# Ordering Transactions with Prediction in Distributed Object Stores

Ittay Eyal<sup>1</sup> Ken Birman<sup>1</sup> Idit Keidar<sup>2</sup> Robbert van-Renesse<sup>1</sup>

<sup>2</sup> Technion <sup>1</sup> Cornell

### In a world of big data



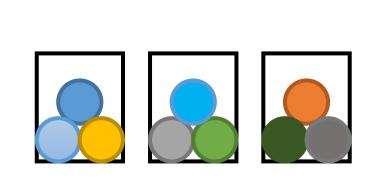
### we want transactions

begin txn Reads (return value) writes (return ack)

end txn

### of sharded data





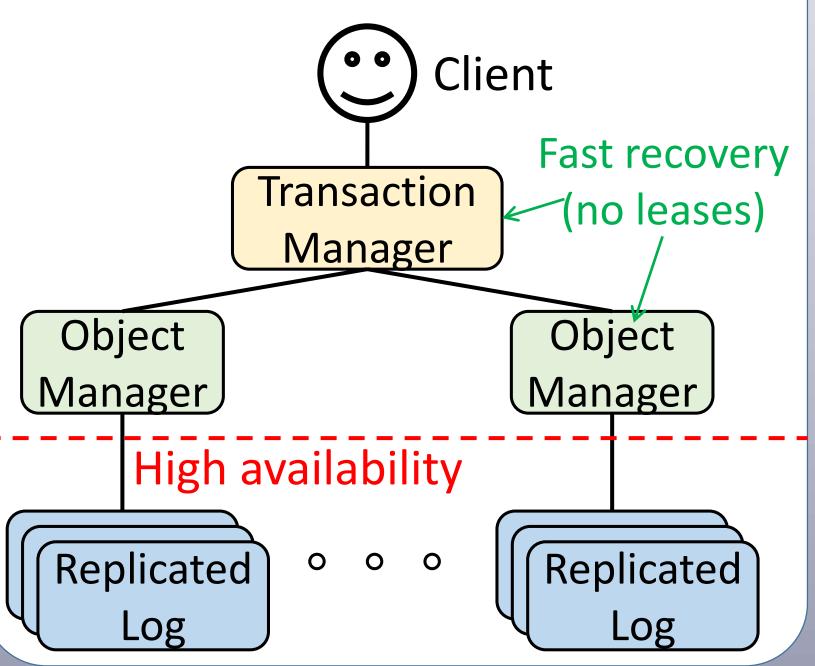
### with ACID guarantees

- Atomic Atomic Consistent transactions Isolated
- Durable > High availability

## ACID-RAIN: Ordering with Prediction, Committing with Independent Logs

(returns commit/abort)

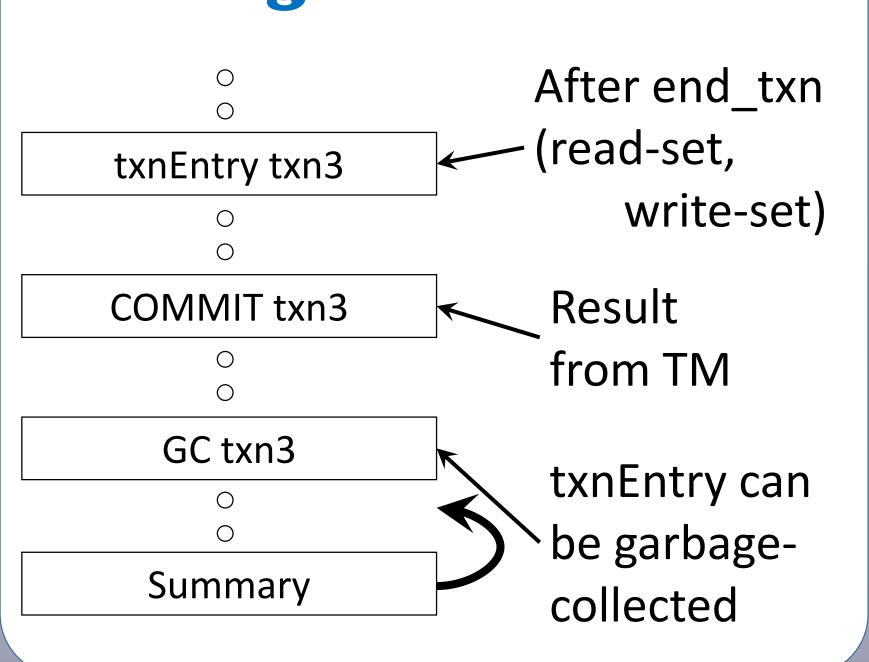
# Architecture



### Concurrency Control

- 1. Optimistic, transactions run speculatively and then certify.
- 2. Conflict detection w/ timestamps.
- 3. Version reservation (lock on future version) by prediction.
- 4. Final certification at transaction end → lock-free: can replace slow/failed nodes immediately; reservations are only hints.

### Log Structure



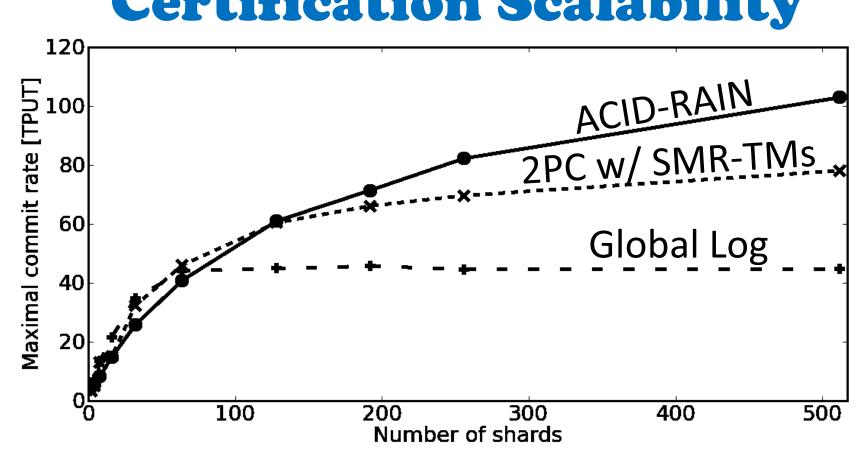
### **Execution Example** with Prediction

- Prediction and reservation.
- 2. Transaction run.
- 3. Certification.
- 4. Garbage collection (asynchronous)

# Simulation Results

- Custom-made simulator.
- Transactional YCSB workloads.
- Uniform random object access.

### Certification Scalability



- Global log: Forms a bottleneck.
- **2PC with SMR TMs**: longer certification time so higher contention.

### **Benefits of Prediction**

•• recall = 0.0

■ recall = 0.5

**—** recall = 1.0

precision = 0.25

precision = 0.50

**—** precision = 1.00

Number of objects

Number of objects

Different recall ratios with perfect precision (no wrong guesses). recall = 0: no prediction and no reservation (classical approach) recall = 1.0: predicting all accesses.

Better recall 

higher commit ratio

Different precision ratios (wrong guesses) with perfect recall. Bad precision 

more conflicts

in small data sets

