Is Distributed Consistency Scalable?

Today’s cloud systems exhibit visible inconsistencies!

- Today, very few cloud applications need consistency in a strong, interactive, real-time sense of the term.
  - There is a pervasive belief that consistency can compromise scalability / stability
  - ... rather than risk poor performance, cloud systems opt for asynchronous, potentially inconsistent behaviors.
- To compensate for inconsistency on the server side, cloud applications are expected to tolerate occasional problems and to reissue requests if a server crashes
Dangers of Inconsistency

- Inconsistency causes bugs
  - Responses often based on stale, incorrect data

- Weak or "best effort" consistency?
  - Strong security guarantees demand "right now" (real-time) consistency
  - Is it safe to trust a medical electronic-health records system or a bank that accepts "weak consistency" for better scalability?

Cloud systems embrace CAP

- A folk-theorem proposed by Eric Brewer
  - In a nutshell, you must make a choice between Consistency, or Availability during Partitioning
  - A data center that embraces CAP will often exhibit inconsistencies.

Replicated data used by cloud services will often be stale, perhaps for extended periods of time, and the cloud services won’t realize they are giving incorrect answers.
The underlying issue

- CAP is really about bandwidth!
  - Cloud systems scale services by replicating them on large numbers of nodes, and this entails replicating the data on which they operate
  - When updates occur, those replicas need to be brought up to date by applying them in the right order, everywhere
- So CAP is an assertion about the speed of data replication and reflects the assumption that scalable data replication is infeasible

Two paths to scalable data replication...

- One option is to use IP multicast: one message will reach as many destinations as you like.
  - Sounds like “replication for free”... and it is!
  - But this is the less travelled path, as we’ll see....

- A second is to run purely on TCP, the standard web services protocol for cloud computing
  - To send an update to \( n \) replicas takes \( n \) TCP sends
  - Often uses a tree of TCP connections so that the update source itself only does a few sends, maybe 2
Issue: IP multicast is broken!

- There turn out to be two problems, seen mostly in massive cloud settings
  - First, most IPMC replication protocols lack any form of flow control (unlike TCP)
  - Second, the hardware can break down with heavy use
    - IPMC filtering quality collapses, overloading receivers with unwanted messages, triggering loss
- This is why data centers rule out option 1!

---

eBay’s Five Commandments

- As described by Randy Shoup at LADIS 2008

*Thou shalt…*

1. Partition Everything
2. Use Asynchrony Everywhere
3. Automate Everything
4. Remember: Everything Fails
5. Embrace Inconsistency
Vogels at the Helm

- Werner Vogels is CTO at Amazon.com...
- His first act? **He banned reliable multicast**!*!
  - Amazon was troubled by platform instability
  - Vogels decreed: all communication via SOAP/TCP

- This was slower... but
  - **Stability matters more than speed**

* Amazon was (and remains) a heavy pub-sub user

James Hamilton’s advice

- Key to scalability is decoupling, loosest possible synchronization
- **Any synchronized mechanism is a risk**
  - His approach: create a committee
  - Anyone who wants to deploy a highly consistent mechanism needs committee approval

.... They don’t meet very often
... embodied into Azure

- Applications structured as stateless tasks
  - Azure decides when and how much to replicate them, can pull the plug as often as it likes
  - Any consistent state lives in backend servers running SQL server... but application design tools encourage developers to run locally if possible

Consistency

Consistency technologies just don’t scale!
Leading to a challenge

- A belief that inconsistency is inevitable is the common thread...
- But what if all three gurus are wrong?
  - They all reject IP multicast as an unstable protocol
  - The whole CAP logic stands on this cornerstone
- Can we fix IP multicast?

An agenda

- Fix IP multicast.
  - *Cornell & IBM solved this! Dr. Multicast [nsdi 09]*.
- Now develop a new generation of scalable IP multicast-based data replication solutions
  - Data replication at the speed of light!
- Need: flow control, reliability, stability at scale
- We’ve done some of this, using ideas that originate in the scalability work on web search!
Finally, offer strong “models”

- Make a rigorous, theoretically sound statement of what we are able to provide
  - Consistency used to offer coordination, fault-tolerance, security, real-time properties

- This “improved” technology is also much faster than today’s TCP-based one
  - People who adopt the better approach for its speed gain consistency, security, fault-tolerance

A few slides on consistency

- ... if our time-keeper permits
What exactly is consistency?

A consistent distributed system will often have many components, but users observe behavior indistinguishable from that of a single-component reference system.

Reference Model

Implementation

Why fear consistency?

- They reason this way:
  - Systems that make guarantees put those guarantees first and struggle to achieve them
  - For example, any reliability property forces a system to retransmit lost messages, use acks, etc
  - But modern computers often become unreliable as a symptom of overload... so these consistency mechanisms will make things worse, by increasing the load just when we want to ease off!
- So consistency (of any kind) is a “root cause” for meltdowns, oscillations, thrashing
**Where does it come from?**

- Transactions that update replicated data
- Atomic broadcast or other forms of reliable multicast protocols
- Distributed 2-phase locking mechanisms

**If we rule out such mechanisms...**

- Our systems become “eventually” consistent but can lag far behind reality
- Thus application developers are urged to not assume consistency and to avoid anything that will break if inconsistency occurs
A Consistency Property: Virtual Synchrony

- **Synchronous runs**: indistinguishable from non-replicated object that saw the same updates (like Paxos)
- **Virtually synchronous runs** are indistinguishable from synchronous runs

When virtual synchrony ruled...

- During the 1990’s, Isis was a big success
  - French Air Traffic Control System, New York Stock Exchange, US Navy AEGIS are some blue-chip examples that used (or still use!) Isis
  - But there were hundreds of less high-profile users

- However, it was not a huge *commercial* success
  - Focus was on server replication and in those days, few companies had big server pools
Cornell is reinventing Isis

- Our new Isis$^2$ platform is starting to limp along; it offers scalable virtual synchrony in cloud-scale settings
- But the work, up to now, was done by one person (me) and far more is needed before the system can be widely deployed

Challenge to OSD?

- Take all of this to warp factor 2!
  - Create a new program aimed at building data center platform technologies that provide high assurance for cloud computing
  - Such a technology should consist of a theory... a useful prototype... and real applications
  - It would be a very expensive undertaking. How much did it cost for Microsoft to create Azure?
- Benefits? Enormous.
Discussion questions

1. Proposal rejects the CAP “religion”. But are there reasons (other than the rejection of IPMC) that data centers might be unstable if they attempt to offer consistency? (e.g. convoys, spooky self-synchronization...?)

2. Do applications really need strong consistency? Maybe consistency requirements simply reflect poor application design choices?

3. The proposal asserted a connection between consistency and data center security. Is this assertion valid?