Lock-free Transactional Support for Distributed Data Stores
Daniel Go`mez Ferro, Flavio Junqueira, Benjamin Reed, Maysam Yabandeh

Lock-based Transactional Support
- Data Server
- Lock Manager
- Client

• Locks hold by a failed client prevent the others from making progress.
• Need for additional resources to maintain the transactional data, e.g., locks.

Google Percolator
- Data Server + Lock Manager
- Client

• Extra load on the data servers → use batching to reduce the load.
• Inflicted non-negligible latency due to heavy use of batching

Lock-free Transactional Support
- Status Oracle (SO)

• A centralized server servicing transactional traffic such as start timestamp request, commit request and commit query.
• Service the requests from memory to achieve the highest performance

Sequence Diagram

Challenges & Solutions
• Partial data in memory → remember T_max, the maximum removed timestamp from memory.
• Reliability of in-memory data → Write-ahead logging via BookKeeper
• Limited CPU power → read-only replicas of SO at data servers or clients

Replication on Data Servers
- Status Oracle (SO)

By maintaining a read-only copy of SO data on some other nodes, some part of transactional traffic (queries about commit timestamps) could be answered by the read-only replicas, reducing the load on processing load on the SO. The read-only replicas could be stored in some new servers, the same data servers, or the clients. Our approach lightly replicates the SO data on clients.

Replication on Clients
- Status Oracle (SO)

• Piggyback ΔSO on Start Timestamp Response message
• No modification into data server → run on top of any key-value store
• Close-to-zero overhead on data servers
• A prototype is implemented on top of Hbase
• Scalable up to
• 50,000 write transactions per second
• 1000 client connections

SO Data Replication

This work has been partially supported by the Cumulo Nimbo project (ICT-257993), funded by the European Community.