Extraction More Concurrency from Distributed Transactions with Rococo
Shuai Mu (THU, NYU), Yang Cui (NYU), Yang Zhang (NYU), Wyatt Lloyd (USC, Facebook), Jinyang Li (NYU)

Motivation
- Large OLTP systems need scalable distributed transactions
  - e.g. e-commerce sites
- Strict-serializability is preferred
  - Correctness
  - Simplicity in programming
- Conventional protocols are costly
  - OCC aborts
  - 2PL blocks

Rococo: Key Insights
- Delay execution until commit
  - Most pieces are deferrable
- Decentralized dependency tracking and propagating
  - Non-serializable interleavings are reorderable
- Offline checking to eliminate undesirable execution
  - A few pieces require immediate execution

Two-Phase Framework
- Start phase
  - Ship the pieces to each server
  - Establish a provisional execution order
  - Execute immediate pieces
- Commit phase
  - Establish a serializable execution order
  - Execute deferred pieces

Decentralized Dependency Tracking & Propagating
- Dependencies capture arrival order of conflicting pieces
- Dependency propagation detects non-serializable orders and reorders them

Immediate Pieces and Offline Checking
- Immediate Pieces
  - Pieces generating intermediate results cannot be arbitrarily deferred.
  - Executed “immediately” in the start phase, results used by other pieces
  - Tracking extra immediacy dependency information with immediate pieces
  - Immediacy must kept in execution order

- Offline Checking
  - Shasha’s SC-cycle analysis
  - Each SC-cycle should be merged with 2PL/OCC
  - Rococo’s SC-cycle analysis
  - After immediacy propagation, all SC-cycles with deferrable pieces are safe

Evaluation
- Rococo has higher throughput and consistently lower latency with increasing contentions
- Rococo performs well in a complex workload where contention increases as system scales