Oracle Database Native Sharding: a Customer Perspective

John Kanagaraj, Sr. Member of Technical Staff, PayPal Core Data Platform

Speaker Qualifications

Currently Sr. Database/Data Architect @ PayPal

Has been working with Oracle Databases and UNIX for 28 years

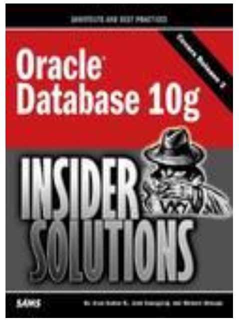
Working on various NoSQL technologies for the past 3 years

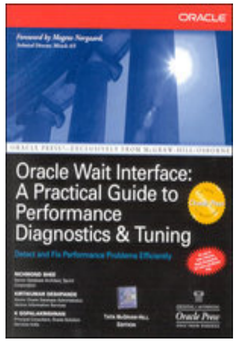
Has worked on many Sharded applications – Both Oracle and NoSQL

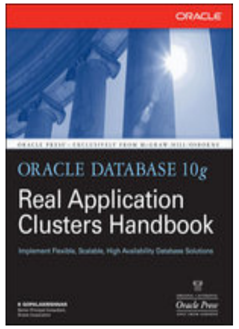
Author, Technical editor, Oracle ACE, Frequent speaker

Loves to mentor new speakers and authors!

http://www.linkedin.com/in/johnkanagaraj


Oracle Database 10g
INSIDER SOLUTIONS


Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning


ORACLE DATABASE 10g
Real Application Clusters Handbook

Sharding at PayPal

- PayPal runs 800+ Oracle databases
  - Oracle is used for all site-critical applications; NoSQL is onboarding
  - Core *transactional* systems on Oracle
- Sharding based scaling for selected use cases
  - Custom, fixed # sharding for high scale workloads started in 2009
  - Maintaining custom sharding code and infrastructure challenging
  - NoSQL provides sharding out of box
  - Evolving standard pattern for sharded access (client, connection pooling, routing, logical-to-physical mapping, etc.)
- Oracle Sharding will allow us to more easily and effectively scale our mission-critical transactional databases

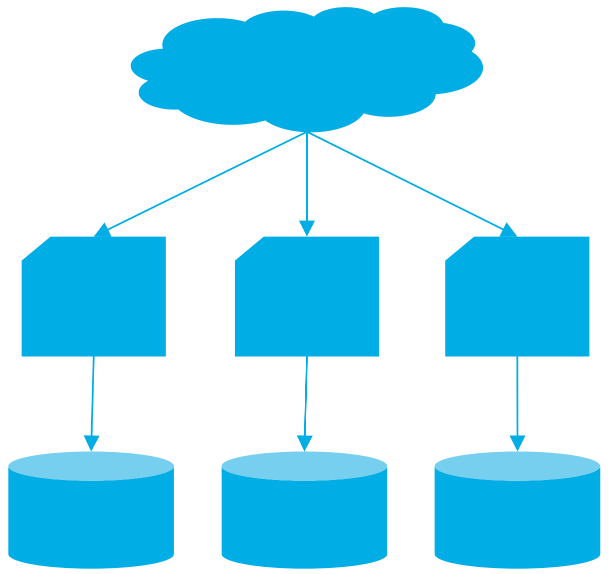
Challenges at Scale

- Pushing the limits
  - Connections
  - Memory
  - Interconnect
  - CPU
  - DDL on busy tables
  - RAC reconfiguration
  - Redo rate
  - I/O latencies
  - SAN Storage limits
  - Replication latencies

- Solutions
  - Separate reads for RO scaleout (ADG/GG)
  - Microservices
  - Connection multiplexing
  - Write isolation
  - Custom caching
  - …. And finally ....
  - Sharding!

Custom Sharding Architecture evolution

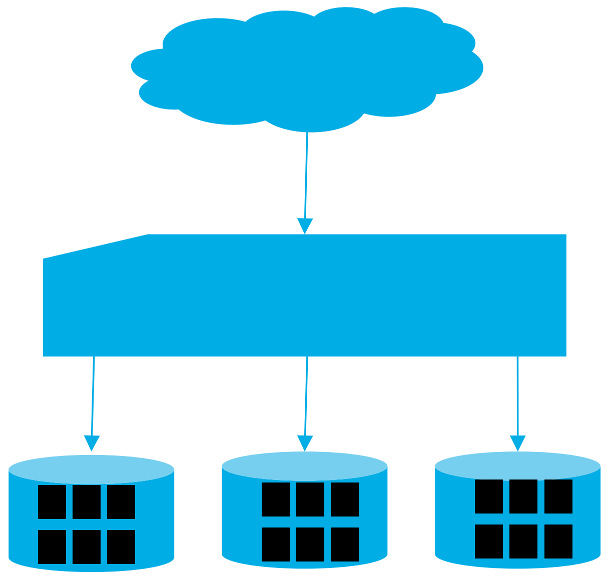
- App code determines shard to query
- Dev needs to understand sharding!



Logical Shard # 1- 8 9- 16 16- 24

- Tables T1 – T24 spread on 3 physicals
- Shards hardcoded to Physical servers
- Connection pool per physical
- Shard migration is Hard!

- Data Access Layer determines Shard to query (not App code!)



1 2 3 Physical Shard #

- Logical Shards mapped to Physical shards
- Logical to Physical mapping metadata cached in Data Access Layer (Shard Catalog!)
- One Connection pool per logical schema
- Moving Shard is Easy! (or rather: Easier)

PayPal

Sharding Principles

- Sharding is mostly a Scalability play
- ”Key is Key” – Choose one STRONG key and align. E.g. Account/User ID
- Data model changes (drastically!) – Normalization is relaxed
- Cross-key transaction boundaries are broken
- Non-shard key access can be expensive; forces “scatter-gather” pattern
- CAP Theorem: 2 of 3 for (C)onsistency, (A)vailability and Network (P)artition
- “Lookup” requirement for Common data elements
- Scheme for mapping logical to physical is critical for future scale-out
- Joins and ACID principles usually not available in Sharded systems (e.g. NoSQL)
- *Oracle Sharding supports the sharding principles while providing consistency, transactions, and relational capabilities such as Joins*

Oracle Sharding – Feedback from Beta software

- What we liked
  - Auto-configuration (Create and Manage Shards and Routing)
  - Out of box support for Shard Catalog, Routing layer, Duplicated Tables
  - Cross-shared query support (Not available with custom sharding)
  - Familiar SQL for sharding management
  - Two level sharding (not available in other technologies such as NoSQL)
  - Consistent hashing to prevent large data movement as new shards are added
  - Dynamically changing the number of shards
- Things to be careful of
  - Sharding requires major rework in Data Model and Data Design
  - Duplicated Table – Make sure not to over-use; Not for write-heavy use cases
  - Enable connection pool limits per Shard