Recitation 19 — Consistency Rationing

Intro

- Consistency Rationing is motivated by the idea that many applications don't need very strong consistency guarantees all of the time (or sometimes ever), but often need something stronger than eventual consistency.
- Motivated by a few use cases: a web shop (esp. the problem of tracking inventory), auction system, collaborative editing (e.g., google docs)

Types of data/consistency

- Session consistency: weakest form of consistency used in this system. A client sees its own writes ("read-your-own-writes monotonicity"), but other clients may not see those writes. Good performance, but can lead to inconsistencies.
- Serializability: conflict-serializability, just like what you saw in lecture.
- Adaptive: Switch between session consistency and serializability according to various adaptive policies.

Adaptive Policies

- General Policy: estimate the probability of a conflict, apply strong consistency guarantees only when that probability is high.
- Time Policy: change consistency type based on a timestamp (e.g., "session consistency between 2pm and 4pm").
- A variety of policies for numeric types, including fixed threshold (strong consistency once the value is below a threshold), demarcation (let servers change local values of a variable within some predefined bound), dynamic policy (probabilistic guarantees)

Broader questions

- Lots of trade-offs here: Performance, monetary cost, understandability, …
- How would you as a systems designer make these trade-offs? Is "understandability" important?
- How do the papers you've been reading in recitation relate to what you're seeing in lecture?