DTCC2011

Scaling your Database

Presented by:
Cory Isaacson, CEO
CodeFutures Corporation
http://www.dbshards.com

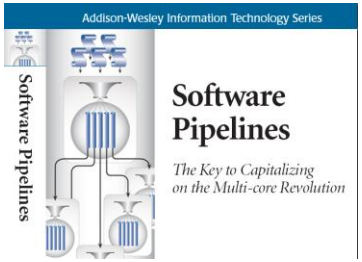
In partnership with:
UnGeo（安久科技）
www.ungeo.com

Fall 2010

Introduction

DTCC2011

- Who I am
  - Cory Isaacson, CEO of CodeFutures
  - Providers of dbShards
  - Author of Software Pipelines



dbShards

- Partnerships:

  UnGeo（安久科技）
  - Provider of cloud computing in China
- Leaders in scalability, performance, high-availability and database solutions…
  - …based on real-world experience with dozens of high-performance applications
  - …social networking, gaming, data collection, mobile, analytics
- Objective is to provide useful experience you can apply to scaling (and managing) your database tier…
  - …especially for high volume applications

Challenges of high-volume applications

- Need to scale
  - Must handle large growth in users, transaction volume
- Hardware failures
  - Partial network outages
  - Server failures
  - Disk failures
- Resources are limited
  - CPU
  - I/O Rates

DTCC2011

Scaling your application

DTCC2011

- Scaling Load Balancers is easy
  - Stateless routing to app server
  - Can add redundant Load Balancers if needed
  - Round-robin or intelligent routing for larger sites
  - If one goes down…
    - …failover to another
- Scaling Application Servers is easy
  - Stateless
  - Add or remove servers as need dictates
  - If one goes down…
    - …failover to another

Scaling your application

DTCC2011

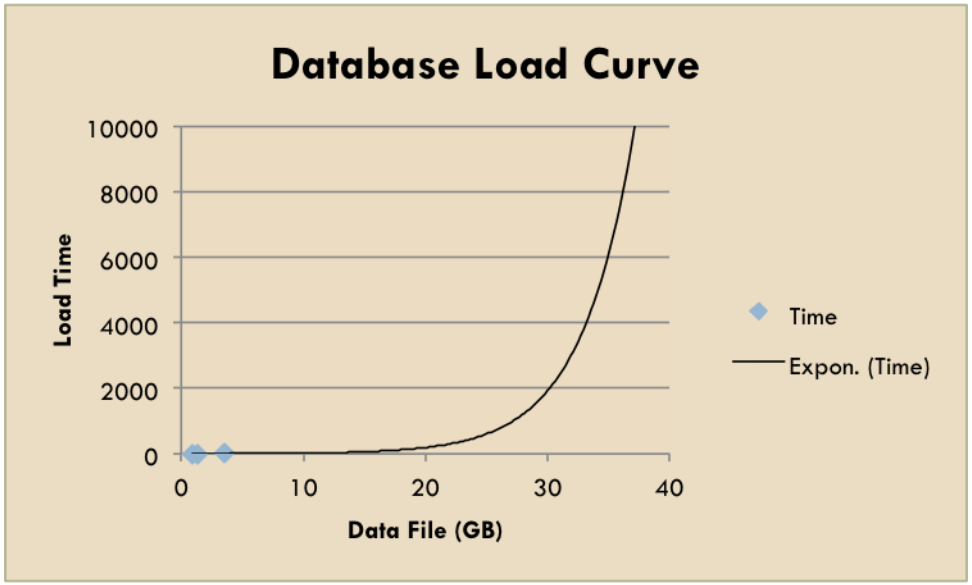
- Scaling the Database tier is hard
  - "Stateful" by definition (and necessity…)
  - Large, integrated data sets…
    - 10s of Gigabytes to Terabytes (or more…)
    - Difficult to move, reload
  - I/O dependent…
    - …disk speeds are limited
  - If one goes down…
    - …ouch!
- Databases form the "last mile" of true application scalability
  - Initially simple optimization produces the best result
  - Implement a follow-on scalability strategy for long-term performance goals…
    - …plus a high-availability strategy is a must

DTCC2011

More Database scalability challenges

- Databases have many other challenges that limit scalability
  - ACID transaction compliance…
    - …especially Consistency
    - …user contention
  - Operational challenges
    - Failover
      - Planned, unplanned
    - Maintenance
      - Index rebuild
      - Restore
      - Space reclamation
    - Lifecycle
      - Reliable Backup/Restore
      - Monitoring
      - Application Updates
      - Management

Database slowdown is not linear



GB	Load Time (Min)
.9	1
1.3	2.5
3.5	11.7
39.0	10 days…

Challenges apply to all types of databases

- Traditional RDBMS (MySQL, Postgres, Oracle…)
  - I/O bound
  - Multi-user, lock contention
  - High-availability
  - Lifecycle management
- In-memory Databases (NoSQL, Caching, Specialty…)
  - Reliability
  - Limits of a single server
    - …and a single thread
  - Data dumps to disk
  - High-availability
  - Lifecycle Management
- No matter what the technology, big databases are hard to manage…
  - …scaling is a real challenge
  - …degradation from growth in size and volume is a certainty

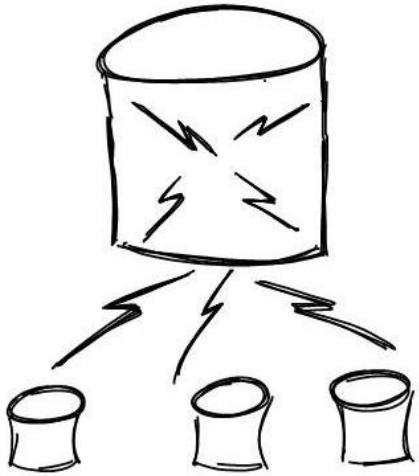
The Laws of Databases

DTCC2011

- Law #1: Small Databases are fast…
- Law #2: Big Databases are slow…
- Law #3: Keep databases small

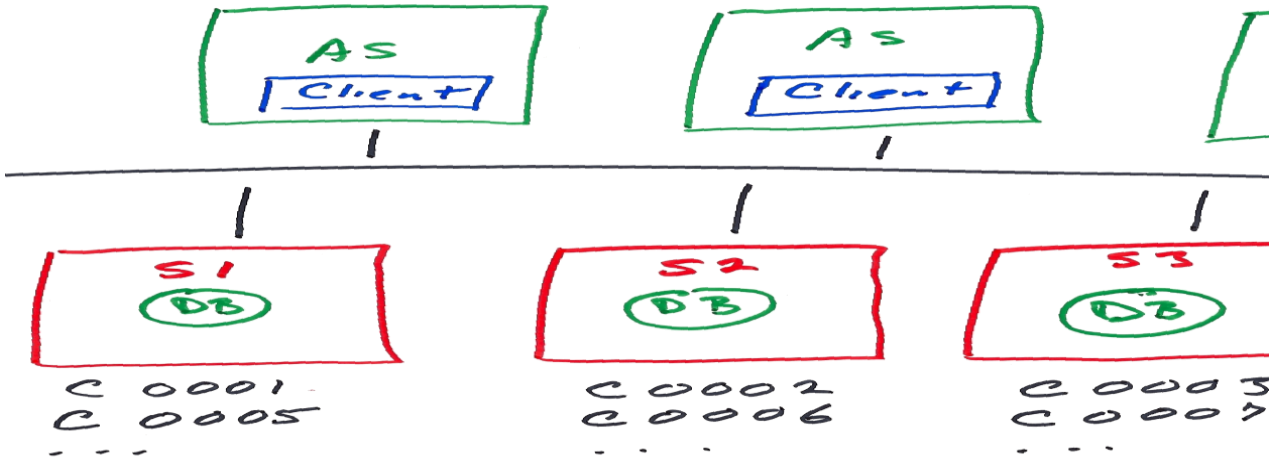
What is the answer?

- Database sharding is the only effective method for achieving scale, elasticity, reliability and easy management…
  - …regardless of your database technology



What is Database Sharding?

- "Horizontal partitioning is a database design principle whereby rows of a database table are held separately... Each partition forms part of a shard, which may in turn be located on a separate database server or physical location." *Wikipedia*



The key to Database Sharding...

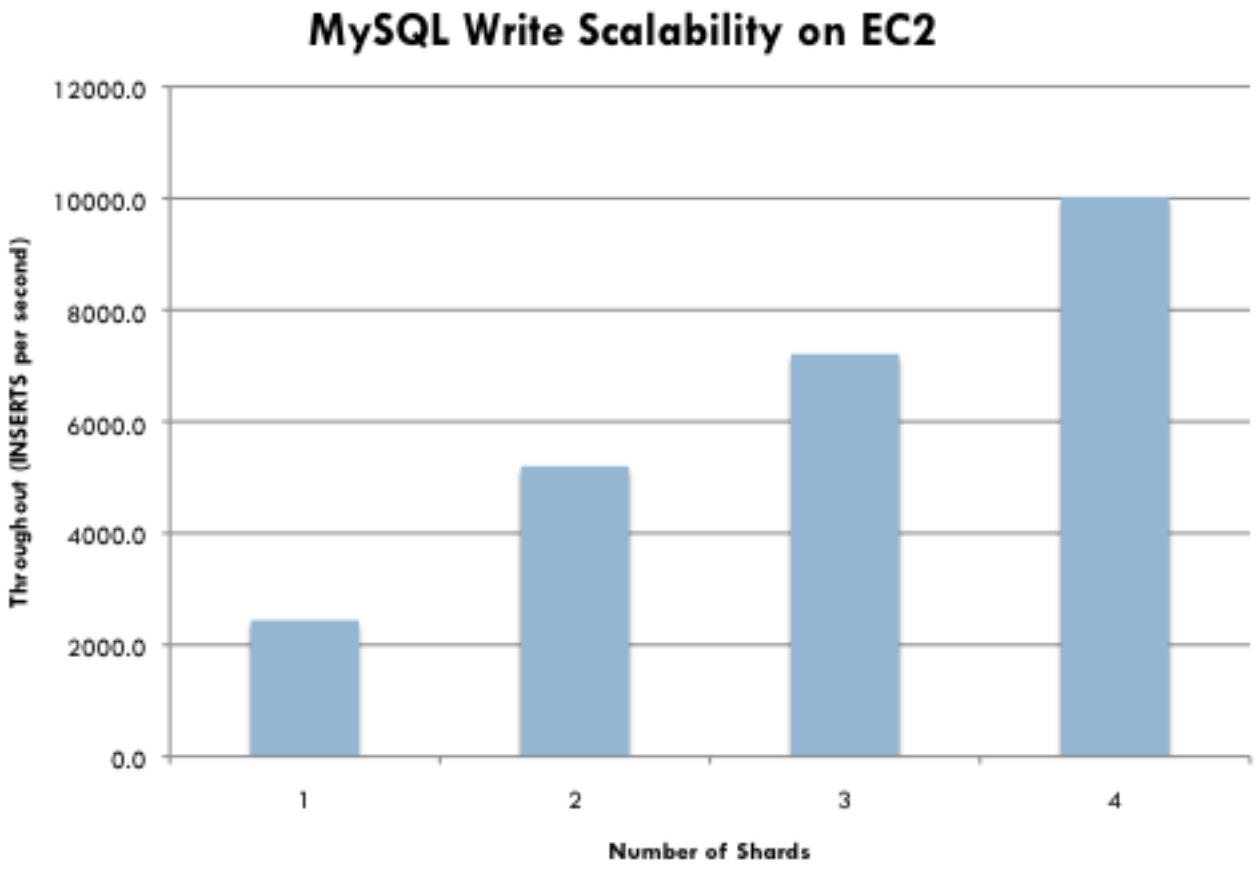
DTCC2011

Share Nothing.

DTCC2011

DTCC2011

Database Sharding… the results



Why does Database Sharding work?

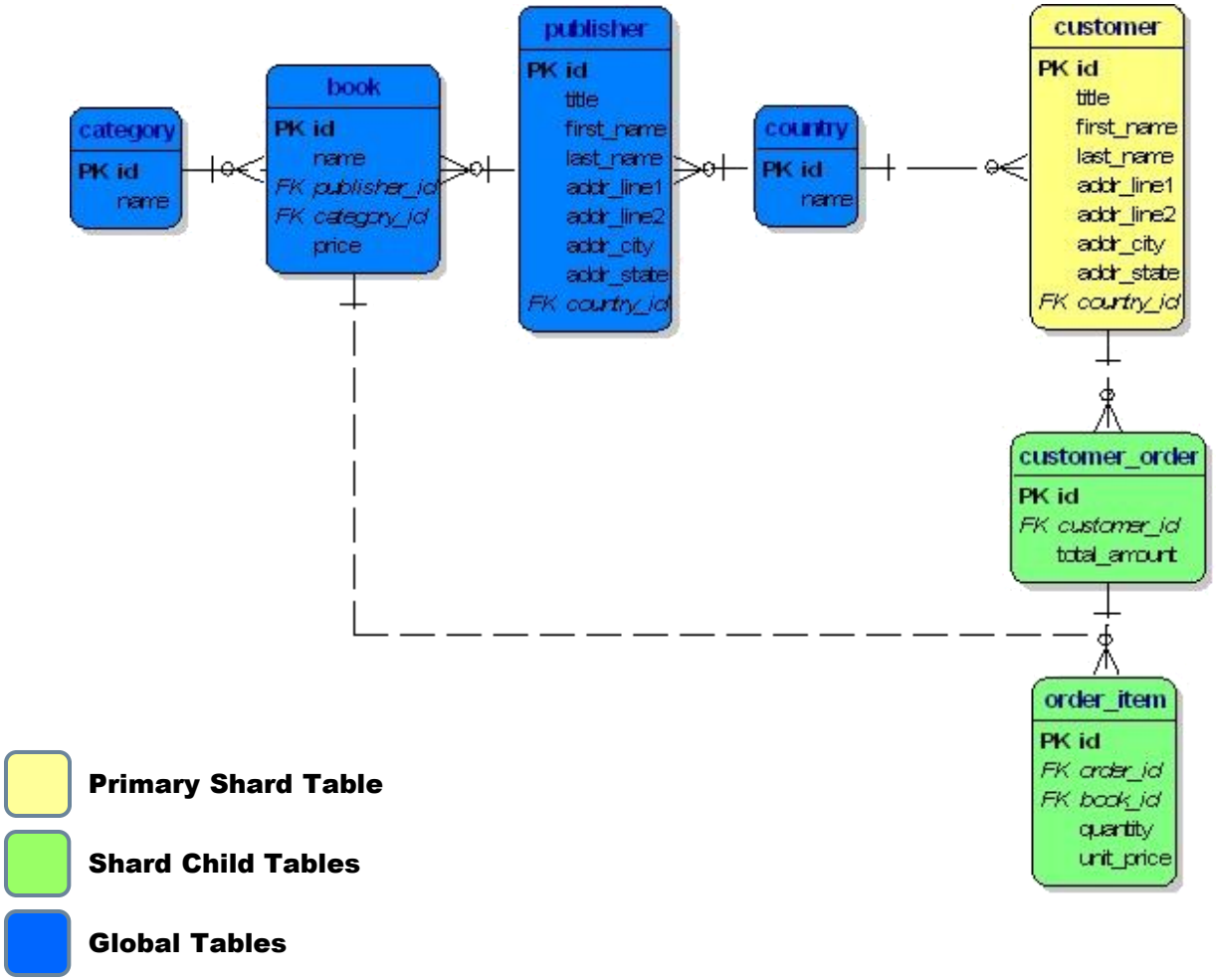
- Maximize CPU/Memory per database instance…
  - …as compared to database size
- No contention between servers
  - Locking, disk, memory, CPU
- Allows for intelligent parallel processing…
  - …Go Fish queries across shards
- Keep CPUs *busy* and *productive*

DTCC2011

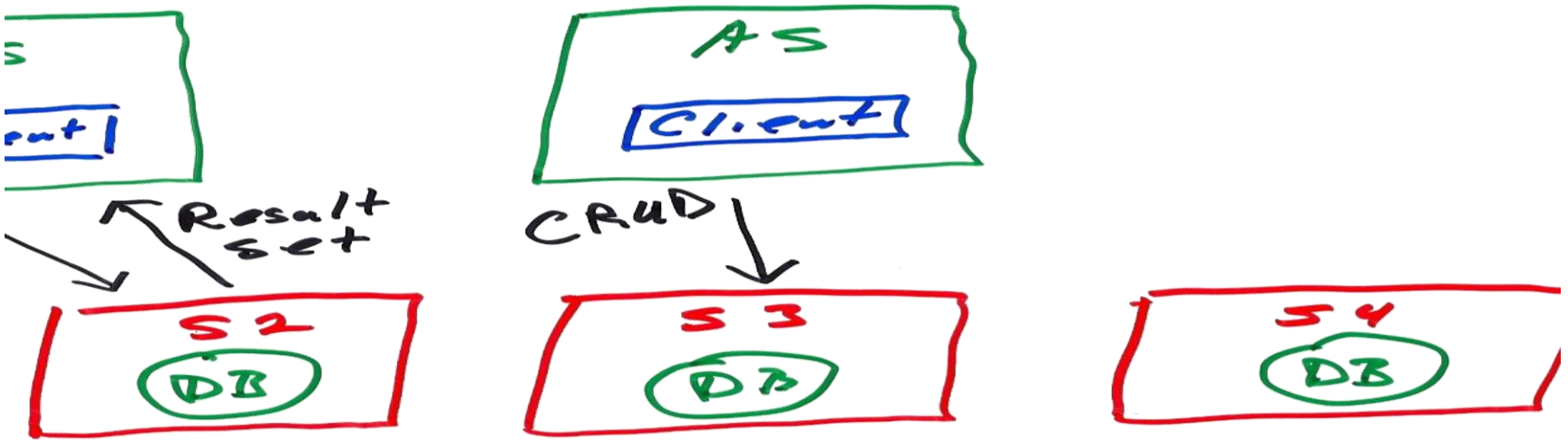
Black box vs. Relational Sharding

- Both utilize sharding on a data key…
  - …typically modulus on a value or consistent hash
- Black box sharding is automatic…
  - …attempts to evenly shard across all available servers
  - …no developer visibility or control
  - …can work acceptably for simple, non-indexed NoSQL data stores
  - …easily supports single-row/object results
- Relational sharding is defined by the developer…
  - …selective sharding of large data
  - …data from related tables remain together
  - …explicit developer control and visibility
  - …tunable as the database grows and matures
  - …more efficient for result sets and searchable queries

How Relational Sharding works



How Relathional Sharding works



DTCC2011

More on Cross-Shard result sets…

- Black Box approach requires “scatter gather” for multi-row/object result set…
  - …common with NoSQL engines
  - …forces use of denormalized lists
  - …must be maintained by developer/application code
  - …Map Reduce processing helps with this (non-realtime)
- Relational Sharding provides access to meaningful result sets of related data…
  - …aggregation, sort easier to perform
  - …logical search operations more natural

What about High-Availability?

- Can you afford to take your databases offline:
  - …for scheduled maintenance?
  - …for unplanned failure?
  - …can you accept some lost transactions?
- By definition Database Sharding adds failure points to the data tier
- A proven High-Availability strategy is a must…
  - …system outages
  - …planned maintenance

DTCC2011

DTCC2011

DTCC2011

Database Sharding…elastic shards

- Expand the number of shards…
  - …divide a single shard into N new shards
- Contract the number of shards…
  - …consolidate N shards into a single shard
- Due to large data sizes, this takes time…
  - …regardless of data architecture
- Requires High-Availability to ensure no downtime…
  - …perform scaling on live replica

Database Sharding…the future

- Ability to leverage proven database engines…
  - …SQL
  - …NoSQL
  - …Caching
- Allow developers to select the best database engine for a given set of application requirements…
  - …seamless context-switching within the application
  - …use the API of choice
- Improved management…
  - …monitoring
  - …configuration “on-the-fly”
  - …dynamic elastic shards based on demand

Database Sharding summary

DTCC2011

- Database Sharding is the most effective tool for scaling your database tier…
  - …spread your database across many servers
- Relational Sharding is strongest approach…
  - …keep related data together
  - …developer controlled, can be optimized over time
- Ensure your High-Availability is proven and bulletproof…
  - …must support failure and maintenance

Questions/Answers

DTCC2011

Cory Isaacson
CodeFutures Corporation
cory.isaacson@codefutures.com
http://www.dbshards.com

Gary Jia
UnGeo（安久科技）
gary@ungeo.net
http://www.ungeo.net