

Michael Stiefel

Reliable Software, Inc.

[www.reliablesoftware.com](http://www.reliablesoftware.com)

# Do Relational Databases Make Sense in the Cloud?

Session Code: AR. 12

# Critical Questions

- How does the Internet affect modelling cloud data stores?
- What is the best way to think about data in the cloud?

# Some Facts of Life...

# Relational Model

- A query operation on a relation (table) produces another relation (table).
- Based on the relational algebra and calculus, a query engine can produce provably correct results.

# Consistency Required

- Transactional consistency
  - Relational algebra and calculus do not specify insert, update or delete.
  - Non clustered indices consistent with data
- Design consistency
  - Denormalized data must be kept consistent
  - Lossless join decompositions

# Latency Exists

- Speed of light in fiber optic cable: 124,000 miles per second
- Ideal ping Japan to Boston takes 100 ms.
- Fetch 10 images for a web site: 1 second
- Ignores Latency of the operation

# Bandwidth is Limited

- Shannon's Law:  $C = B \log_2 (1 + S / N)$
- Capacity = bit / second
- Bandwidth (hertz)
- S/N \* 5 to double capacity given bandwidth

# Latency is Not Bandwidth

- Size of the shovel vs. how fast you can shovel
- Infinite shovel capacity(bandwidth) is limited by how fast one can shovel (latency).

# Great Bandwidth Terrible Latency

Buy a two terabyte disk drive

Drive with it from Boston to New York

# Implications...

# Expensive to Move Data

- Computational Power Gets Cheaper Faster than Network Bandwidth
- Cheaper to compute where data is instead of moving it
  - *Distributed Computing Economics Jim Gray*

# Connectivity is Not Always Available

- Cell phone
- Data Center Outages
- Equipment Upgrades
- Data redundancy to improve reliability

# Waiting for Data Slows Computation

# Partition Data To Improve Performance

- Data Naturally Lives In Multiple Places
- Data Close to Customer
- Bandwidth, Latency for Throughput
- Human Interaction

# Performance Scenarios

- Large Number of Users
- Geographic Distribution

# Classic Ways to Handle Partitioning

# Distributed Objects

- Distributed Objects Fail
  - Separate Address Space
  - Disparate Lifetimes
  - Location is Not Transparent
- RPC Model Fails
- Cannot Hide Network

# Distributed Transactions

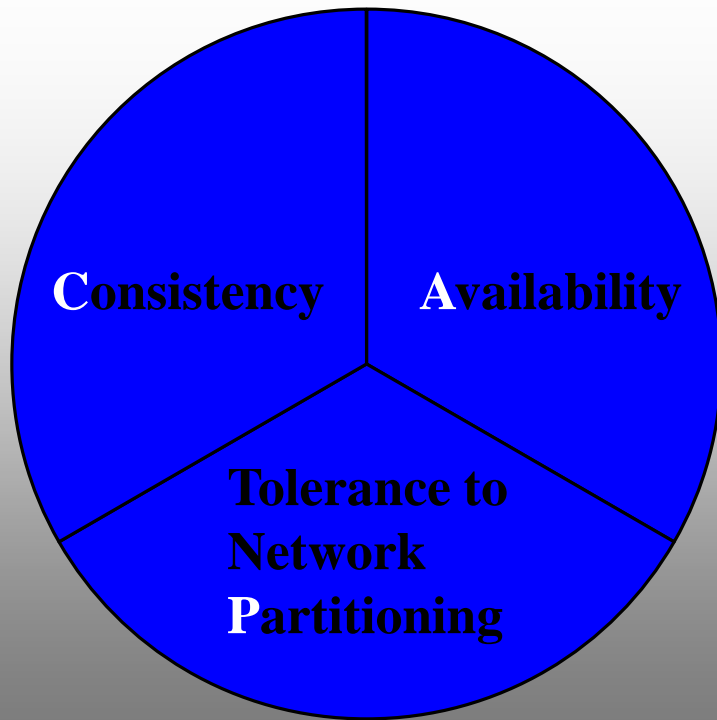
- Relational Model scales single node/ cluster
  - Complexity of relations
  - Query plans with hundreds of options which query analyzer evaluates at runtime
  - Normalization
  - ACID Transactions
- Quick hardware scale up difficult
- Two Phase Commit works with infinite time

# Economics Dictate Scale Out Not Up

- Cheap, commodity hardware argues for spreading load across multiple servers
- How do you distribute data among several databases?
- How do you achieve consistency without distributed transactions?

# To Scale a Distributed System Design Focuses on Data, Not Just Computation

# CAP Theorem



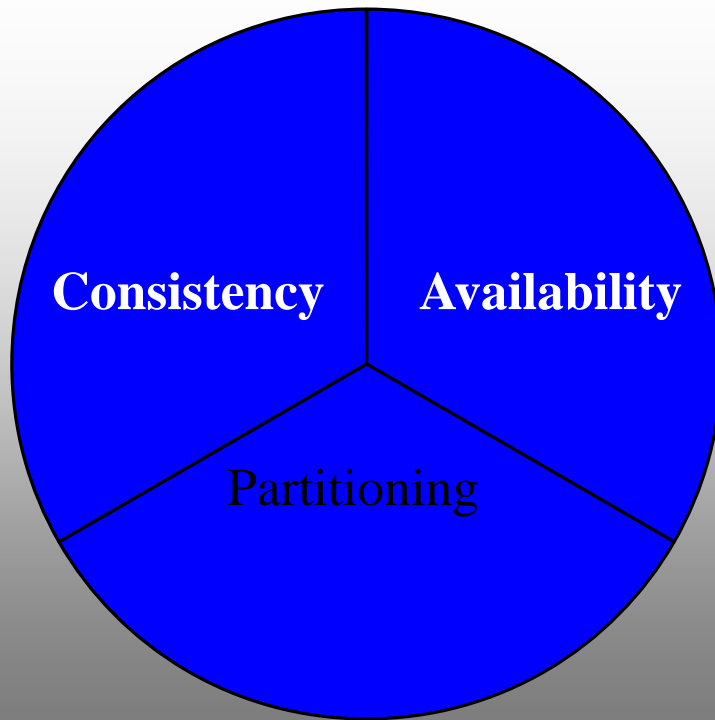
Can Have Any Two

Eric Brewer

UC Berkeley, Founder Inktomi

<http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>

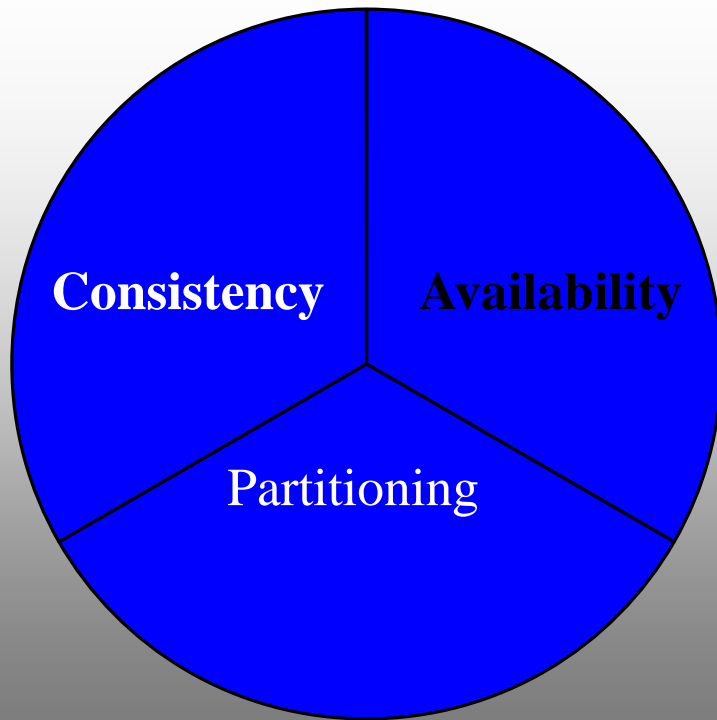
# Consistency and Availability



Single site Database  
Database Cluster  
LDAP

Two phase commit  
Validate Cache

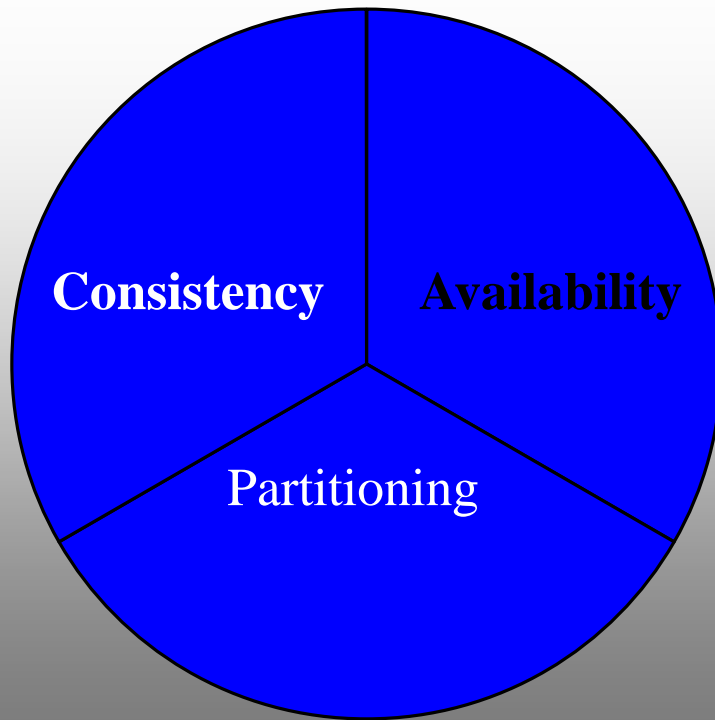
# Consistency and Partitioning



Distributed Database  
Distributed Locking

Pessimistic Locking  
Minority Partitions  
invalid

# Availability and Partitioning



Forfeit Consistency

Google Big Table

Amazon Simple DB

Optimistic Locking

Can Denormalize

# Data Storage Has Two Flavors in Azure

 Windows Live™

 Microsoft Office Live

 Microsoft Exchange Online

 Microsoft SharePoint Online

 Microsoft Dynamics CRM Online

## Azure™ Services Platform

 Live Services

 Microsoft .NET Services

 Microsoft SQL Services

 Microsoft SharePoint Services

 Microsoft Dynamics CRM Services

 Storage Services



Windows® Azure™

# SQL Azure

- SQL Server in the sky
  - Tables, Stored Procedures, Triggers, Constraints Views, Indices
  - Uses TDS (Tabular Data Stream) Protocol
- Change connection string to get to another SQL Server

# Windows Azure Storage Services

- Tables of key/value pairs for highly scalable structured storage
- CRUD operations
- No FK relations, Joins, Constraints, Schemas
- Partition / Tables / Entities / Properties
- Entity has Unique Row Key

# Azure Storage Services

- Fit well with tens or hundreds of commodity servers
- Better mapping with objects than ORM
- No integrity constraints
- No joined queries
- No standards among vendors (lock in)
- Will Microsoft have query limits?
  - Amazon no query longer than 5 seconds
  - Google no more than 1000 items returned

# Car Table

Key	Attribute 1	Attribute 2	Attribute 3	Attribute 4	
1	Make: BMW	Color: Grey	Year 2003		
2	Make: Nissan	Color : Red Yellow	Year: 2005	Transmission: Easytronic	
3	Plane: Boeing	Color: Blue		Engine: Rolls Royce	

# Availability or Consistency ?

# What is the Cost of an Apology?

- Amazon
- Airline reservations
- Stock Trades
- Deposit of a Bank Check
- Deleting a photo from Flickr or Facebook

# Sometimes the cost is too high

- Authentication
  - SAML tokens expire
- Launching a nuclear weapon

# Businesses Apologize Anyway

- Vendor drops the last crystal vase
- Check bounces
- Double-entry bookkeeping requires compensation
  - at least 13<sup>th</sup> century
- Eventually make consistent

# Software State $\neq$ State of the World

- Software approximates the state of the world
- Best guess possible
- Could be wrong
- Other computers might disagree

# How consistent?

- Business Decision
- What is the cost to get it absolutely right?
- What is the cost of lost business?
- Computers can remember their guesses
- Can replicate to share guesses
- May be cheaper to forget, and reconcile later

# Design For Eventual Consistency

- Decouple unrelated application functionality
- Determine data partition strategy
- No distributed transactions
- Asynchronous processing

# Partition Your Data to Scale?

- No Partitioning
- Natural Partitioning
- Partitioning for Availability

# Partition Basics

- Transactions only apply on per object basis
- Identify objects by unique key (partition key / row key)
- Objects can move when repartitioning
- Cannot assume two objects remain on the same machine
- Data might go offline

# How To Partition

- **Shard by Row**
  - Topic
  - Region
  - Duplicate in several places
    - Flickr stores comments in commenter and commentee records
- **Partition by table**
  - According to Function or Topic
  - Can maintain normalization of original, but copies might exist

# Eventual Consistency

- Different computations might come to different conclusions
- Define message based workflows for ultimate reconciliation and replication of results

# Querying Gets Interesting

- Secondary indices might be on different machines from the data
- No transactional queries
- No transactional updates of alternate keys and data
- Can query stale data

# Some Data is Easy to Cache

- Immutable Data
  - Prices on December 24, 2008
  - Wall Street Journal April 1, 2008
  - Bank Statement of a given Data
  - Identified By Some Identifier
- Version Independent Data
  - Get me today's newspaper
  - Get me the last price list
- Same rules for Metadata
- Timestamp or version data

# Conclusions

- Partitioning Changes the Data Model
- Design for eventual consistency
- No need for massive scalability, relational model works
- But what happens if things change rapidly?
- Azure gives you both options

# Evaluation form

Vul je evaluatieformulier in en maak kans op een van de prachtige prijzen!!

Fill out your evaluation form and win one of the great prizes!!

**Session Code: AR 12**