Cloud Computing

The Three Taxes:
1. Complexity
2. Fragility
3. Vulnerability
Systems can fail in catastrophic ways leading to death or tremendous financial loss. Although there are many potential causes including physical failure, human error, and environmental factors, **design errors are increasingly becoming the most serious culprit**.

Key Computer Science Problems

Reliable Consensus

- Generals Problem (no fixed length protocol exists to guarantee a reliable solution in an environment where messages can get lost)
- Slow Node vs. Link Failure Indistinguishability. I.e. what can one side of a failed link assume about a partner or cohort on the other side?

FLP Result

- Impossibility of Distributed Consensus with One Faulty Process

Key Idea:

- Don’t depend on processes to provide *liveness*, use a new kind of link
Problem: Event Ordering is Hard

- In a distributed system over a general network we can’t tell if event at process R happened before event at process Q, unless P caused R in some way.

- Causal Trees provide this guarantee when they are stable.

- Dynamic Causal Trees provide guarantees through failure & healing, iff you have AIT on each link.

- Needs Atomic Information Transfer (AIT) in the Link.
Problem: Consensus is Hard

- Failure detectors have failed to solve the problem

- 2PC (Fail-Stop)
  - Vulnerable to coordinator failure (no safety proof)
  - 3PC vulnerable to network partitions (no liveness proof)

- Paxos (Fail-Recover)
  - Robust Algorithm but hard to understand & get right.
  - Causal Trees make roles robust, easier to understand & verify
Why? Because The Network is Flaky!

- App developers believe the network is the problem
  - Networks drop, delay, duplicate & reorder packets

- Networking people believe the apps are the problem
  - The network end to end principle: Apps should retry to distinguish between delays & drops … but … retries* ruin TCP’s ordering guarantees

- Both are incorrect. Solution requires a simple, but fresh perspective

* Application retries (i.e. opening a new socket)
Datacenter Failures
Cascade

Interdependent failures
Reconstruction storms
Timeout storms
Gossip storms
Cascade failures

Switches are **DReDDful**
They *Drop, Reorder, Delay* and *Duplicate* Packets
It’s Time to Simplify

Delta
Amazon
Google
Apple
Netflix
Paypal
…
The Big Idea

World Wide Web

Key Idea:
2 Simple Sets of Rules

Tim Berners Lee

Document Language (html)

ONE WAY LINKS

Connection Protocol (http)

Mere mortals can now get their computers to talk to each other

Cloud Computing

Earth Computing

Key Idea:
2 Simple Sets of Rules

Graph Language (gvml)

TWO WAY LINKS

Connection Protocol (eclp)

Mere mortals can now manage their infrastructures

Earth Computing
### CAS

- **Atomic Instruction**

  **Key Idea:**
  - **Lock-Free data structures**

  ```
  {While (CAS(oldvalue,newvalue, ) != new value}
  ```

### AIT

- **Atomic Information**

  **Key Idea:**
  - **Recoverable Atomic Tokens**

  ```
  {Transfer (AIT(tokenID,Notify=NO, ) != Continue}
  ```

---

**Distributed Systems Primitives**

**Concurrent Safety**
- Non-Blocking

**New Concurrency Libraries**
- Shared Memory
- Atomic RMW

**Deterministic, In-Order**
- Reversible Token

**Deterministic Recoverability**
- Durable Indivisible Property
Simpler Wiring: N2N, Switchless

Today's Networking: Servers & Switches

EARTH Computing: Cells & Links

Servers, Any to Any (IP) addressing

C2C Lattice of Cells & Links
Fundamentally Simpler

Today: Internal Segregation Firewalls  EARTH: Dynamic Confinement Domains

The Datacenter Today  The Datacenter Simplified
Split infrastructure into:

**Cloud** datacenter accessed by untrusted legacy protocols

**Earth** dynamic, resilient, *programmable* topologies

**Core** where data is immutable, secure, protected, & resilient to perturbations *(failures, disasters, attacks)*
The Big Idea

Cloudplane

Outside World

EarthCore
Logical Foundation for Resilience

Fabric
EARTH Computing Link Protocol (ECLP)

- Events: Replaces Heartbeats, Timeouts
- Addresses the Common Knowledge* Problem

Composable Presence Management
Composable Presence Management
Demo
Two Generals Problem
Example Use Cases

Two Phase Commit

Paxos

Deterministic Link Reversal Algorithms

Link Reversal

Figure 1 - The two-phase commit protocol
Example Use Cases

- **Two-phase commit** The prepare phase is asking if the receiving agent is ready to accept the token. This serves two purposes: communication liveness and agent readiness. Links provide the communication liveness test, and we can avoid blocking on agent ready, by having the link store the token on the receiving half of the link. If there is a failure, both sides know; and both sides know what to do next.

- **Paxos** “Agents may fail by stopping, and may restart. Since all agents may fail after a value is chosen and then restart, a solution is impossible unless some information can be remembered by an agent that has failed and restarted”. The assumption is when a node has failed and restarted, it can’t remember the state it needs to recover. With AIT, the other half of the link can tell it the state to recover from.

- **Reliable tree generation** Binary link reversal algorithms work by reversing the directions of some edges. Transforming an arbitrary directed acyclic input graph into an output graph with at least one route from each node to a special destination node. The resulting graph can thus be used to route messages in a loop-free manner. Links store the direction of the arrow (head and tail); AIT facilitates the atomic swap of the arrow’s tail and head to maintain loop-free routes during failure and recovery.
Common Knowledge

- **Unknown**: \( P \) is a fact but the group cannot deduce it.

- **Distributed Knowledge**: \( D_G^p \) by combining all the knowledge of the group, we can know \( P \).

- **Someone knows**: \( S_G^p \) some member of the group \( G \) knows \( P \).

- **Everyone knows that everyone knows**: \( E_G^p \) all members of \( G \) know \( P \).

- **The Hierarchy of Knowledge**

- **Common Knowledge**: for all \( k \geq 1 \), \( C_G^P \) everyone has full situational knowledge.

Implementation On Smart NIC’s
New Distributed Systems Foundation

TRAPHs (Tree-gRAPHs)

- Simple Provisioning, Confinement, Elasticity, Migration, Failover
Demo Simulator
Questions?

- Don’t Make Datacenter Look Like the Internet
  - Simpler to Configure/Reconfigure
  - More Resilient to all perturbations
  - Easier to Secure

- Key Ideas
  - RAFE: Reliable Address-Free Ethernet
  - Replace switches with cell to cell links
  - Don’t rely on blueprints, discover wiring
  - Event driven => No network timeouts
  - Keep state in links for recovery
  - No VLANs, no network-layer encryption
  - Scalable design - local only view
  - NO IP; service addressing
  - Self recovering from link & server failures
  - RAFE is a discovery process rather than a configuration process