Parallel Traversal of Graphs Stored in RDBMSs

Christine F. Reilly
Skidmore College
Saratoga Springs, New York, USA
creilly@skidmore.edu

1. INTRODUCTION

This abstract proposes an approach for parallel traversal of graph structured data that is stored in a relational database management system (RDBMS). For applications with a data size and workload that is suitable for a single server system, an RDBMS may be a better option than a specialized graph system [3]. An open challenge with using RDBMSs for graph data is how to support the recursive operations that are commonly utilized for traversing graphs.

Prior work has explored graph queries in RDMBSs. One proposal is to extend relational algebra with matrix operations that support graph queries [5]. While the performance experiments show that these new relational algebra operations allow graph queries to be processed in a reasonable amount of time, the authors identify the use of parallel processing for enhancing performance as an area for future work. Another approach creates a system where a column-store RDBMS and a graph processing system share a storage engine and an enhanced query optimizer includes cost models for graph traversal query algorithms [4]. The approach proposed in this abstract differs from previous work in two main ways. First, it operates as an application on top of the RDBMS, allowing an organization to use their choice of RDMBS for the storage layer. Second, it employs parallel programming techniques to fully utilize the multiple and multithreaded processors available in modern computers.

2. GRAPH SCHEMA

In order to support a wide variety of graph applications, we utilize a simple schema similar to that used by Facebook's Linkbench [1]. There are two database tables: one for the nodes of the graph and one for the edges of the graph. Each node and edge has an integer type code and arbitrary properties. Instead of having an attribute for each of the properties in a property graph, this schema has a single property attribute that is a binary large object. It is the responsibility of the application that uses this database to translate node and edge logical types that are utilized by the application into the integers that are stored in the database, and to interpret the binary large object property. For example, based on the node type the application would select the corresponding format for property binary data. The specific property data is extracted from the binary large object based on the selected format using similar methods as are used by RDBMSs for storing record data in binary form, such as fixed length records or variable length records with data pointers.

3. PARALLEL GRAPH TRAVERSAL

We propose a new approach for traversing graph data that is stored in an RDBMS: move the recursion into a graph search application that is placed between the user application and the RDBMS. The graph search application uses parallel programming techniques for improved performance and uses simple queries to obtain data from the database. For example, an existing algorithm for parallel breadth-first search [2] is modified to obtain the list of neighboring nodes through a simple (non-recursive) database query. The RDMBS and the search application can be run on different servers in order to expand the available computing resources.

Experiments will be conducted to compare this approach for parallel graph traversal with the use of recursive queries in an RDMBS and with specialized graph systems such as Neo4j. Datasets that represent a variety of applications will be used during testing to explore how the performance of this approach for parallel graph traversal varies for different types of graph data.

4. REFERENCES

[1] T.G. Armstrong et al. LinkBench: A database benchmark based on the Facebook social graph. In *SIGMOD*. ACM, 2013.
[2] A. Buluç and K. Madduri. Parallel breadth-first search on distributed memory systems. In *Proc. of 2011 Intl. Conf. for High Performance Computing, Networking, Storage and Analysis*, SC '11. ACM, 2011.
[3] A. Pacaci et al. Do we need specialized graph databases?: Benchmarking real-time social networking applications. In *Proc. of the Fifth Intl. Workshop on Graph Data-management Experiences & Systems*, GRADES'17. ACM, 2017.
[4] M. Paradies et al. GRAPHITE: An extensible graph traversal framework for relational database management systems. In *SSDBM*. ACM, 2015.
[5] K. Zhao and J.X. Yu. All-in-one: Graph processing in RDBMSs revisited. In *SIGMOD*. ACM, 2017.