ORACLE®

The Java Petstore v.Cloud

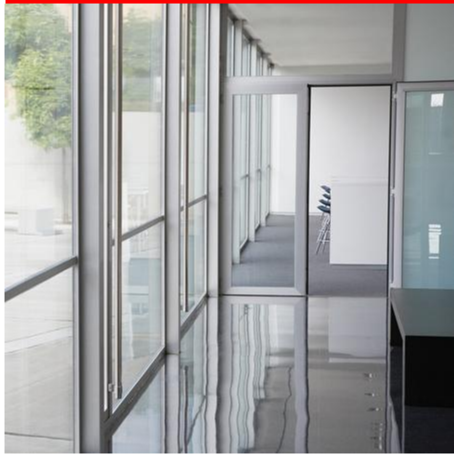
Building an ecommerce system to survive Cyber Monday

Cameron Purdy
Oracle Vice President of Development

About the Speaker: Cameron Purdy

- Vice President of Development, Oracle Coherence Data Grid
- Founder and CEO of Tangosol, acquired by Oracle in 2007
- Involved with 5 of the top 10 eCommerce sites (based on Cyber Monday 2009 rankings)

Program Agenda



- Concepts of Scalability and HA (S+HA)
- Applying S+HA to aspects of eCommerce
  - Problem
  - Solution
  - Example
- Conclusion

Concepts

Scalability

- Push the ability to answer the question as close to the user as possible
  - AJAX, CDN/Edge Caching, Global Load Balancing
- Partition & Load-Balance
  - Horizontal scalability
  - Design for elasticity
- Only “buy” what you need
  - What doesn’t scale easily? State Consistency, Ordering & Durability guarantees

Scalability: Orders of Magnitude

- CPU+RAM
- SSD (Local Flash)
- HDD (Local Disk)
- LAN Asynchronous
- LAN Partitionable
- WAN Asynchronous
- WAN Partitionable
- LAN Centralized
- WAN Centralized

ORACLE

High Availability (HA)

- The concept behind HA is simple: *No SPOF*
- Three simple solutions:
  - Stateless – any server can provide availability
  - Cached – *most* information can be recent, not current
  - Replicated – only essential information is guaranteed to be consistent and durable
- Eliminate SPOFs: Go redundant + load balance

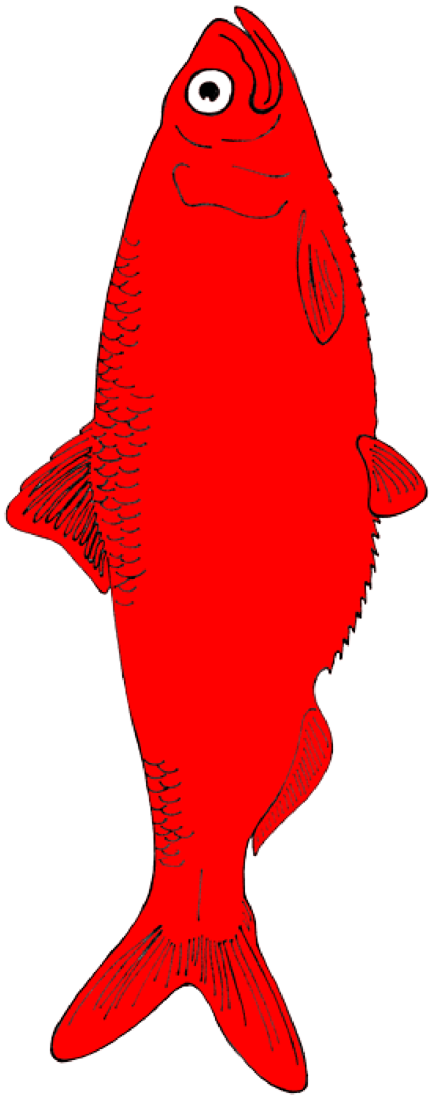
ORACLE

S+HA = Scalability + High Availability

- Natural Partners: S+HA
  - Scale Out *is* Redundancy
  - Load Balancing *is* Failover
  - Geo-distribution *is* S+HA
- The same solutions:
  - Stateless – always S+HA
  - Cached – S+HA iff recent data is in cache *or* loadable
  - Replicated – Challenging!
    - You *must* partition!
    - Synchronous Geo-Replication is *hard!*

Why not “Performance”?

- It’s is a red herring
  - No substitute for scalability
  - When a user is waiting on a response from a busy site, it may *feel* like a performance problem, but it is actually a scalability problem
- Solving scalability first allows you to address performance
  - SSL acceleration, Caching, CDN, geo-locality, AJAX



Defining “Scalable Performance”

- Scalable Performance is focused on insuring that the application performance does not degrade beyond defined boundaries as the application gains additional users, how resources must grow to ensure that, and how one can be certain that additional resources will solve the problem

Application

Catalog

- Problem: Large volumes of high-frequency, read-mostly, hierarchical data
  - Update propagation
  - Tied to Inventory & Search
- Solution: Pre-massage data, key-based access only, push content out
  - No Queries
  - Pre-build and cache trees
- Example

YOU SHARE IN OUR PROFITS ON EVERY PURCHASE. SEE LAST 16 PAGES.
SOLID GOLD AND SOLID SILVER SEAL AND ENGRAVED RINGS.
SOLID SILVER RINGS.

Inventory

- Problem: Separate SOR, free inventory affects catalog, search and order placement
- Solution: Real-time inventory mirrors SOR
  - Partition locally, async push globally, async refresh
  - Enables depletion events
  - Eliminates most SOR load
- Example

Search

- Problem: Computationally *and* data intensive; better search translates directly to increased revenue
- Solution: Design search as stateless as possible to enable the cloud option
  - Catalog by periodic push and event-based update
  - Inventory by exception
- Example

User Sessions

- Problem: Mutable; only needed by one user; must survive failure; total size grows in direct relation to concurrent user count
- Solution: Localize to a site (ownership)/server (partition); geo replicate async and transfer sync
- Example

Shopping Cart

- Problem: Must survive a user session, be durable, tied to account as soon as possible, persisted to free up memory
- Solution: Pre-auth, use a session; make durable post-auth
  - Session caches the cart
  - Async persist e.g. write-behind or session expiry
- Example

Order Placement

- Problem: Transactional bottleneck; involves inventory and other SOR; durability implications for billing, fulfillment
- Solution: Fully async workflow; minimize synchronous checks up front (inventory & credit)
- Example

ORACLE

A/B Testing

- Problem: How to test possible changes to the application to measure impact on page time, user conversion, margin, etc.
- Solution: Tie to user session, record stats, tally asynchronously e.g. when the session expires
- Example

Conclusion

Java SE, Java ME
JDK, Java EE
Ajax
JavaServer Faces
JDBC
Java ME

Conclusion

- Solve scalability at an architectural level; it’s a make-or-break
  - Partition, shard, distribute
  - Cache and/or replicate “must be present” state
- HA only matters when you need it, and it’s only missing when it matters
  - Redundancy at all levels
  - Isolate “must survive” state; nothing else matters

SOFTWARE. HARDWARE. COMPLETE.

ORACLE®