Don’t Settle for Eventual Consistency
Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, David G. Andersen
Princeton University, Intel Labs, and Carnegie Mellon University

**Motivation**
- Distributed data stores support complex online applications
  - e.g. social networks
- Theory constrains properties
  - CAP Theorem
  - Seq Consistency || Low Latency
- Most practical systems adopt eventual consistency
  - Complicates application logic
  - Exposes inconsistencies to users

**Web Service Architecture**

**Ideal Properties**
- Availability
- Low Latency
- Partition Tolerance
- Scalability
- Stronger Consistency
- Systems with the first four properties are ALPS systems

**Client Library**
- Interface hides complexity from programmer
- Calls include a context that tracks causality
- Multiget provides a consistent view of multiple keys, even from diff. nodes

**Key-Value Store**
- Client operations are local, replication occurs in the background
  - Available, Low Latency, Partition Tolerance
- Lamport timestamps version writes
- PK Seq Consistency
- Put_after and dep_check operations order replication b/w clusters and nodes
  - Causal Consistency

**Clusters of Order Preserving Servers**

**CKS Consistency**
- Causal Consistency
  - Related ops appear in the correct order
- Per-Key Sequential Consistency
  - Global order of writes to each key
  - Strongest ever achieved for an ALPS system
  - Linearizability > Seq. > CKS > Causal
  - PK Seq. > Eventual

**CKS Example**
1. Alice changes her blog ACL to “friends only”
2. Alice posts “Anyone for coffee? First reply wins!”
3. At similar times, Bob and Charlie both reply

CKS (Causal): Only friends can see the blog post
Eventual: Anyone can see the post

CKS (PK Seq): Only one friend wins coffee
Eventual: Both friends can win

**Challenges**
- Minimize space footprint
- Garbage collect old state
- Minimize overhead of consistent replication
- Leverage transitivity of causality
- Ensure fast multigets: Worst-case 2 rounds under concurrent writes
  - Get_by_version

**Implementation**
- Built on top of FAWN-KV
- ~13,000 LOC
- Latency < 1ms
- Get throughput ≈ FAWN
- Put throughput < FAWN