Hybrid Transaction and Analytics Processing Systems for Graph Structured Data

Puneet Mehrotra
University of British Columbia

The Problem: A lot of information is naturally expressible as graphs - from proteomics to business applications, large graphs are ubiquitous and there is a great demand for scaling graph processing to trillion-edge datasets. We have come to understand graph processing to require the use of specialized storage and processing engines that work best when distributed across many compute nodes to gain maximum parallelism. This is contrary to several works that obviate the need for specialized graph processing systems and challenge the notion that distributing the analytics tasks always performs better. An unfortunate side effect of this design is the need to have costly preprocessing, partitioning, and ingestion steps that can quickly come to dominate the overall runtime of the system. The preprocessing step can be amortized if the data doesn't evolve, but the ingestion (and transformation to in-memory data structures) must be repeated for every run.

Creating a distributed graph processing system comes with an implicit understanding that the parallelism needed to scale computation is not available on a single compute node; this is not quite true since there is a lot of untapped parallelism available on commodity servers with high core counts and ample amounts of RAM. Besides, the average graphs are not large enough to warrant distribution. Additional care must be taken to scale these systems to work for an evolving graph and under conditions where real-time analytics must be performed.

At first glance, databases appear to be the obvious solution to store and process graphs, especially considering that most data to be analyzed is often already present in some relational database. Databases such as Postgres and MySQL have mature storage backends that support high insertion rates and provide strong consistency guarantees, but they remain largely difficult and non-intuitive to use for graph processing. Graph Databases store data natively as a graph, but are usually unable to scale to graphs with billions of nodes and suffer from poor ingestion rates. Besides, there are very few real-life use-cases that exclusively rely on graph queries, and therefore graph databases are usually used alongside more traditional databases.

Utopia: This tension between the graph data storage and processing is one that must be addressed and reconciled. An ideal midpoint in the design space would be a hybrid transaction and analytics system that stores the graph structured data in an on-disk layout that is close to the in-memory structure needed for the analytics tasks that follow. If the data is stored using an engine that delivers robust performance on reads while also supporting high insertion rates, we can use the same backend as the transactional store as well as run analytics workloads. Once we optimize for strong single node performance, we can scale out the computation to other nodes [1].

Getting there: Our goal is to use performance driven design to produce a single-node graph storage and processing system that can be configured to efficiently handle variable workloads and from which a distributed graph processing system can be developed.

[1] "You can have a second computer once you've shown you know how to use the first one." – Paul Barham