Weak Consistency

Dan Ports, CSEP 552

CAP Theorem

- Can't have all three of *consistency*, *availability*, and *tolerance to partitions*

- (but the devil is in the details!)

CAP

- Eric Brewer, 2000: conjecture on reliable distributed systems
- Gilbert & Lynch 2002: proved
(for certain values of “consistency” and “availability”)
- really influential and really controversial
    - motivated the consistency model in many NoSQL systems
    - Stonebraker: “encourages engineers to make awful decisions”
- usually misinterpreted!

Usual Formulation

- Choose any two of:
  consistency, availability, partition tolerance
- Then: want availability, so need to give up on consistency
- Or maybe: want consistency, so availability must suffer
- Implies 3 possibilities: CA, AP, CP

First problem: type error

- Consistency and availability are properties of the system

- Partition tolerance is an assumption about the environment

- What does it mean to (not) choose partition tolerance?

  - i.e., what does it mean to have a CA system?

- Better phrasing: when the network is partitioned, do we give up on consistency or availability?

Other problems

- What does (not) choosing consistency mean?
  What about weak consistency levels?
- What does not providing availability mean?
  Does that mean the system is always down?
- What if network partitions are rare?
  What happens the rest of the time?

A more precise formulation

- (from Gilbert & Lynch's proof)
- model: a set of processes connected by a network subject to communication failures
  - meaning messages may be delayed or lost
- it is impossible to implement a non-trivial linearizable service
- that guarantees a response to any request from any process

Proving this statement

- Not too surprising
- Suppose there are two nodes, A and B
and they can't communicate
- first: write(x) on A
- then: read(x) on B
- availability says B's request needs to succeed,
linearizability says it needs to return A's value

How does this relate to FLP?

- CAP: when messages can be delayed or lost in the network, can’t have both consistency and availability
- FLP: when one node can fail and the network is asynchronous, can’t reliably solve consensus
- FLP is a stronger (i.e., more surprising) result
  - CAP allows network partitions / packets lost entirely
  - CAP: every node to remain available
    FLP: failed nodes don’t need to come to consensus

Examples

- Where do systems we've seen before fall in? Are they consistent? Available?
  - Lab 2
  - Paxos
  - Chubby
  - Spanner
  - Dynamo

Paxos availability

- Wasn't Paxos designed to provide high availability and fault tolerance?
- Remains available as long as a majority is up and can communicate
- not availability in the CAP theorem sense!
  would require any node to be able to participate even when partitioned!
- Is this enough?

Do partitions matter?

- Stonebraker: "it doesn’t much matter what you do when confronted with network partitions" because they’re so rare
- Do you agree?

Do partitions matter?

- OK, but they should still be rare
- When the system is not partitioned, can we have both consistency and availability?
- As far as the CAP theorem is concerned, yes!
- In practice?
  - systems that give up availability usually only fail when there's a partition
  - systems that give up consistency usually do so all the time. Why?

Another "P": Performance

- providing strong consistency means coordinating across replicas
- means that some requests must wait for a cross-replica round trip to finish
- weak consistency can have higher performance
  - write locally, propagate changes to other replicas in background

CAP implications

- Need to give up on consistency when
  - always want the system to be online
  - need to support disconnected operation
  - need faster replies than majority RTT
- But can have consistency and availability together when a majority of nodes can communicate
  - and can redirect clients to that majority

Dynamo and COPS

- What kind of consistency can we provide if we want a system with
  - high availability
  - low latency
  - partition tolerance

Dynamo

- What consistency level does Dynamo provide?
- How do inconsistencies arise?

- Sloppy quorums: read at quorum of N nodes
  - …but might not be a majority
  - …but might not always be the same N nodes (just take healthy ones)

COPS

- Guarantees *causal* consistency instead of eventual (or no) consistency
    - recall Facebook example: remove friend, post message
    - if get returns result of update X, also reflects all updates that causally preceed X
    - but causally concurrent updates can proceed in any other
- “Causal+”: conflicts will eventually converge at all replicas

COPS Implementation

- Multiple sites, each with full copy of the data
  - partitioned and replicated w/ chain replication
- Writes return to client after updating local site
- then updates propagated asynchronously to others
  - Lamport clocks and dependency lists in update message — ensures they're applied in order

Next week

- Co-Designing Distributed Systems and the Network: Speculative Paxos and NOPaxos (Adriana Szekeres)
- MetaSync: File Synchronization Across Multiple Untrusted Sources (Haichen Shen)
- Verdi: A Framework for Implementing and Formally Verifying Distributed Systems (James Wilcox and Doug Woos)
- Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency (Naveen Kr. Sharma)