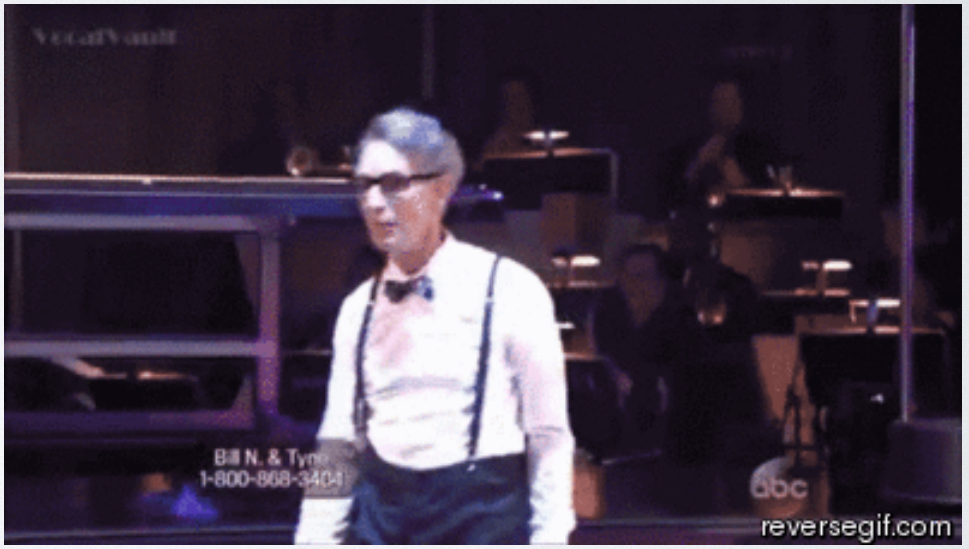
A DECADE OF CAP

9 October 2015

OUR MISSION

- Context
- Definitions and Proof
- Legit Criticism
- Less Legit Criticism
- Discussion



I'M NOT A THEORETICIAN…

…so this is a *practitioner's* view of a theoretically complex topic.

"There ain't no such thing as a free lunch."

–Robert Heinlein (and every economist ever)

- *Safety*: nothing bad ever happens
- *Liveness:* eventually something good happens
- *The world is* 💩*:* networks fail, processes fail, and the hackers are smarter than me

"Consistency, availability, and tolerance to network partitions: you can have at most two of these properties for any shared-data system."

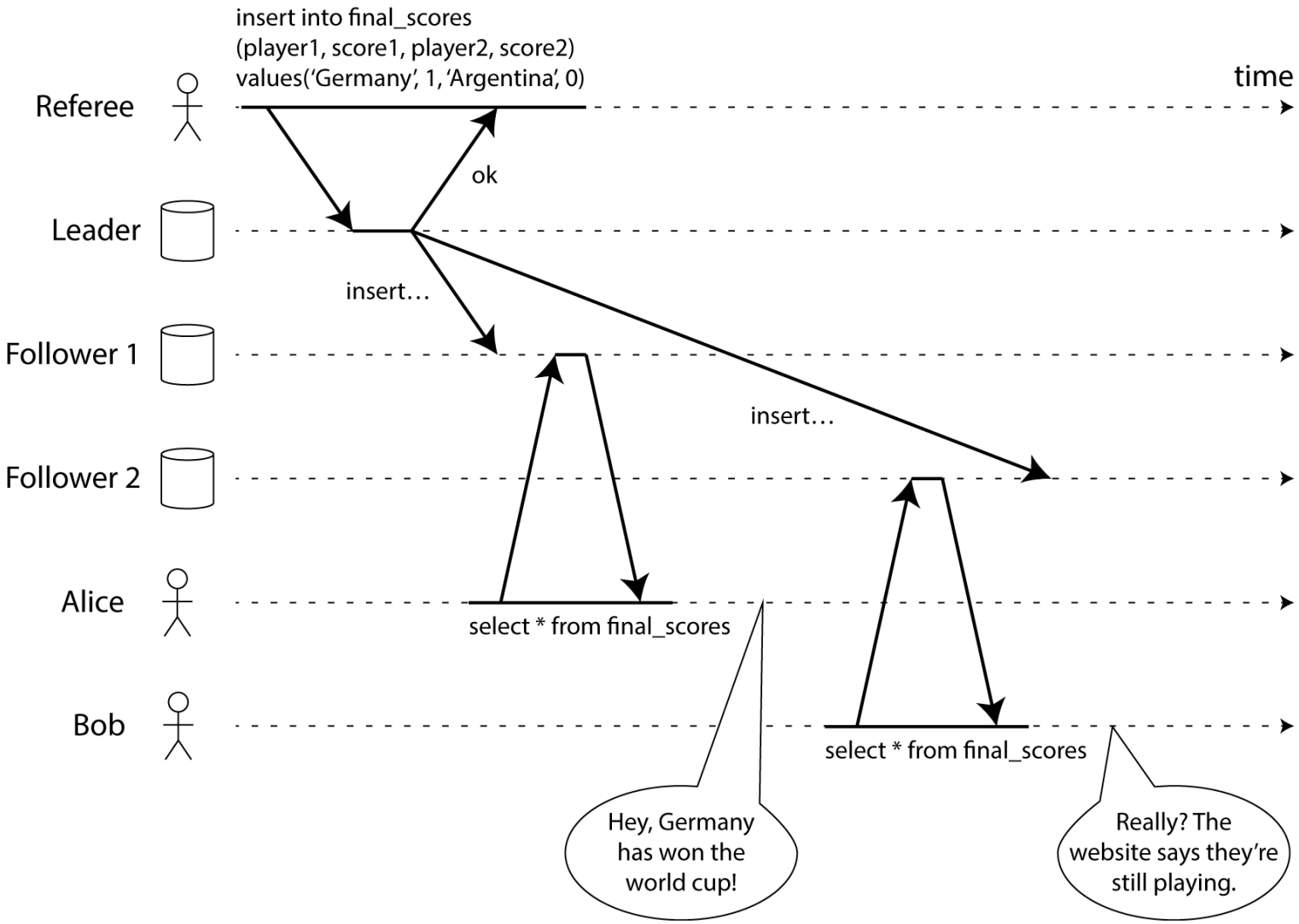
–Eric Brewer, 2000

CAP: SYSTEM

- Imagine a set of servers, in multiple data centers, receiving requests from an arbitrary number of clients.
- Links between servers have at-most-once delivery and never introduce messages de novo, but can drop an unbounded number of messages and introduce unbounded message delay.
- Processes never crash.
- Processes are partially synchronized: local clocks progress monotonically at a rate approximately equal to real time, but different clocks aren't synchronized
- Shared data is a single read-write register.

CAP: CONSISTENCY

- Consistency is a strong *safety* property.
- From the clients' perspective, system behaves as though it ran on a single node (*atomic*).
- Equivalently, "there must exist a total order on all operations such that each operation looks as if it were completed at a single instant" (*linearizability).*

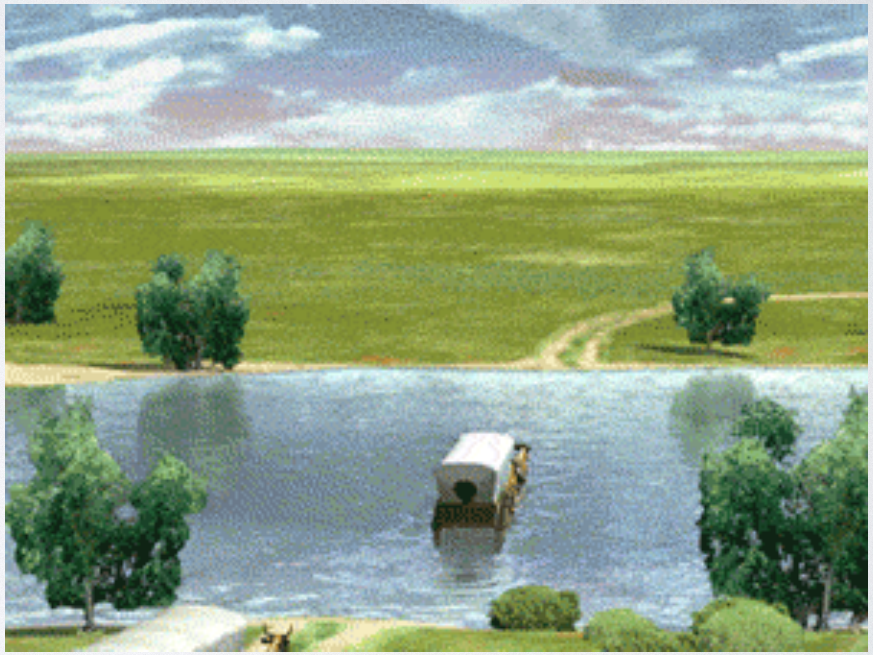

insert into final_scores
(player1, score1, player2, score2)
values('Germany', 1, 'Argentina', 0)
time
Referee
ok
Leader
insert…
Follower 1
insert…
Follower 2
Alice
select * from final_scores
Bob
select * from final_scores
Hey, Germany has won the world cup!
Really? The website says they're still playing.

CAP: AVAILABILITY

- Availability is a *liveness* property.
- "For a distributed system to be continuously available, every request received by a non-failing node in the system must result in a response. That is, any algorithm used by the service must eventually terminate."

CAP: PARTITION-TOLERANCE

- Partitions are the 💩 ruining your service.
- Links between processes can permanently stop transmitting messages.



PROOF

Divide the set of processes into *A* and *B*. The initial value in the register is `foo` for all processes. Partition the two. A client makes a request to a process in *A* and sets the register to `bar`. To maintain availability, the process in *A* must eventually respond with a success, even though it will never be able to communicate with *B*. Once the write in *A* completes, another client reads from *B*. The process in *B* must either fail to respond or return `foo`, since it doesn't know the register's value has changed.

To me, legit criticism of CAP focuses on the fact that its definitions are *too restrictive*. Because the definitions are so narrow, the result is too weak to be practically useful (though it's still true).

PROBLEMS: CONSISTENCY

- Linearizability is an unrealistically strong safety property. Modern CPUs don't provide linearizable access to local memory by default.
- Proof depends on an infinitely long partition, which is unusual. If we restrict ourselves to bounded-length partitions, we can achieve eventual consistency (whatever that means).
- What about probabilistic consistency?

PROBLEMS: AVAILABILITY

- No latency bounds.
- Real-world systems can be extremely fault-tolerant without leaving *every* node available.

"Liveness properties are inherently problematic. The question of whether a real system satisfies a liveness property is meaningless; it can be answered only by observing the system for an infinite length of time, and real systems don't run forever. Liveness is always an approximation to the property we really care about. We want a program to terminate within 100 years, but proving that it does would require the addition of distracting timing assumptions. So, we prove the weaker condition that the program eventually terminates. This doesn't prove that the program will terminate within our lifetimes, but it does demonstrate the absence of infinite loops."

–Leslie Lamport

PROBLEMS: PARTITION-TOLERANCE

- Partitions are only one type of failure, and infinitely long partitions aren't even that interesting.
- Packet loss? Crashing processes? Malicious actors?

PROBLEMS: TERMINOLOGY

- CAP encourages to talk about systems as CA, CP, and AP.
- What does it mean to be CA? Can't choose to not experience partitions.
- CP and AP are extremes, most useful and reliable systems give up both C and A.

LESS LEGIT CRITICISM

- "Instead of CAP, we should talk about {FLP, HAT, PACELC, delay-sensitivity, …}."
- Translation: "Everyone else should read my favorite paper."
- Deal with the world we live in. CAP won this round of marketing.
- Read CP as "favoring safety" and AP as "favoring liveness" and reason from there.

DISCUSS.