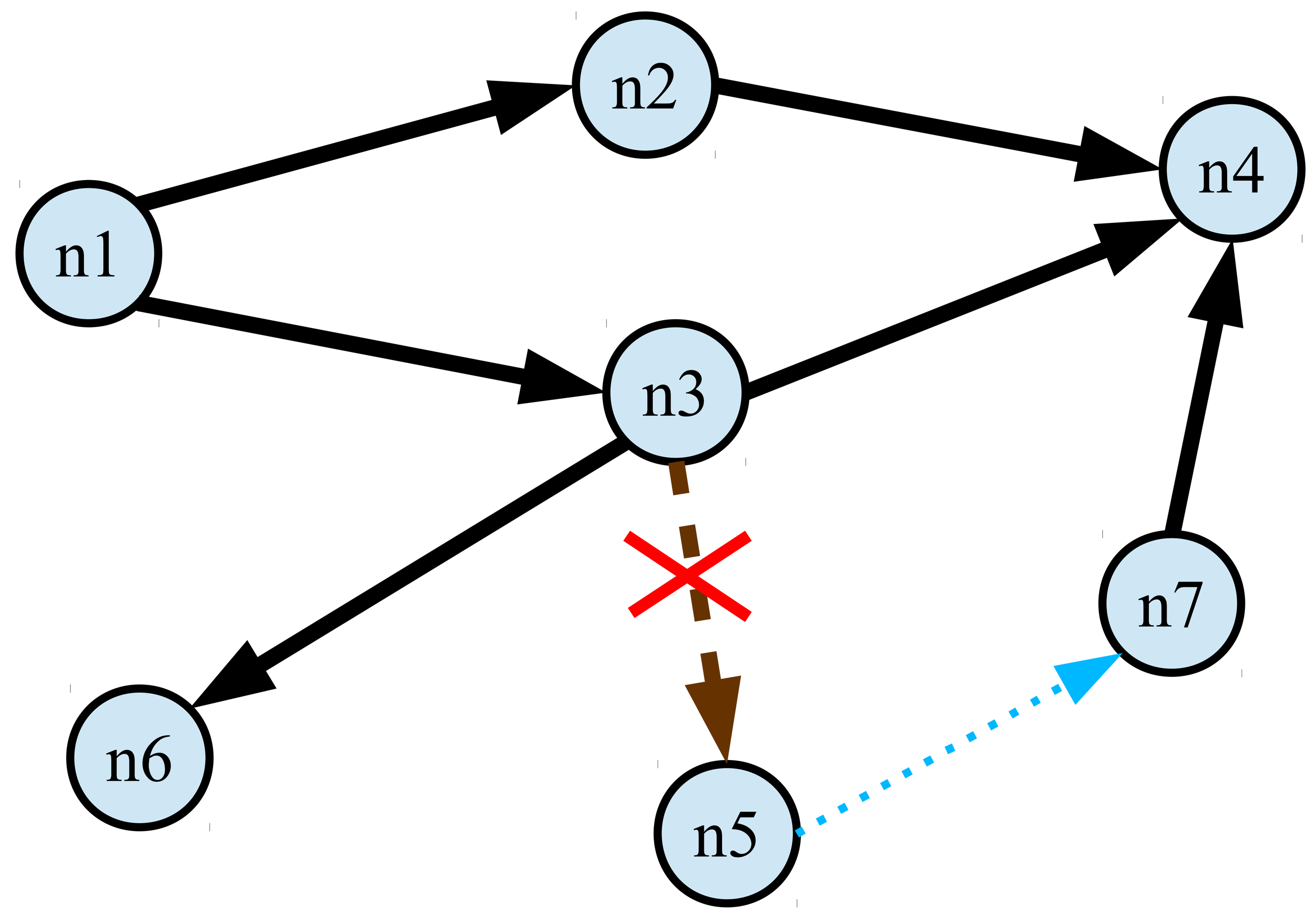
Weaver: A High-Performance, Transactional Graph Store Based on Refinable Timestamps

Ayush Dubey, Gregory D. Hill, Robert Escriva, Emin Gün Sirer
dubey@cs.cornell.edu, greghill@stanford.edu, {escriva,egs}@cs.cornell.edu
Cornell University

Motivation

Large graphs are ubiquitous

Key challenge is strong consistency and high performance for dynamic graphs



A graph undergoing an update which creates link (n_5, n_7) and an update which deletes (n_3, n_5), interleaved by a concurrent traversal query starting at host n_1. In absence of strong guarantees, the query can return path (n_1, n_3, n_5, n_7) which never existed at any instant.

Refinable Timestamps

Novel transaction ordering mechanism

- Parallel bank of gatekeepers assign a vector timestamp to each incoming transaction to achieve coarse, partial order
- Fine-grained timeline oracle reactively resolves conflicts between concurrent and conflicting transactions
- Establishing fine-grained order on-demand enables Weaver to reduce unnecessary synchronization by not ordering transactions that do not affect each other

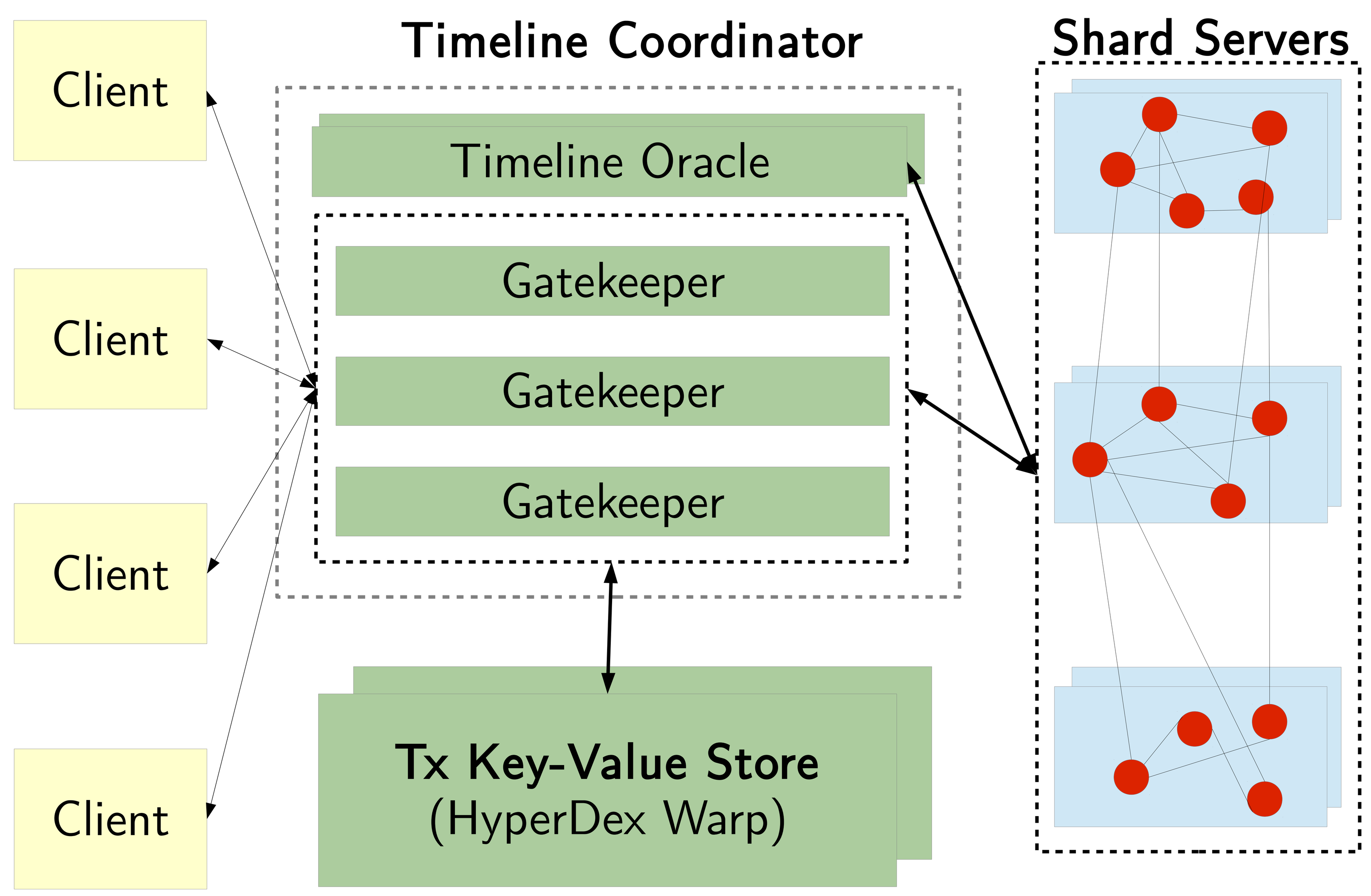
Reduced coordination is critical for long-running graph queries with large read set

Design and Implementation

Weaver provides ACID transactions on graphs

- **Generalized transactions** comprise read and write operations to create, read, modify, and delete pre-specified vertices and edges
- **Node programs** enable efficient execution of arbitrary read-only transactions such as traversals

Weaver Architecture

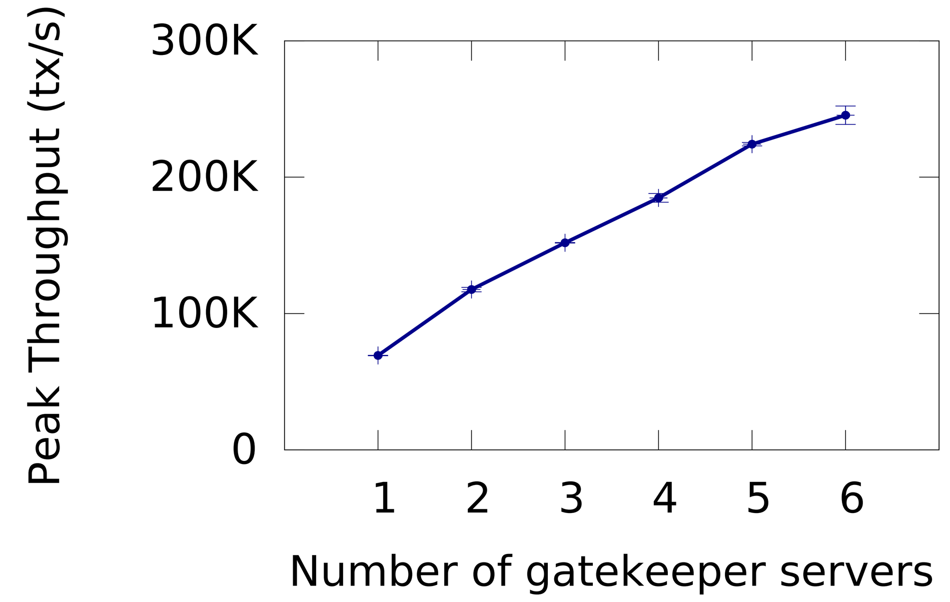


- Shard servers store in-memory, multi-versioned graph and execute node programs
- HyperDex Warp stores the graph data for fault tolerance
- Gatekeepers timestamp each request and periodically gossip their clocks amongst each other
- Shards dynamically migrate graph data across partitions to balance load and reduce query processing overhead

Refinable timestamp based transaction ordering mechanism presents a tradeoff between proactive costs—timestamp gossip—and reactive costs—timeline oracle

Scalability

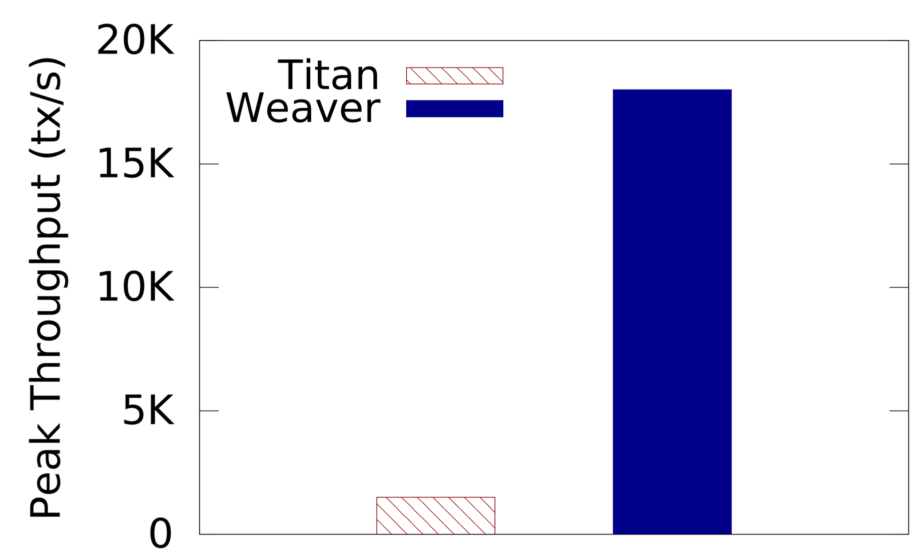
High Scalability



- Weaver's throughput scales linearly with additional gatekeeper servers on Twitter graph with 41M nodes, 1.47B edges

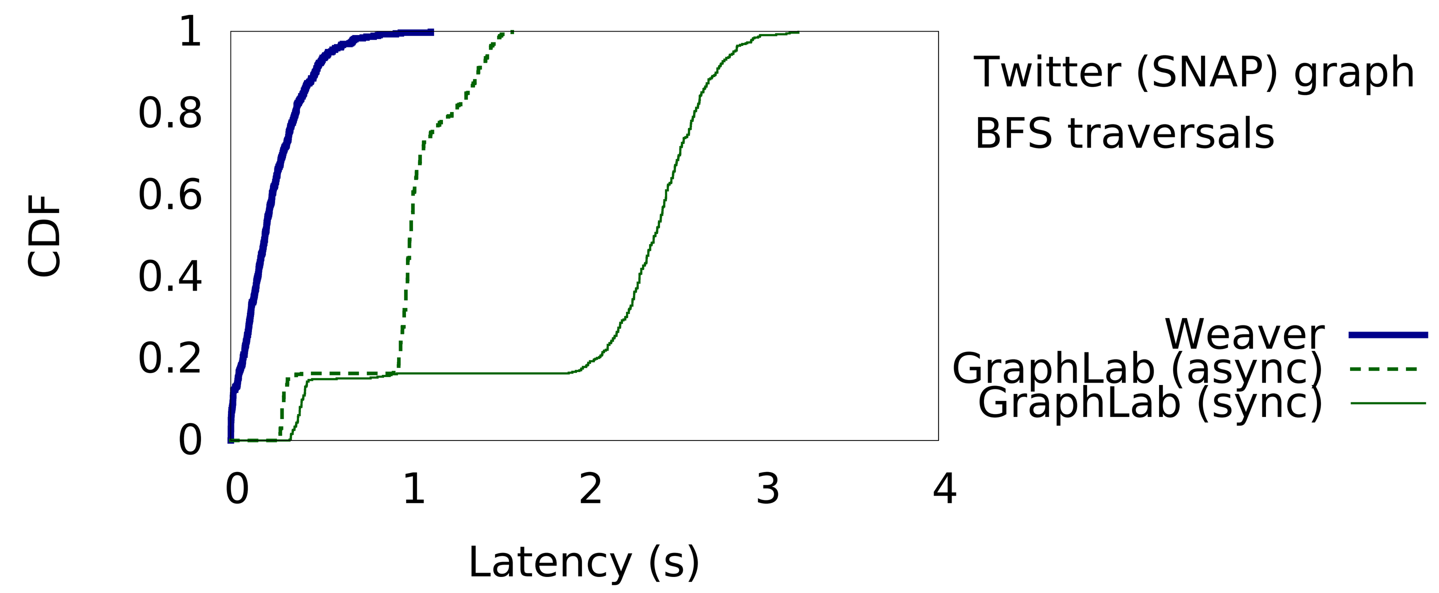
Performance

High Throughput



- Refinable timestamps achieve higher throughput than distributed locks due to higher concurrency and fewer aborts

Low Latency



- Refinable timestamps have low overhead, even compared to state-of-the-art, in-memory, offline graph processing systems