Scalable Isolation Guarantees for Distributed Graph Databases

Paul Ezhilchelvan, Jack Waudby

February 7, 2019

Abstract

A distributed graph database partitions graph data across multiple machines in a cluster. Partitioning a graph is non-trivial and much research has been dedicated to finding the optimal solution. Regardless of the chosen partitioning some edges will be distributed between servers: with the nodes an edge connects residing on separate machines. It is a challenge to maintain the correctness of distributed edges as records are being concurrently modified.

Several contemporary distributed graph databases (e.g. JanusGraph) use existing NoSQL databases (e.g. Apache Cassandra) as a storage backend, adapting them with an API in order to handle a graph data model. This approach is attractive as the underlying store offers high scalability. However, it only offers weak isolation guarantees which has serious ramifications for the integrity of graph database systems. It has been recently established that weak isolation can lead to corruption of distributed edges and then to irreversible database corruption to a significant degree, in intervals that are worryingly small compared to database lifetime.

Maintaining correctness of edge information at nodes residing on separate partitions requires the introduction of ACID transactions into the graph database. A straightforward approach may use existing solutions: two-phase commit for atomicity and durability, pessimistic two-phase locking for concurrency control, and Paxos for data replication. However, this architecture would be critically harmful for the performance of graph database workloads.

The focus of this research is to develop an efficient distributed transactions protocol tailored for graph database workloads, queries tend to involve multiple reads before a write, with write-write conflicts being rare. Early work has taken motivation from the Linear Transactions protocol, which imposes an order only between transactions that conflict in their write sets, aborting all other conflict types. Due to the nature of graph queries outlined above, this approach may introduce an intolerably high number of aborts. Focus has been on developing a protocol suite that allows for efficient and safe ordering amidst various forms of conflicts. We will present our early results that seek to offer suitable transactional throughput and latency.