Transactions, Dogma, & Karma

The More I Know about Transactions, the More Confused I Get…

Pat Helland

HPTS Oct 2011

Introduction

1970s to mid-1990s	**Pat Says:** *"There's ACID transactions and there's EVIL"* We LOVE Two-Phase-Commit
mid-1990s to 2011	**Pat Says:** *"Natural Selection Will Handle Applications Depending on 2PC"* Two-Phase-Commit Is Too Brittle for Practical Use… Deal with It
2011 to ???	**Pat Says:** *"Two Phase Commit Is Great… Sometimes…"* Some New Cases Offer Great Cost/Benefit Tradeoffs

Summary:

Never Believe a Damn Thing I Ever Say!

Including This Talk…

Life on ACID…
the 1970s and 1980s Were Great!

- I grew up worshipping at the altar of ACID transactions…
  - Those smart old guys at IBM and Tandem made that shit up…
  - They were working down the hall at Tandem on Relational Databases
  - That relational stuff REALLY counted on transactions…
  - ACID transactions were like Atlas holding up the World…

In the Early 1990s, the Internet Came Along

They Weren't Using Transactions… They Weren't Using Locks!

Obviously, This Internet Thing Was Doomed to Failure, Wallowing in It's Inconsistent Data

Those Luddites Didn't CARE They Were Inconsistent…

They Just Stuffed Data Out There and Did What They Wanted !!!

Brittleness, Cynicism, and Apostacy

Mid-1990s: Distributed Transactions Are Brittle in Practice

If Any One Machine Is Down, the Transaction May Be Locked Up

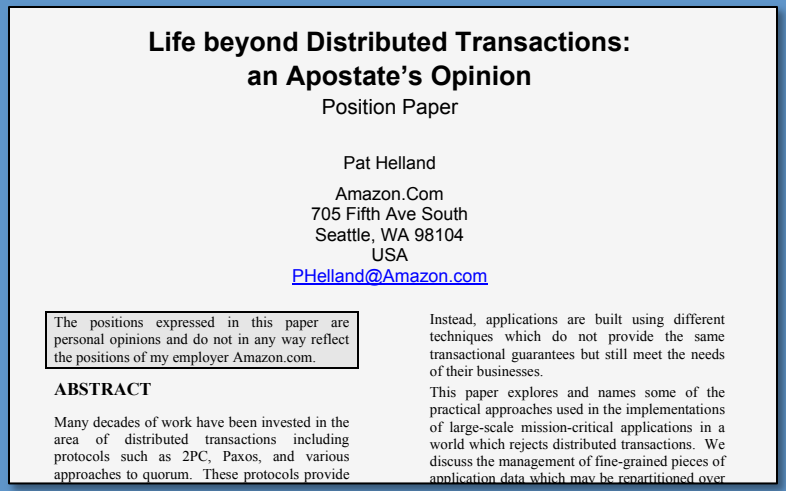
If One Transaction Is Locked Up, Other's May Pile Up in a Jam, Too…

Lots of Real-World Applications Failed If Dependent on Distribute 2PC

I Said "Don't Use Distributed Transactions!!!"

They're Fragile… It Will Hurt!

Do the Following Annoying Workflows to Get Around Your Desire for Transactions…



Life beyond Distributed Transactions: an Apostate's Opinion

Position Paper

Pat Helland

Amazon.Com
705 Fifth Ave South
Seattle, WA 98104
USA
PHelland@Amazon.com

The positions expressed in this paper are personal opinions and do not in any way reflect the positions of my employer Amazon.com.

ABSTRACT

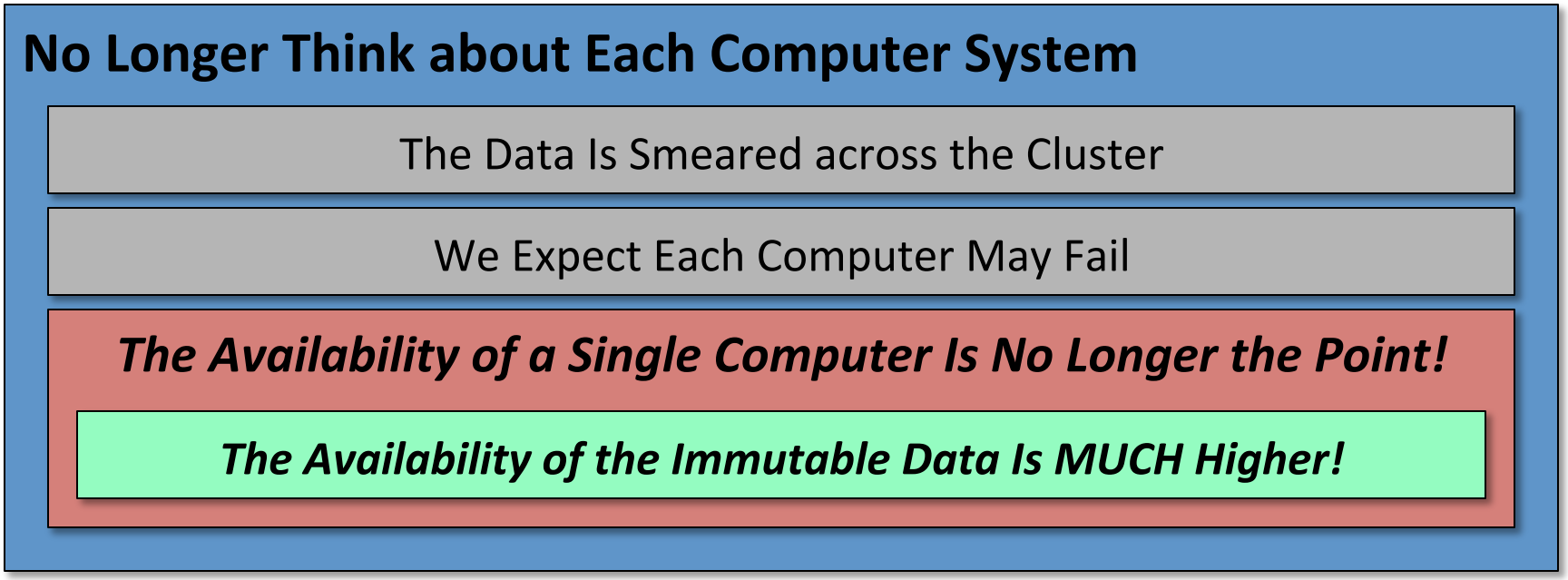
Many decades of work have been invested in the area of distributed transactions including protocols such as 2PC, Paxos, and various approaches to quorum. These protocols provide

Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.

This paper explores and names some of the practical approaches used in the implementations of large-scale mission-critical applications in a world which rejects distributed transactions. We discuss the management of fine-grained pieces of application data which may be repartitioned over

Immutability and Replication in Massive Datastores

- Recently, we see MASSIVE datastores with highly available immutable data
  - GFS (Google): Massive storage of immutable files
  - Cosmos (Microsoft Bing): Massive storage of immutable byte streams
- Triple redundant automatically replication storage of chunks (extents)
  - The system automatically keeps at least three replicas
  - The data is very highly available



Deltas and Immutable Datasets

- BigTable, HBase, and others allow changeable data by key
  - Updates are captured as deltas which are written as immutable bytestreams
  - The base set of records is captured as immutable bytestreams
  - Deltas are captured as immutable bytestreams
- We have changeable records built on top of immutable data
  - Pretty similar to what databases have done for a while
- We have extremely highly available and updateable records!
  - Because the underlying bytestreams are highly available, we can make highly available updateable key-value records!

Percolator, Cosmos, and More…

- Percolator: Google's plumbing for the Caffeine web-index generation
  - Supports full transactional update across ANY record across tens/hundreds of petabytes across tens of thousands of computers (in one data center)
- Cosmos: Evolving to support transactional random update of records
  - Will support full transactional update across ANY record across tens/hundreds of petabytes across tens of thousands of computers (in one data center)
- The state of the Two Phase Commit of each transaction is stored within the records comprising the transaction
  - The state of a participating record is stored in the record (pointing to a master)
  - The outcome of the transaction is captured in a "master" record
  - The services to advance the transaction are restartable in seconds/minutes
- Transactional changes are highly available
  - The state is triple-replicated as records
  - The service to advance the state is quickly restartable
  - The transaction is not brittle!!!

Dogma and Karma

I Have a Track Record for Spouting Dogma about Transactions

Transactions Have a Karma that Keeps Running over My Dogma…

You Should Never Believe a Damn Thing I Ever Say!

Takeaways

- Transactions (and distributed transactions) lower the app complexity barrier
  - It is much easier to write an application with transactions to handle failures
  - It is much easier to write a composite app spanning many components with Txs.
  - Some of us (including me) had strong opinions that transactions are *essential*
- The availability of the application using distributed transactions has been (pretty much) the intersection of the availability of the servers touched
  - This has led to many transactional applications that work well in test and are a major problem when deployed in the real world
  - Some of us (including me) had strong opinions that distributed transactions should be associated with winning the Darwin Award
- New scalable systems are emerging which represent transactions as state in records made highly available over tens of thousands of computers
  - Spanning tens of thousands of computers but in one datacenter
  - Highly available record state means highly available distributed transactions
- Emerging cases: the benefit of large 2-phase commit may exceed the cost
  - The availability of 2-Phase-Commit (across H/A records in a datacenter is GREAT)
  - The application is much easier
  - The participating applications must trust the infrastructure.