System Support for Managing Large Graphs in the Cloud

Sameh Elnikety and Yuxiong He
Microsoft Research

1 Motivation

Large graphs are at the heart of online social networks and many other applications including routing in road networks and online collaboration systems. In this paper, we argue that a novel distributed infrastructure is needed to manage and query large graphs to meet the demands of these applications.

In a public social network, a node represents an entity such as a person, event, or photo. An edge represents a binary relationship between two nodes indicating for example friendship, participation at an event or appearance in a photo. Both nodes and edges may have a set of attributes because they model real world entities and interactions. The resulting social graph is challenging to manage: It is too large to manage on a single server, there are frequent updates and users want to pose ad-hoc queries. For example, a user may ask "which photos include me and my friends X and Y", "how I am connected to person Z", or "who among my friends is attending this event".

Private social networks pose similar challenges. For example, Codebook [BPZ10] is a social network of software developers and their software artifacts within an enterprise. Codebook manages a large graph modeling software developers with their organizational hierarchy, source code (including files and functions and their revisions), bug reports, and design documents. Such data are gathered from source code repositories and the employee database. Codebook allows engineers to ask "who resolved this bug", and "who built this feature", and "who will be impacted if I change this source code function".

2 Limitations of Current Graph Systems

In contrast to batch graph processing systems, which have important applications such as social network analytics and webpage raking, we focus here on online systems that answer interactive queries within a few hundred milliseconds.

Current graph management systems offer limited functionality to answer the queries mentioned in Section 1, which include both reachability queries and graph pattern matching. Those types of queries are not suited for relational engines [ADJ87] for several reasons: (1) Some queries are recursive (e.g., reachability queries), (2) nodes and edges are accessed in pattern not suitable for disk-based data structures, and (3) pattern queries require an excessive number of join operations with large intermediate results. This motivates graph systems to build their specialized graph engines.

Existing graph systems lack declarative query languages. For example, neo4j, which is among the most popular centralized graph systems, provides a navigational interface, where programmers need to write programs, rather than declarative queries. Distributed graph engines such as Google Pregel [MAB+10] and Microsoft Surfer [CWHY10] accept programs that execute at graph nodes and send messages across graph edges, focusing more on batch processing.

3 Cloud Infrastructure

Cloud computing offers two opportunities. First, the availability of a large number of servers allows using large main memories to host the topology of large graphs, enabling online query processing. This is important because

processing queries over disk-based graphs is too slow for interactive queries. Second, multiple sources of information can be aggregated into a multi-graph, enabling richer queries. A user query can access data from several social networks: Facebook and LinkedIn have portions of the user social network.

4 Where We Stand

We outline several problems which we are investigating and invite our colleagues to reshape and solve them.

4.1 Graph Query Languages

The interface of graph system should be a declarative query language to allow users to write queries rather than navigational programs. We find that the majority of graph queries can be expressed as regular expressions or graph patterns. Both types of queries can be expressed declaratively, allowing execution engines to optimize their processing.

4.2 Execution Engines

Graphs are processed on a cluster of servers. Hybrid execution engines, which use both a graph engine and a relational engine, offer important advantages. A graph engine maintains the graph topology in main memory to answer reachability queries, and a relational engine manages node and edge attributes to retrieve predicated graph elements.

4.3 Isolation

Graph operations are dominated by traversals with more reads than writes. Multi-version concurrency control models offer clear correctness semantics as well as low overhead for read dominated workloads. In particular, generalized snapshot isolation [EPZ05] extends conventional snapshot isolation to distributed systems in a manner that allows graph traversals to neither block or be blocked by updates while providing serializability [BHEF11].

4.3 Query Optimization

With a declarative query language, a query optimizer can generate efficient execution plans customized for the managed graph instance to visit fewer nodes. Mid-query re-optimization [KD98] and budget-based techniques [BPS11] seem effective for traversing scale-free graphs [BB03].

Graph indexes and materialized views are active areas of research but they are developed in isolation and have not been integrated into graph systems. Current optimizers do not fully exploit them.

References

[ADJ87] R. Agrawal, S. Dar, H.V. Jagadish. "Direct Transitive Closure Algorithms: Design and Performance Evaluation." VLDB 1987.

[BB03] Albert-László Barabási, Eric Bonabeau. "Scale-Free Networks." Scientific American May 2003.

[BHEF11] Mihaela Bornea, Orion Hodson, Sameh Elnikety, Alan Fekete. "One-Copy Serializability with Snapshot Isolation under the Hood." ICDE 2011.

[BPS11] Matthias Bröcheler, Andrea Pugliese, V. S. Subrahmanian. "A budget-based algorithm for efficient subgraph matching on Huge Networks". GDM 2011

[BPZ10] Andrew Begel, Khoo Yit Phang, Thomas Zimmermann. “Codebook: Discovering and Exploiting Relationships in Software Repositories.” ICSE 2010.

[CWHY10] Rishan Chen, Xuetian Weng, Bingsheng He, Mao Yang. “Large Graph Processing in the Cloud.” SIGMOD 2010.

[EPZ05] Sameh Elnikety, Fernando Pedone, Willy Zwaenepoel. “Database Replication Using Generalized Snapshot Isolation.” SRDS 2005.

[KD98] Navin Kabra, David DeWitt. "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans." SIGMOD 1998.

[MAB+10] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski. “Pregel: a System for Large-Scale Graph Processing.” SIGMOD 2010.

[SEHK12] Mohamed Sarwat, Sameh Elnikety, Yuxiong He, Gabriel Kliot. “Horton: Online Query Execution Engine for Large Distributed Graphs (Demo)”. ICDE 2012.