2021.

Topic： Semantic Enrichment of Texts-based Generative Models with Graphs

- In NLP tasks, such as question answering or summarization, models often rely on dense vector representations of text, such as those provided by transformers like BERT. While these representations capture a lot of contextual information, they may lack explicit structural information about the relationships between words or concepts within the text.
  - One emerging solution is GraphRAG (Graph Representation Augmented with Graphs). It addresses the limitation by incorporating a graph-based representation of the text alongside the traditional token-based representations. The graph captures relationships between entities, events, or other salient features in the text, allowing the model to reason about the content more effectively.

    - Alternatively, one can integrate LLMs with knowledge graphs, which falls under knowledge representation, knowledge reasoning, and enhanced NLP. This approach aims to combine the strengths of large language models with structured knowledge to provide more accurate and contextually rich responses.
- At the system level, study
    - how to boost the efficiency and accuracy of these models, including the training process, the inference phase and how the integration with graph-based information is carried out
    - how to simplify the programming model by generalizing the models across various different applications
- At the application level, one possible project is on early drug detection (in collaboration with 中石大华东):
    - Background:
        - Computer-Aided Drug Design (CADD) was proposed in the 1960s, relying on disciplines such as computational chemistry, computer science, and biology. It involves computational simulations and predictions of the binding processes between target proteins and ligand drugs, evaluating the relationships between drug molecular structures and their biological activity, toxicity, and metabolism, aiding in the discovery and optimization of drug molecules. The development and application of high-throughput technologies have generated abundant data on drugs, diseases, genes, and proteins, making it feasible to conduct artificial intelligence-driven drug discovery. In recent years, artificial intelligence systems, represented by Google's AlphaFold, have achieved significant breakthroughs in the life sciences, greatly promoting the application of AI in drug research and development. Key AI technologies such as Natural Language Processing (NLP), Machine Learning (ML), Deep Learning (DL), and Knowledge Graphs (KG) have been widely applied in various stages of drug discovery, including target identification, hit compound screening, de novo drug design, drug repurposing, prediction of drug properties, prediction of adverse drug reactions, and interpretable models in drug design.
    - Task:
        - To build a composite knowledge graph centered around disease metabolic pathways that integrates drugs, diseases, and targets. Based on this knowledge graph, screen candidate drugs for diseases, predict drug properties, discover potential targets and pathways, and generate artificially designed new drugs. This approach aims to reduce costs and

improve efficiency, accelerating the process of drug discovery.

- Seed Papers:
  - Zheng S, Rao J, Song Y, et al. PharmKG: a dedicated knowledge graph benchmark for bomedical data mining[J]. Briefings in bioinformatics, 2021, 22(4): bbaa344.
  - Zhu C, Yang Z, Xia X, et al. Multimodal reasoning based on knowledge graph embedding for specific diseases[J]. Bioinformatics, 2022, 38(8): 2235-2245.