Transactional Caching of Application Data using Recent Snapshots

Dan R. K. Ports  Austin T. Clements  Irene Y. Zhang
Samuel Madden  Barbara Liskov

MIT CSAIL
How to improve performance of DB-driven web site?

Distributed in-memory caching (e.g. memcached)
Distributed In-Memory Caching

Database

Application
Distributed In-Memory Caching

Application

Cache

Database
Distributed In-Memory Caching

- in-memory DHT; very lightweight
- stores *application* objects (not part of DB)
Databases work hard to provide transactional consistency.

Existing application caches violate these guarantees!
Consistency Properties

usual goal: 
  *freshness*: cache is up-to-date with database

our goal: 
  *consistency*: all accesses to cache and database in a transaction see the same snapshot

Can’t guarantee both without blocking!
Consistency Properties

usual goal:

freshness: cache is up-to-date with database

our goal:

consistency: all accesses to cache and database in a transaction see the same snapshot

Can’t guarantee both without blocking!
Embracing Staleness

Run r/o transactions on *previous* snapshots
  – avoids blocking
  – improves cache utilization

Safe: stale data is *already* everywhere!

Allow application control over staleness
TxCache Anatomy

- Cache is a versioned DHT, tagged by validity interval
TxCache Anatomy

- Cache is a versioned DHT, tagged by *validity interval*
- Database returns validity interval with each query

```
SELECT * FROM users...
  [...result...]
  VALID FROM
  t=50 TO t=53
```
TxCache Anatomy

- Cache is a versioned DHT, tagged by *validity interval*
- Database returns validity interval with each query
- Library assigns timestamp to each transaction
- Uses timestamp to request data from cache & DB