CAP Theorem

1. Gilbert and Lynch. *Perspectives on the CAP Theorem. Computer J. 2012.*

2. Brewer. *CAP Twelve Years Later: How the "Rules" Have Changed*

Course updates

- Compose update + email me to schedule a chat with me by this Friday

- *Note*: Piazza response now due 18 hours before class (no longer 24)

  - e.g., 2PM instead of 8AM the day before class (Vancouver time)

CAP

- CA, AP, CP system — different types of tradeoffs
- Simple proof — consider two nodes with a potential partition between them
- Note, CA property of system. P is a property of the environment (network).
- Introduce a third node, things get more complex: can have asymmetric relationships between each pair of nodes. Could choose CA on one edge and CP on another edge
- Facebook example: core nodes CA-style and have consistent state. Periphery nodes write through to the core, and handle reads. Scalable, partitions geographically and also by type of operation.

CAP

- Breakout discussion:
  - What are some approaches that the papers introduce for handling the seemingly impossible way of reconciling C,A,P?
  - (The papers encourage this) Get rid of consistency! But only for the systems where it makes sense for the type of service you are providing. Trade off consistency for more availability.
  - Shopping cart/seats on a plane: large inventory — no issues with being inconsistent (in accounting # of items). When more scarce (closer to 0), then consistency of # of items is more important.
  - Spectrum of consistency, e.g., *k-set agreement*
  - No simple universal measure of consistency
  - In most cases strong consistency is not what the client needs
  - Okay to make mistakes and compensate for them later (vouchers on overbooked flights) and banks that charge you fees (win for the bank even!)
  - CRDT eventual consistency — has a notion of recovery in terms of merging different replicated states
  - Continuous consistency — monitor at runtime and modify as you see fit

C,A,P

- Giving up C => choosing a weaker C
- *Weaker C* requires a definition
- You have to reason (more carefully) about what's acceptable for your system: *requirements*.
- Strong C: easy consensus (between people)
- Brewer talks about *invariants:* build a matrix of properties (invariants) X operations in your system. And decide on how each invariant will be re-established after a partition.
    - Fine-grained reasoning at the operation/data layer. Doesn't require a catch-all choice of C.

CAP

- What are some approaches that the papers introduce for handling the seemingly impossible way of reconciling C,A,P?
- Segmentation/partition of the system based on different aspects. Geography, user, data, function
- Web cache versus Chubby (consensus service)
- Data centers simplify assumptions: Google builds the centre => knows its properties well => closer to the edge when making trade-offs between C,A,P.
- “Eventual (weak) synchrony” assumption => minimum necessary for consensus (Dwork et al. [14] in the first paper)
    - Discusses synchrony (network liveness) but not during partitions
- Failure detectors (and weakest variants for consensus)
    - Important for detecting “failures” including partitions
    - Determine performance of the system during failure events

CAP

- What are some approaches that the papers introduce for handling the seemingly impossible way of reconciling C,A,P?

- Partitions may be one-way (observable by only some of the parties). Example is a routing failure.

- Partitions may flip-flop (observable one moment, and not observable another moment). (Different from fail-stop / halting of processes.)

CAP ~ FLP

- What's the relationship between CAP and FLP?
    - FLP result is about impossibility of consensus in an async network
- *Safety: some property is true at all times*
- *Liveness: some property is eventually true*
- Consensus problem: requires safety (agreement) and liveness (*eventually* there is agreement)
- "CAP" problem: also encodes safety (strong C) and liveness (A)
- Both are about the tension between safety and liveness in the presence of failures (CAP: partitions, Consensus: node failure/async net)
- Safety in consensus is "harder" than safety in CAP: consensus is a superset of C in CAP.
- CAP =/=> FLP and FLP =/=> CAP (different notions of failure)
    - CAP => "partitioned version of FLP"
- *"Getting around FLP might inform how to get around CAP"* — thesis of the paper
    - K-agreement: relax agreement in consensus — relax consistency in CAP (weak consistency)
    - Weakest failure detector: find boundary for failure tolerance — role that detection of P plays in CAP
    - Eventually synchrony: boundary for min. synchrony necessary for consensus — relax availability guarantee in CAP

CAP ~ FLP

- So, “impossibility” results: are they actually useful?
- *“despite… practitioners .. must still do the impossible” :-) :-(*
- “Excuse to let developers build systems with strange guarantees that are theoretically unsound” ?

- Impossibility -> design trade-offs in practice
- Impossibility results lead to more theory work to better understand the underlying constraints. *Those* results lead to more practical considerations (eventual synchrony)
- Trivially satisfy CAP if your system can reply with “I don’t know”. Can also satisfy it if all you return are 100 digits of \pi
    - What is a *trivial* system that can avoid C.A.P?

Choosing (or not) P

- Partition tolerance in WAN is a reality. Is this really a choice?

- Versus Partition tolerance can be engineered

Next: CRDTs

- So, what are some ways of resolving CAP?
- ..why not try to weaken the consistency level? (“AP systems”)
- We’ll discuss two flavours of this weakening
  - CRDTs
  - OR (optimistic replication)