Geo-replication integrated all the way to the client machine

Motivation

limitations of server-centric geo-replication

- Ad-hoc client-side caching – today’s solution plagued with issues:
  - Error-prone application-level logic
  - Inconsistent on partial cache misses or failures (no metadata/updates)

Goal

extend geo-replication to the client machine

- Lower latency and improved availability for some operations
- (Causally) consistent access to partial replicas despite faults

Problem (1) with naïve approach

liveness of causal consistency w/o full (meta)data

example execution: replicated updates and causal dependencies

Failover problem: reads in NY blocked (y=7) or inconsistent (y=0)!

Cause: non-replicated causal dependency

Problem (2) with naïve approach

inefficient or insufficient metadata

Approach A

client-assigned update id + Version Vectors encoding causal dependencies

id: \( \{C_0, 17\} \) depends on:

\[ C_0=16, C_0=5, \ldots, C_3=3 \]

Efficiency problem: unsustainable vector size

Approach B

server-assigned update id + any efficient encoding

depends on:

\[ WA8, WA7, WA6, \ldots, WA0 \]

Hybrid approach

separated concerns: update identity and summary

update lifecycle

1. Client assigns a unique id:
   prevents duplicated update log entries in case of failures

2a. Server assigns an alias id:
   reference for efficient summary of updates

2b. Rare failure path (failover/retry):
   Assign new server id if needed identify any duplicates by client id

3. Metadata compaction
   upon eventual full replication

This research is supported in part by ANR project ConcoRDanT (ANR-10-BLAN 0208), by the Google Europe Fellowship in Distributed Computing awarded to Marek Zawirski, by Portuguese FCT/MEC projects PEst-OE/EEI/UI0527/2011 and PTD/EEI-SRC/1837/2012 and PhD scholarship awarded to Valter Balegas (SFRH/BD/87540/2012).