The RO-6 Theorem and Latency-Optimal Read-Only Transactions

Haonan Lu (USC)

Khiem Ngo (USC)

Chris Hodsdon (Rutgers-Camden, undergrad)

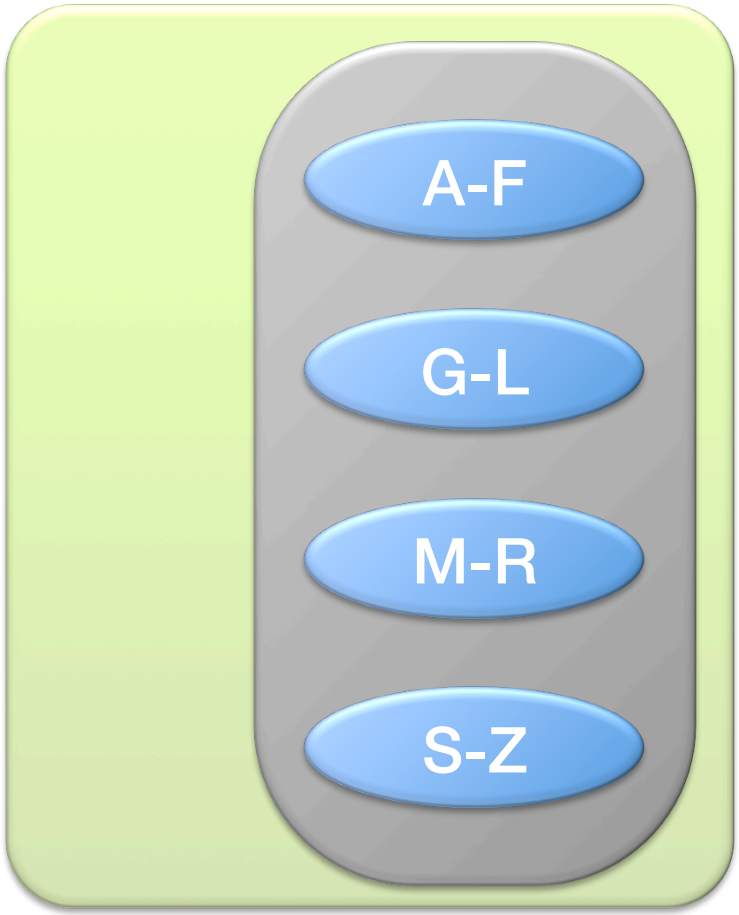
Shuai Mu (NYU, postdoc)

Wyatt Lloyd (USC)

Outline

- Read-only transactions
- Latency-optimality
- The RO-6 theorem
- New algorithms
- Preliminary results

Target: Scalable Data Stores

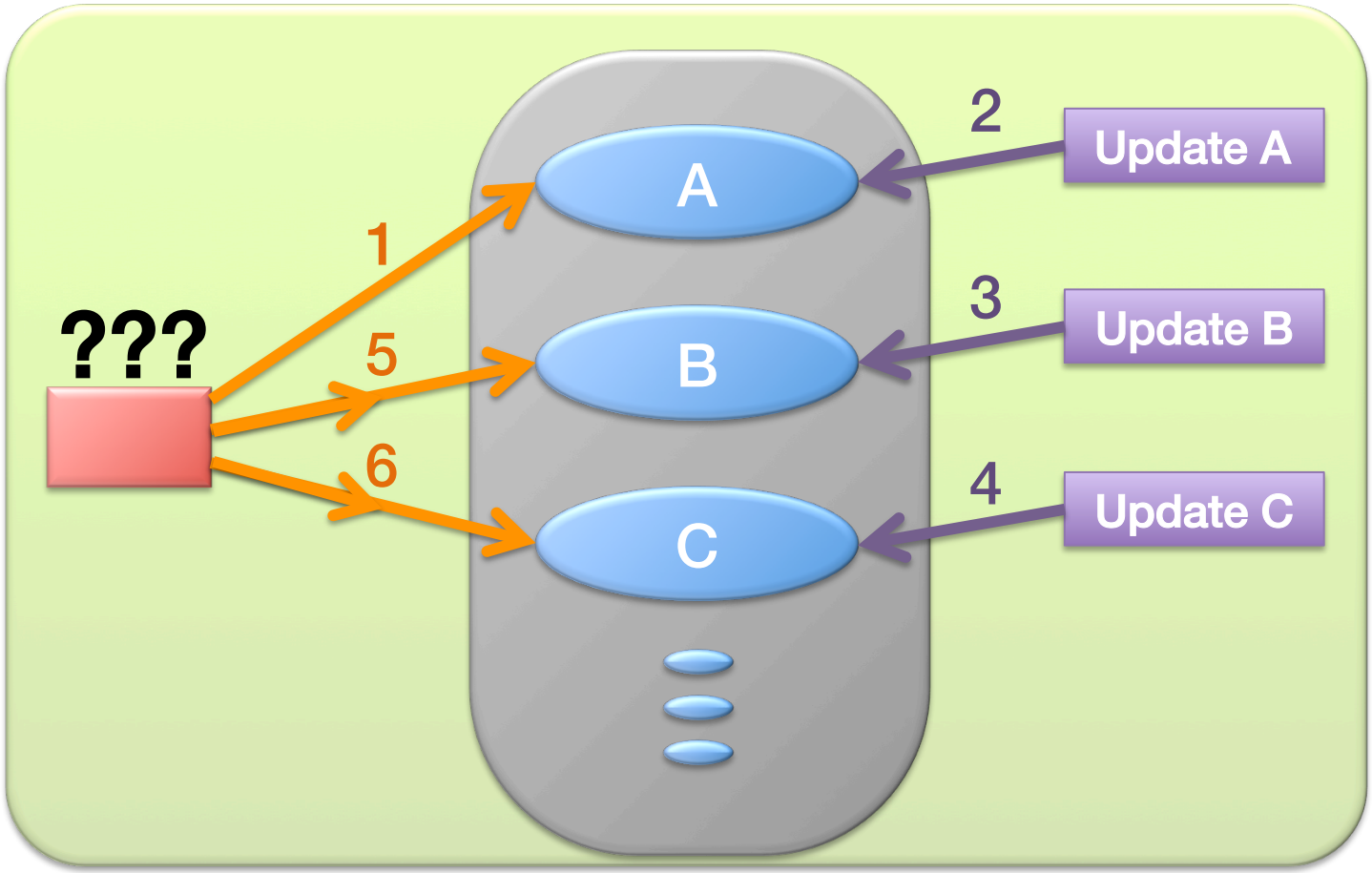


Key Point: Data is distributed across multiple nodes

Must contact multiple disjoint servers to read data

Read-Only Txns Necessary

Asynchronous requests + distributed data = ?????



Why Latency?

- Directly impacts user experience
- Multiplicative
  - "In practice, a single user request [has] critical path that is dozens of subqueries long." [Facebook, HotOS '15]
- Matters even more for geo-partitioned data
  - Khiem Ngo "K2" project looking at this
- Tractable!
  - Throughput is much harder to capture precisely

The RO-6 Theorem:
Impossibility Result for All 6 Properties

1. One round of messages
2. One version returned by servers
3. Non-blocking

} Latency

4. Strict serializability
5. Conflicting write transactions

} Power

6. Point-to-point messages

RO-6 Theorem

- Impossible to achieve all 6 together
  - Proof by contradiction

- Set of 6 properties is minimal
  - Any 5 of the 6 properties is achievable
    - We have the algorithms!
  - Still determining exact bounds on properties
    - 1 rounds vs 2 rounds vs N rounds

Why RO-6 Theorem?

- Avoid trying to do the impossible
  - This is how we got here
- Let us actually say this is the best we can do
  - Caveat: latency + read-only transactions
- Guide for understanding when we can improve read-only transaction algorithms

New Algorithms

Make reads faster at the expense of writes

- COPS-RO6: No conflicting write-only transactions
  - Writes check causal dependencies before being applied to determine preceding read-only txns
- Eiger-RO6: Blocking
  - 2PC for write transactions carries metadata about “earlier” read-only transactions, read-only transactions block to eliminate race-condition
- Eiger-Chris: Strict serializability
  - Provides “process serializability”, delays write completion
- Rococo-RO6: Blocking

Implementation & Evaluation

- COPS-RO6: Still to be implemented
- Eiger-RO6: Implemented
  - Lower latency, especially for large “expanding” read-only txns
  - No throughput improvement
    - Conjecture: throughput will improve for highly skewed workloads
- Eiger-Chris: Still to be implemented
- Rococo-RO6: Implemented
  - Lower latency & higher throughput
  - Rococo eliminated aborts for read/write txns, but read-only txns could abort. New algorithm for read-only txns avoid all aborts.
    - TPC-C: All 5 types of transactions avoid aborts now

Feedback

- New targets for RO-6 Theorem?
- Related work pointers
- Better terminology
- Eliminate “point-to-point” messages