Bancos de Dados em Grafos

Gustavo Estrela de Matos
Instituto de Matemática e Estatísticas
Universidade de São Paulo
São Paulo, Brasil
gestrela@ime.usp.br

Hector Montenegro Terceros
Instituto de Matemática e Estatísticas
Universidade de São Paulo
São Paulo, Brasil
hector@ime.usp.br

***Resumo*—Vamos fazer este por último...**

***Index Terms*—component, formatting, style, styling, insert Isso aqui eu não entendi direito, não sei como fazer aqui.**

I. Introdução

Grafos são objetos matemáticos que podem ser usados como estruturas de dados em diversas aplicações computacionais. Sistemas de Bancos de Dados baseados em grafos são considerados bancos de dados NoSQL [1] e podem ser aplicados em diversos contextos, como o de aplicativos da web; áreas industriais de transportes, telecomunicação e comércio [2]; e também em áreas de pesquisa, como em bioquímica [4], biologia molecular [5] e web semântica [6].

Um grafo pode ser definido como uma dupla $G = (V, E)$, em que V é um conjunto de vértices (ou nós) e E é um conjunto de arcos, que conectam dois nós. Podemos representar um arco $e \in E$, por uma dupla $e = (v_i, v_j)$ com $v_i \in V$ e $v_j \in V$. No contexto de bancos de dados de grafos, usualmente um nó representa uma entidade modelada, e arcos representam relacionamentos entre essas entidades. É comum que os nós recebam rótulos que estão associados ao tipo de entidade modelada, e também um conjunto de propriedades em formato chave-valor, capaz de armazenar atributos da entidade. Além disso, os arcos do modelo também podem receber rótulos que identificam o tipo de relacionamento que é modelado, e um conjunto de propriedades que representam atributos do relacionamento.

Bancos de dados baseados em grafos costumam ser aplicados em contextos em que os dados de interesse possuem relacionamentos complexos ou simplesmente quando boa parte da informação está contida nos relacionamentos. Nestes casos, um banco de dados relacional pode ser inadequado ou ineficiente. Considere, por exemplo, a relação `FABRICA (id_fabricante, id_produto)`. Em uma consulta em que se deseja saber as informações dos produtos fabricados por uma fábrica, no modelo relacional, é necessário fazer uma junção com a relação que armazena os dados dos produtos, enquanto no modelo baseado em grafos, basta percorrer os arcos do relacionamento `FABRICA` que estão ligados ao nó que representa a fábrica de interesse. Além de ser mais eficiente para algumas consultas, sistemas baseados em grafos podem ter consultas mais expressivas, capazes de representar relacionamentos complexos entre os dados [3].

Assim como outros bancos de dados NoSQL, os bancos de dados em grafos são muito utilizados em aplicações que precisam armazenar um grande volume de dados. Para atender a este requisito, alguns bancos de dados baseados em grafos permitem armazenamento distribuído. Este tipo de solução precisa implementar particionamento de dados. Além disso, um sistema distribuído deve providenciar maneiras de responder a consultas que acessam informações de vértices alocados em diferentes máquinas de uma rede.

Neste artigo, nosso principal objetivo é apresentar os conceitos fundamentais e as principais soluções para implementar bancos de dados baseados em grafos distribuídos, e também sistemas para o processamento distribuído de grafos. Ao longo deste trabalho, vamos... informações a serem adicionadas no futuro, com o que vamos colocar de fato no artigo.

II. Metodologia

- Como escolhemos os artigos que lemos para criar esse artigo? - Qual tipo de grafos estamos focados em tratar? (Os que a gente achou... acho que é maioria usado pra rede social(?))

III. Conceitos Fundamentais

Nesta seção apresentamos os conceitos fundamentais para entender o funcionamento de bancos de dados e ferramentas de processamento de dados baseados em grafos, mais especificamente, em contextos distribuídos.

A. Graph DBMS versus Graph Analytics Systems

B. Cortes de grafos

O conceito de corte em grafo é importante para formalização e análise do particionamento de um grafo em um contexto distribuído. Para se trabalhar com grafos de maneira distribuída, é necessário criar um particionamento do grafo, para que cada parte seja alocada em um nó da rede de computadores usada. Um particionamento precisa ser feito de maneira que o processamento do grafo seja balanceado, ou seja, as máquinas devem ter números de acessos similares; além disso, é importante fazer com que os processamentos do grafo sejam feitos com o menor número de nós de computação possível, pois o processamento que envolve mais de um nó de computação necessita de maior comunicação pela rede, aumentando a latência da operação. Para analisar estes dois aspectos com maior formalismo, definimos o conceito de corte.

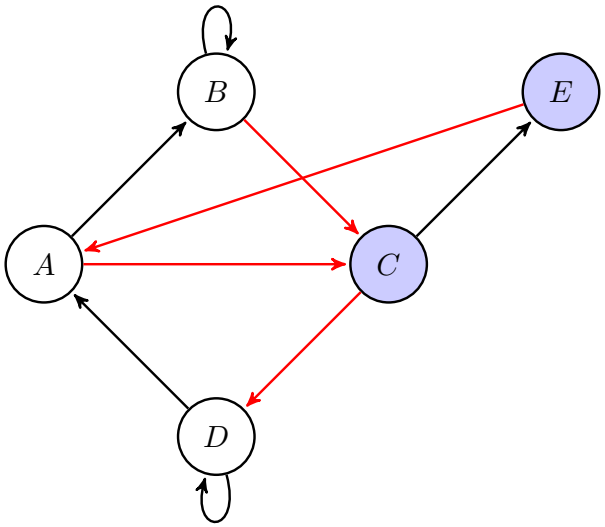

Figura 1. Grafo com exemplo de corte. O corte apresentado é definido pelos conjuntos de vertices E, C, em azul, e A, B, D, em branco. Os arcos de cor vermelha são os arcos deste corte. No total, são 4 arcos vermelhos neste corte, ou seja, o tamanho deste corte é 4.

Um corte é um particionamento em dois conjuntos disjuntos de vértices de um grafo. Seja $G = (V, E)$ um grafo, então um conjunto $S \subseteq V$ de vértices induz o corte $(S, V \setminus S)$. Um arco $e \in E$ é chamada de arco do corte se ele atravessa as partes, ou seja, se $e = (u, v)$ com $u \in S$ e $v \notin S$, ou $u \notin S$ e $v \in S$. Chamamos de tamanho do corte o número de arcos que são arcos do corte. A figura 1 apresenta um exemplo de corte em um grafo.

Note que qualquer particionamento (não apenas bipartições) do conjunto de vértices de um grafo pode ser realizado após recursivas aplicações de cortes. Portanto, é possível construir e analisar cortes olhando apenas para bipartições do grafo original. Perceba também que ao fazer um particionamento, podemos analisar o balanceamento de processamento dos nós de computação de acordo com o número de partes (e vértices) que são alocadas aos nós de computação. Além disso, é possível analisar a quantidade de comunicação entre nós de acordo com a quantidade de arcos que conectam diferentes partes, que podem estar alocadas em dois nós de computação diferentes.

Se formos falar de cortes baseados em arcos, é melhor adicionar mais um pedaço de texto explicando esse tipo de corte.

IV. Particionamento de grafos

O particionamento de um grafo é o processo que permite dividir um grafo em diferentes partes, que podem ser alocadas em diferentes máquinas, permitindo o processamento e armazenamento distribuído do grafo. Nesta seção, apresentaremos três diferentes abordagens de particionamento. A primeira, baseada no algoritmo METIS [7], é uma heurística que tenta minimizar a quantidade de arcos que conectam partes diferentes; a segunda faz um particionamento aleatório; e a terceira particiona os vértices de acordo com o valor de alguns atributos destes vértices.

A. O algoritmo METIS

O algoritmo METIS tem como objetivo produzir um particionamento com k partes que minimiza a quantidade de arcos atravessando partes. O problema de encontrar tal partição é chamado $k-$particionamento de um grafo, e vamos definilo assim: dado um grafo $G = (V, E)$, com $|V| = n$, particione V em k conjuntos $V_1, V_2, \ldots, V_k$ de maneira que $V_i \cap V_j = \emptyset$ para $i \neq j$, $\bigcup_{i=1}^{k} V_i = V$, e $|V_i| = n/k$ (sem perda de generalidade, se n não é divisível por k, então adicione nós "fantasmas"até que $k|n$). O $k-$particionamento também pode ser generalizado ao considerar pesos para arcos, adicionando uma informação de importância ou relevância para as conexões, mas por facilidade vamos considerar apenas o caso em que todos os peso dos arcos é igual, consistente com a definição que apresentamos.

O $k-$particionamento possui importância em aplicações de computação paralela e distribuída, e foi provado ser NP-completo [8], ou seja, o problema se torna intratável rapidamente com o aumento da instância. Por isso, o algoritmo METIS é uma heurística, ou seja, a solução produzida não é ótima em geral.

B. Particionamentos aleatórios

C. Particionamentos com semântica

V. Sistemas Gerenciadores de Bancos de Dados em Grafos

VI. Graph Analytics Systems

A. Pregel

Referências

[1] "NoSQL Databases", http://nosql-database.org/.
[2] "Neo4j Customers", https://neo4j.com/customers/.
[3] M. Hunger, R. Boyd. "RDBMS & Graphs: SQL vs. Cypher Query Languages"Neo4j Blog. Mar. 2016. https://neo4j.com/blog/sql-vs-cypher-query-languages/
[4] N. Swainston et al., "biochem4j: Integrated and extensible biochemical knowledge through graph databases," PLOS ONE, vol. 12, no. 7, p. e0179130, Jul. 2017.
[5] F. Olken, "Graph Data Management for Molecular Biology," OMICS: A Journal of Integrative Biology, vol. 7, no. 1, pp. 75–78, Jan. 2003.
[6] B. McBride, "Jena: a semantic Web toolkit," IEEE Internet Computing, vol. 6, no. 6, pp. 55–59, Nov. 2002.
[7] Karypis, G., & Kumar, V. (1998). "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". SIAM Journal on Scientific Computing, 20(1), 359–392. https://doi.org/10.1137/s1064827595287997
[8] Andreev, K., & Racke, H. (2006). "Balanced Graph Partitioning". Theory of Computing Systems, 39(6), 929–939. https://doi.org/10.1007/s00224-006-1350-7