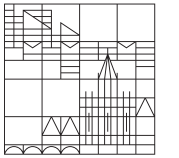
Universität Konstanz



Locality Optimization for traversal-based Queries on Graph Databases

Fabian Klopfer

Database and Information Systems Group
University of Konstanz, 17.12.2020

Motivation

- Neo4J: nodes and relationship stored in insertion order ⇒ Random Access to Disk
- Idea: Store nodes of same community close to each other

  Then verify if:
  - Nodes used frequently together are closer/on the same page
    ⇒ less pages must be accessed
  - The contents of neighbouring pages are frequently used together
    ⇒ pages can be loaded sequentially
  - Overall IO is thus reduced

Related Work I

- Steinhaus, Olteanu: G-Store [1]
  Storage Manager for Graph DB using modified version of Multi-level partitioning [2]
- Gedik with Yaşar and Bordawekar
  - Diffusion sets and coarse partitioning [3], [4]
  - Conductance and Cohesiveness metrics, adaptive [5]
- Pacher, Specht: N-Body Simulation

Most of them quantify query runtime or distance

Current state

- Documented Neo4J record formats in detail
- Implementing: Graph DB Storage Layer with Record & File layouts mimicing Neo4J layouts
- Implement record rearrangement strategy optimizing locality
- Compare random placement with to be derived method for the “queries”: BFS, Dijkstra, A*

The Cost Model

- Runtime-based measurement of own implementation hardly comparable to Neo4J:
  - used language: Java vs. C
  - Buffer Manager: Record layout transforming page cache vs. simple buffer manager
  - Query language: Cypher vs. direct implementations of BFS, Dijkstra, A*

How to quantify IO $\Rightarrow$ What is the cost model?

Sufficient to implement GetNodes and Expand and use Hölsch, Grossniklaus [6]?

Bibliography

[1] R. Steinhaus, D. Olteanu, and T. Furche, “G-Store: a storage manager for graph data”, Ph.D. dissertation, Citeseer, 2010.

[2] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs”, SIAM Journal on scientific Computing, vol. 20, no. 1, pp. 359–392, 1998.

[3] A. Yaşar, “Scalable layout of large graphs on disk”, Ph.D. dissertation, bilkent university, 2015.

[4] A. Yaşar, B. Gedik, and H. Ferhatosmanoğlu, “Distributed block formation and layout for disk-based management of large-scale graphs”, Distributed and Parallel Databases, vol. 35, no. 1, pp. 23–53, 2017.

[5] B. Gedik and R. Bordawekar, “Disk-based management of interaction graphs”, IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 11, pp. 2689–2702, 2014.

[6] J. Hölsch and M. Grossniklaus, “An algebra and equivalences to transform graph patterns in Neo4j”, in EDBT/ICDT 2016 Workshops: EDBT Workshop on Querying Graph Structured Data (GraphQ), 2016.