Building Consistent Transactions with Inconsistent Replication

Irene Zhang, Naveen Kr. Sharma, Adriana Szekeres, Arvind Krishnamurthy, Dan R. K. Ports (University of Washington)
Motivation

- App programmers prefer distributed transactional storage with strong consistency → ease of use, strong guarantees
- **Tradeoffs**
  - fault tolerance: strongly consistent replication protocols are expensive (e.g. Paxos)
    - Megastore, Spanner
  - weakly consistent protocols are less costly but provide fewer (if any) guarantees (e.g. eventual consistency)
    - Dynamo, Cassandra
Common architecture for distributed txn’l systems

- **Distributed Transaction Protocol:**
  - atomic commitment protocol (2PC) + CC mechanism
  - e.g. 2PC + (2PL | OCC | MVCC)

- **Replication Protocol:**
  - e.g. Paxos, Viewstamped Replication
Spanner-like system

- writes buffered at client until commit
- read ops must go to shard leaders to ensure order across replicas (gets value & timestamp of any data read)
- Commit takes at least 2 round trips
Observation

- Existing distributed transaction storage systems that integrate both protocols waste work and performance due to this redundancy.
- Is it possible to remove this redundancy and still provide read-write transactions with the same guarantees as Spanner? **YES.**
  - linearizable transaction ordering
  - globally consistent reads across database at a timestamp
- How? **Replication with no consistency**
Key Contributions

- Define **IR (inconsistent replication)** → new replication protocol → fault tolerance without consistency
- Design **TAPIR (Transactional Application Protocol for IR)** → new distributed transaction protocol → linearizable transaction ordering using IR (Spanner)
- Build/evaluate **TAPIR-KV** → high-performance transactional storage (TAPIR + IR)
Inconsistent Replication

- Fault tolerance without consistency
  - ordered op log replaced by an unordered op set
- Used with a higher-level protocol: application protocol
  - to decide/recover the outcome of conflicting operations
- Can invoke ops in 2 modes: inconsistent and consensus
  - Both: execute in any order
  - Consensus only: returns a single consensus result
- Guarantees:
  - fault tolerance: successful ops & consensus results are persistent
  - visibility: for each pair of operations, at least one is visible to the other
IR Application Protocol Interface
IR Protocol: Operation Processing

- IR can complete **inconsistent operations** with a single *round-trip* to $f+1$ replicas and no coordination across replicas

- **consensus operations**
  - fast path: if $[3/2 f]+1$ replicas return *matching* results
    - common case, single round-trip
  - slow path: if otherwise
    - two round-trips to at least $f+1$ replicas
IR Protocol: Replica Recovery & Synchronization

- uses single protocol for recovering failed replicas & synchronizing replicas → **View change**
- Protocol is identical to Viewstamp Replication (Oki, Liskov) except that the leader must *merge* records from the latest view
  - leader relies on application protocol to determine consensus results
  - result of *merge* is the “master record”, used to synchronize other replicas
**TAPIR**

- Transactional Application Protocol for IR
  - Efficiently leverages IR’s weak guarantees to provide high-performance linearizable transactions (Spanner)
- Clients: front-end app servers (possibly at same datacenter)
- Applications interact with TAPIR (not IR)
  - once an app calls “commit”, it cannot abort
  - this allows TAPIR to use clients as 2PC coordinators
- Replicas keep a log of committed/aborted txns in timestamp order
- Replicas also maintain a versioned data store
TAPIR: Transaction Processing

- Uses OCC
  - concentrates all ordering decisions into a single set of validation checks
  - only requires one consensus operation ("prepare")
    - decide function: commit if a majority of replicas replied "prepare-ok"
Spanner-like system vs TAPIR
Experimental Setup

- built TAPIR-KV (transactional key-value store)
- Google Compute Engine (GCE), 3 geographical regions
  → US, Europe, Asia
  → VMs placed in different availability zones
- server specs:
  → virt. single core 2.6 GHz Intel Xeon, 8 GB RAM, 1 Gb NIC
- comparison systems
  → OCC-STORE (standard OCC + 2PC), LOCK-STORE (Spanner)
- workloads
  → Retwis, YCSB+T
Results: RTT & clock synchronization

- RTTs:
  - US-Europe: 110 ms
  - US-Asia: 165 ms
  - Europe-Asia: 260 ms

- Low clock skew (0.1 - 3.4 ms), BUT has a long tail
  - Worst case ~27 ms

- Unlike Spanner, TAPIR performance depends on actual clock skew, not a worst-case bound
Avg. Rewtis transactional latency vs. throughput

- Rewtis
- single data center
- US region only
- 10 shards
- 3 replicas/shard
- 10M keys
- zipf coef: 0.75
Avg. wide-area latency for Rewtis transactions

- 1 replica per shard in each geographical region
- leader in US (if any)
- client in US, Asia, or Europe
Abort rates at varying Zipf coefficients

- single region
- replicas in 3 availability zones
- constant load of 5000 txns/s
Comparison with weakly consistent storage systems

- YCSB+T
- single shard
- 3 replicas
- 1M keys
- MongoDB & Redis: → master-slave → set to use synch. replication
- Cassandra: → set replication level to 2
Conclusion

- possible build distributed transactions with better performance and strong consistency semantics on top of a replication protocol with no consistency
- relative to conventional transactional storage systems → lowers commit latency by 50%
  → increases throughput by 3x
- performance is competitive with weakly-consistent systems while offering much stronger guarantees
The end!
Techniques to improve performance

- optimize for read-only transactions → Megastore, Spanner
- use more restrictive transaction models → VoltDB
- provide weaker consistency guarantees → Dynamo, MongoDB
Observation

- Existing distributed transaction storage systems that integrate both protocols waste work and performance because both enforce strong consistency.
IR Protocol

- Unique operation ID: IR client ID + op counter
- Replica maintain unordered records of executed ops and consensus results