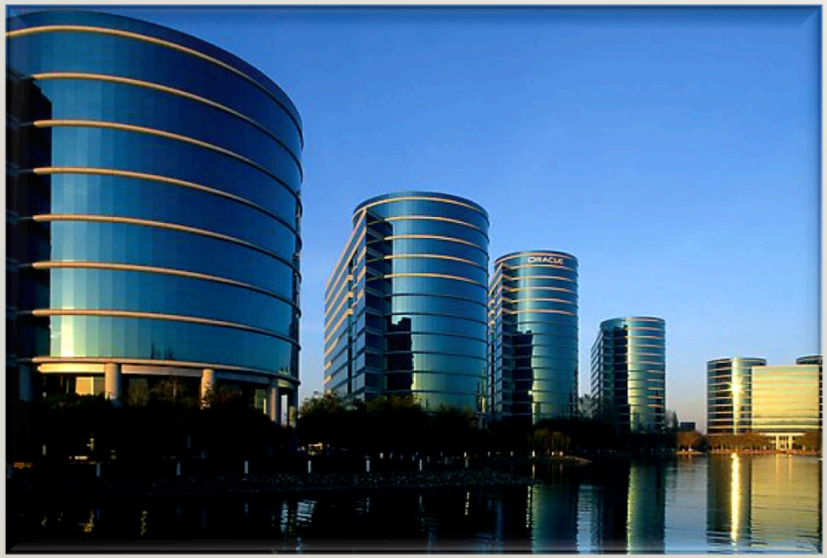
ORACLE®



ORACLE®

Connected Clouds: Middleware Infrastructure

Brian Oliver
Global Solutions Architect | brian.oliver@oracle.com

Oracle Coherence | Oracle Fusion Middleware Product Management

Not about rainbows…

Not about “cloudy” things

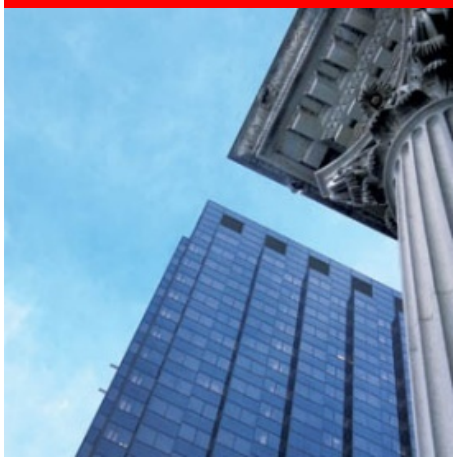
Agenda

- Not about…
  - Amazon EC2, EBS, S3, VIP (or other cloud vendor)
  - Licensing and Pricing Models
  - Auto-Scaling
  - Fault Tolerance
  - High Availability
  - “On demand” / “Map Reduce” …

Agenda

- How to make a globally distributed application appear and operate as a single application.
- Case Study: Globally Distributed Auction

Agenda

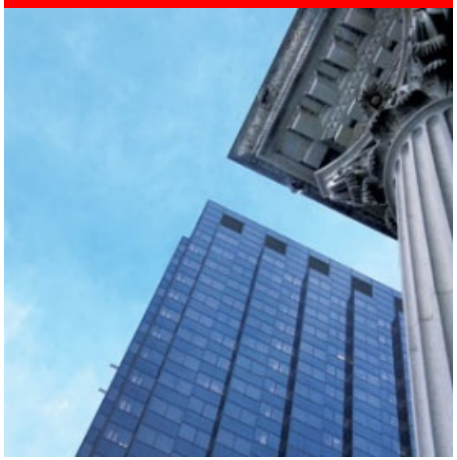


- Why one site isn't enough…
- Introduction to Oracle Coherence
- Multi-Site Challenges
- The Push Replication Pattern
- Deployment Models
- Real-World Use Case
- Demonstration

Why one site isn't enough…

- Two reasons for multi-site deployments
  - Business Continuity / Disaster Recovery
  - Regional Scalability
  - "probably need 2x more than you think"
- You don't need to be a multi-national corporation
  - Simple Web-based Application with global adoption
  - Simple iPhone Application with global adoption
- Use Coherence for Shared Memory
  - Local high-availability and scalability
  - Interconnect for global availability and scalability

Agenda

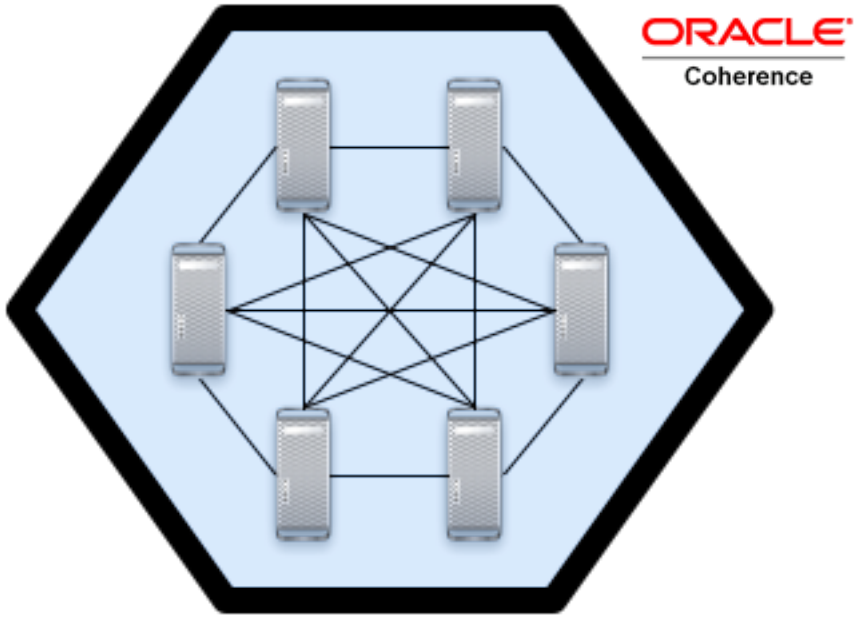


- Why one site isn’t enough…
- Introduction to Oracle Coherence
- Multi-Site Challenges
- The Push Replication Pattern
- Deployment Models
- Real-World Use Case
- Demonstration

Introduction to Oracle Coherence

- Software Development Library
  - Provides a Data Grid for Application Developers
    - Clustering Technology
    - Distributed Data Structures and Compute Services
  - Pure Java 1.4.2+ (servers & clients)
  - Pure .Net 1.1, 2.x, 3.x (client)
  - Pure C++ (client)
  - No Third-Party or Open Source Dependencies
- Other Libraries Support…
  - Database and File System Integration
  - Top Link, Hibernate, Http Session Management…

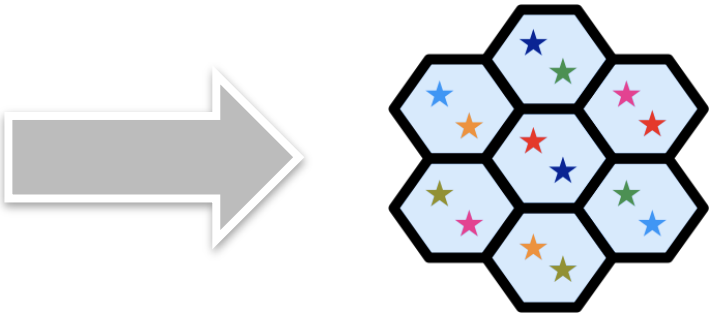
Introduction to Oracle Coherence

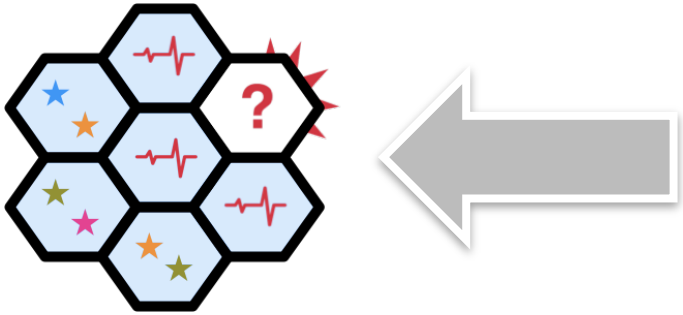


- Peer-to-Peer Clustering and Data Management Technology
- No Single Points of Failure
- No Single Points of Bottleneck
- No Masters / Slaves / Registries etc
- All members have responsibility for;
  - Managing Cluster Health & Data
  - Perform Processing and Queries
  - Self healing
- Communication is point-to-point (not TCP/IP) and/or one-to-many
- Scale to limit of the back-plane
- Use with commodity infrastructure
- Linearly Scalable By Design

Introduction to Oracle Coherence

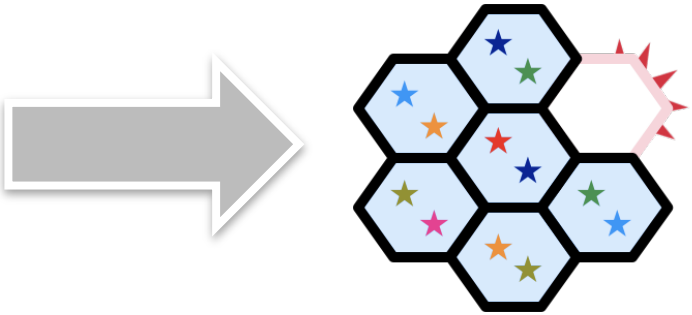
- Data is automatically partitioned and load-balanced across the Server Cluster
- Data is synchronously replicated for continuous availability



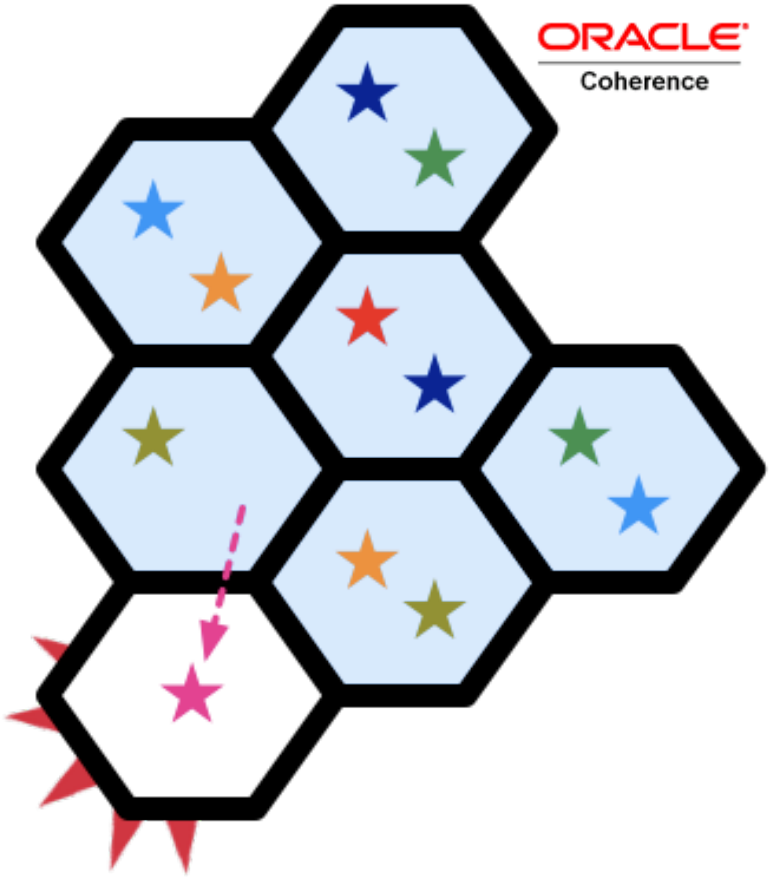


- Servers monitor the health of each other
- When in doubt, servers work together to diagnose status

- Healthy servers assume responsibility for failed server (in parallel)
- Continuous Operation: No interruption to service or data loss due to a server failure

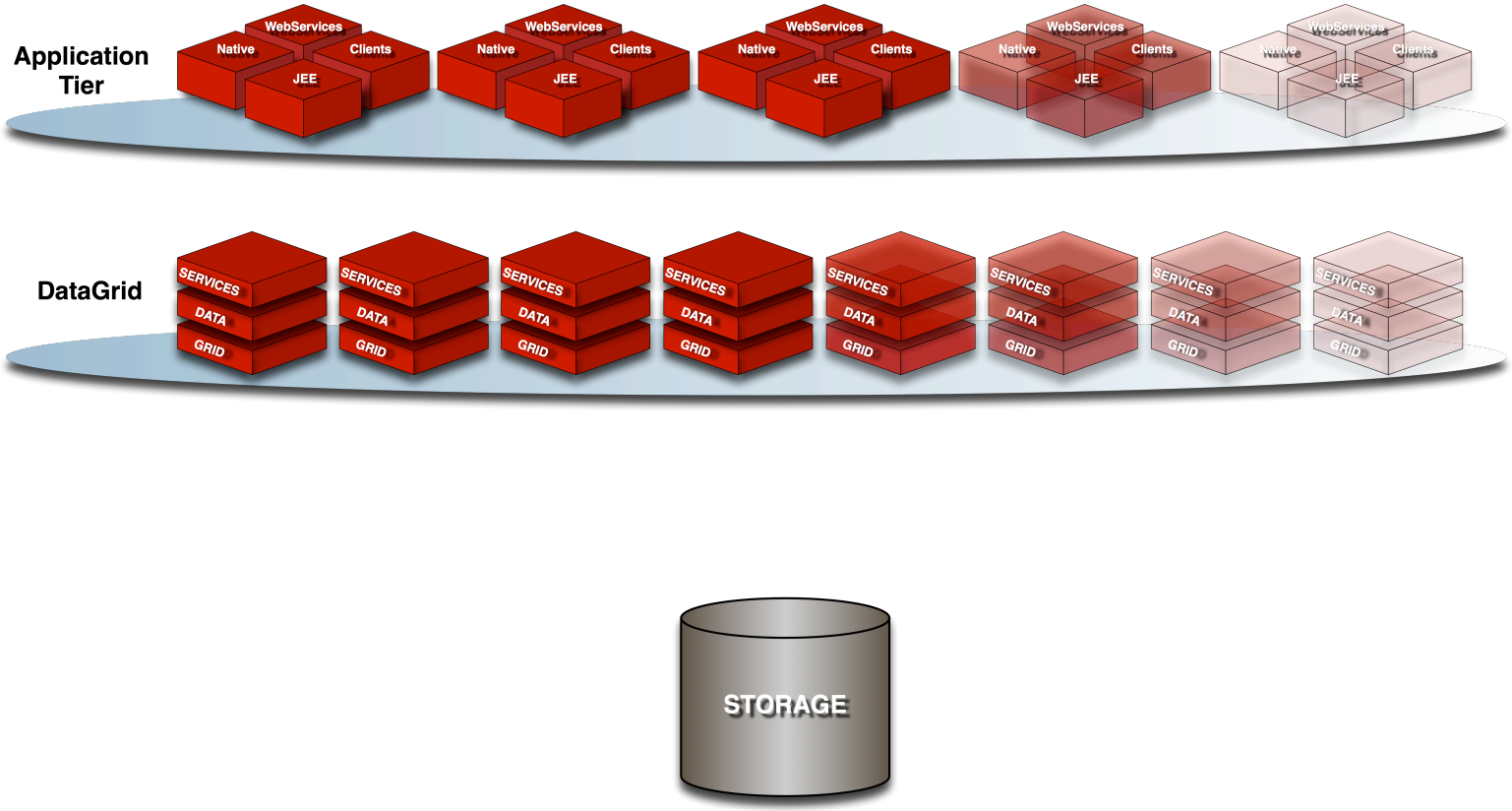


Introduction to Oracle Coherence



- Dynamically scale-out during operation
- Data automatically load-balanced to new servers in the cluster
- No repartitioning required
- No reconfiguration required
- No interruption to service during scale-out
- Scale capacity and processing on-the-fly

Coherence is Middleware



Coherence, Virtualization and Cloud

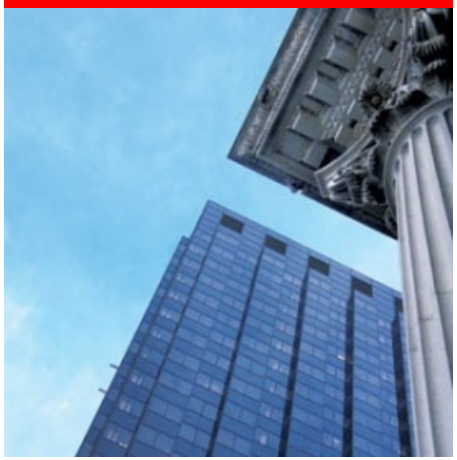
- Coherence is designed…
  - For single data-center
  - To take advantage of physical infrastructure
- Virtualized Infrastructure can suffer packet loss
  - 1Gb network = 110MB/sec throughput
  - Virtualized 1Gb network = 5MB/sec throughput!
    - Worst seen. Usually < 50% physical
- Can Coherence be used virtually or in a cloud?
  - Yes
  - Remember: Clouds provide capacity, scalability and better utilization… not necessarily performance

Coherence in the Cloud

Infrastructure Provider	Audience	Model
Public (out-sourced)	Public	~~Virtualized~~
		Physical
	Private	~~Virtualized~~
		Physical
Private (in-sourced)	Public	~~Virtualized~~
		Physical
	Private	~~Virtualized~~
		Physical

… use physical for production and/or multi-virtual core …

Agenda



- Why one site isn’t enough…
- Introduction to Oracle Coherence
- Multi-Site Challenges
- The Push Replication Pattern
- Deployment Models
- Real-World Use Case
- Demonstration

Challenge #1: User Expectations

- Most users (and some developers) assume;
  - “It doesn’t matter where I am in the world, everything should perform the same way”
    - ie: Local and Distributed Applications should perform the same
  - All networks perform at the same perceived speed
  - The network is not shared
    - ie: All of the available bandwidth is theirs

Challenge #1: The Reality

- Applications aren't deployed everywhere
  - We'd like them to be
- All networks behave differently
- Network is usually shared by many
- The speed of light is actually incredibly SLOW!
  - Very noticeable over long distances
  - Networks are slower than the speed of light

Challenge #1: Distance Matters
Typical Network Latencies

Challenge #1: Distance Matters
Typical Network Latencies

Challenge #1: Distance Matters
Typical Network Latencies

Challenge #1: Distance Matters
Typical Network Latencies

Challenge #1: Distance Matters
Typical Network Latencies

Challenge #1: The Reality

- Communicating between UK and AU servers is 3 orders of magnitude (1000x) slower than locally
  - Ie: Do 1000x more work locally than between UK and AU
  - All users will notice this delay
- But… Bandwidth is usually very high ☺
  - Unfortunately latency is as well.

Challenge #1: The Lessons

- Architectures that **work** "locally" between servers **rarely** work without change between "globally" distributed servers
  - Global Architectures **must** be structured differently (from local architectures) to meet user expectations

- Achieving **good performance** in a globally distributed system means "keeping and operating on data **locally**"
  - Avoiding long-trips to data/operations
  - Means introducing "copies" = challenge of "consistency"

Challenge #1: The Lessons

- It's easy to give users the "illusion" of good performance
  - Perform operations asynchronously
  - This **will** change the application model for the user
- The greater the physical distance between servers, the more "illusion" is required
  - Asynchronous APIs are very different from Synchronous APIs
- Take advantage of available bandwidth!
  - Batch work for Asynchronous Processing

Challenge #2: Where to locate data/services?

- Deciding on “where” isn’t easy
- Different Strategies:
  - Site-based, Geography-based, Team/User-based, Domain-based, Legality-based
  - Can be Static or Dynamic
    eg: follow the sun or load-based

Challenge #2: The Reality

- Global Architectures typically require many strategies
  - Case Study uses two strategies
- Some data/services need to be everywhere ☹
  - “reference data” needs to be everywhere
- Achieving “efficiency” may require changing the business model

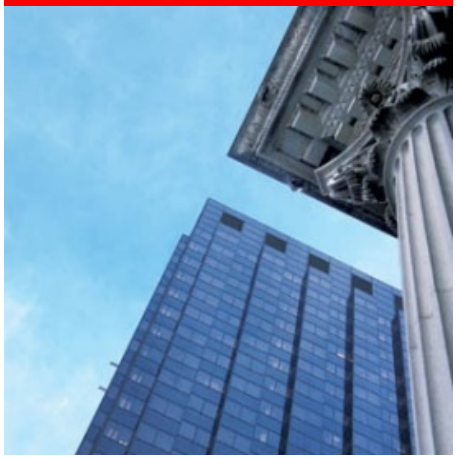
Challenge #3: Who owns the Data/Services?

- Single ownership is the ideal (“single master”)
  - Easy to understand
  - Easy to identify and control
- BUT:
  - May scale very poorly
  - Introduces “hot-spots”, “points of failure” and latency
- AND:
  - Is ownership static or dynamic?

Challenge #3: How is Data updated?

- Pessimistic Strategy:
  - “Global Locking Transactions”
  - Incredibly slow due to multiple round trips
  - Rarely viable over long distances or with multiple sites
  - Delivers “Guaranteed Consistency”
- Optimistic Strategy:
  - “Perform Updates Locally, Replicate and Resolve Conflicts”
  - Latency is close to theoretically possibilities (Real time)
  - Relies on “Eventual Consistency”
  - May be impossible to resolve conflicts

Agenda



- What is a DataGrid?
- Why one site isn't enough…
- Multi-Site Challenges
- **The Push Replication Pattern**
- Deployment Models
- Real-World Use Case
- Demonstration

The Push Replication Pattern

The Rationale

… provides and **extensible, flexible, high-performance, highly-available and scalable solution** to support the in-order optimistic replication of **data and operations** occurring in one Coherence Data Grids to **one or more** possibly globally distributed other Coherence Data Grids.

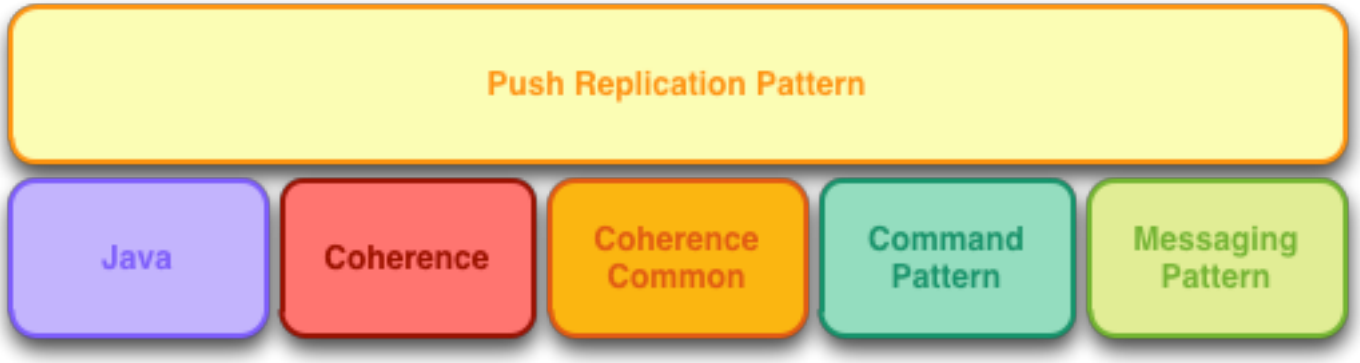
The Push Replication Pattern

- **The Push Replication Pattern advocates that**
  - *Operations* (such as insert, update and delete) occurring on *Data* in one *Location* should be **pushed** using one or more *Publishers* to an associated *Device*.
  - A *Publisher* is responsible for **optimistically replicating** *Operations* (in the order in which the said *Operations* originally occurred) on or with the associated *Device*.
  - If a *Device* is unavailable for some reason, the *Operations* to be replicated using the associated *Publisher* will be queued and executed (in the original order) at a later point in time.

The Push Replication Pattern

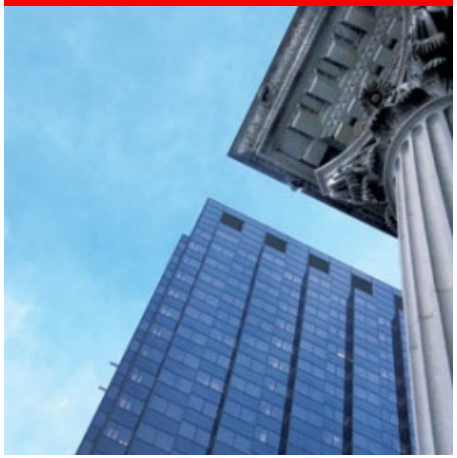
The Coherence Incubator

The Coherence Incubator **hosts a repository of projects** providing **example implementations** for commonly used design patterns, system integration solutions, distributed computing concepts and other artifacts designed to enable rapid delivery of solutions to potentially complex business challenges **built using or based on Oracle Coherence.**



http://coherence.oracle.com/display/INCUBATOR/

Agenda



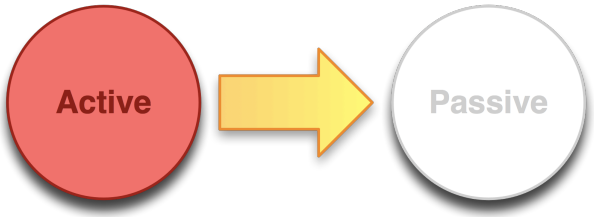
- What is a DataGrid?
- Why one site isn't enough…
- Multi-Site Challenges
- The Push Replication Pattern
- **Deployment Models**
- Real-World Use Case
- Demonstration

Deployment Models

“Master/Slave”

aka "Hot and Warm" aka "Active and Standby”

Updates to data made in the active grid are are sent to the passive grid **asynchronously and ordered**

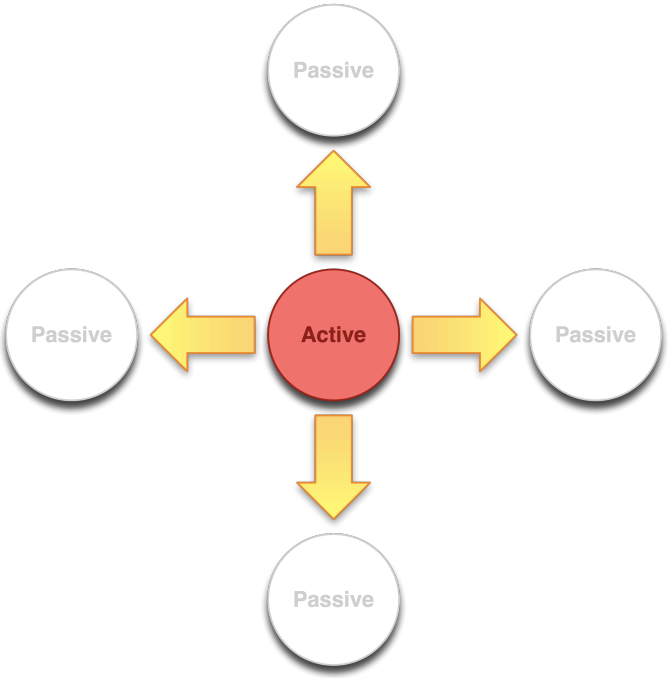


Deployment Models

“Hub and Spoke”

aka ”Master/Slaves”

Updates to data made in the active grid are are sent to **any number** of passive grids **asynchronously and ordered**

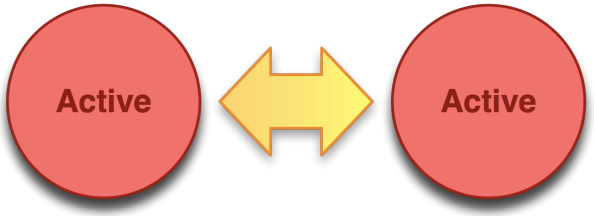


Deployment Models

“Hot Hot”

aka ”Federated”

Updates to data made **in either of the active grids** are are sent to other active grid **asynchronously and ordered.** (Conflicts are resolved on arrival)

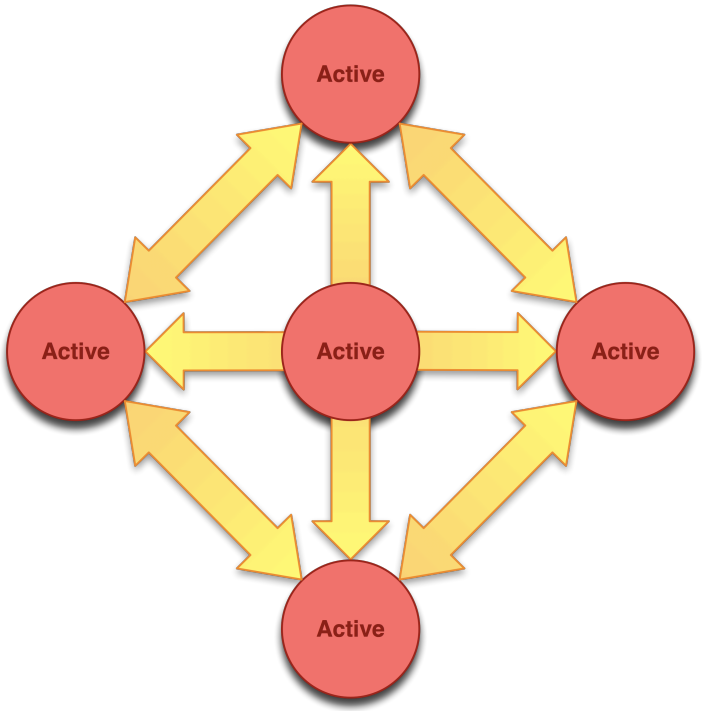


Deployment Models

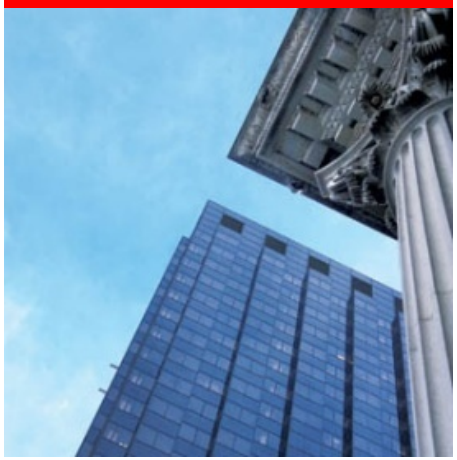
“Federated”

aka ”Multi-Master”

Updates to data made **in any active grid** are are sent **all other** active grids **asynchronously and ordered.** (Conflicts are resolved on arrival)



Agenda



- What is a DataGrid?
- Why one site isn't enough…
- Multi-Site Challenges
- The Push Replication Pattern
- Deployment Models
- **Real-World Use Case**
- Demonstration

Real-World Usecase

Real-Time Auction

- **Real-Time Auction**- Real-time online auction between New York and London.
- **Fairness** – Customers (Bidders) in either location see recent “global” bids, and if they make the highest bid it will be honored.
- **Scalability** - Application must support increase in demand, usage, catalogue, etc.

Real-World Usecase

The Players

- **Auctioneer**
  - Runs as a single instance at a single site (e.g. London)
  - Seeds the auction with items that are to be bid against
  - Establishes the starting price
  - Controls the auction duration
  - Signals bidders that the auction has started in both London and New York
  - Signals that the auction has stopped. The auctioneer then terminates.

Real-World Usecase

The Players

- **Bidders**
  - Runs as multiple instances in New York and London
  - Waits for an auction to start
  - Picks an item up for bid, gets the current bid, increases the bid and submits it to the auction on behalf of a customer
  - Bidders compete with each other **within a site**
    - Replication between sites means they compete against each other globally
  - All bids are processed.
  - Bidder stops bidding when the auctioneer signals that the auction is closed.

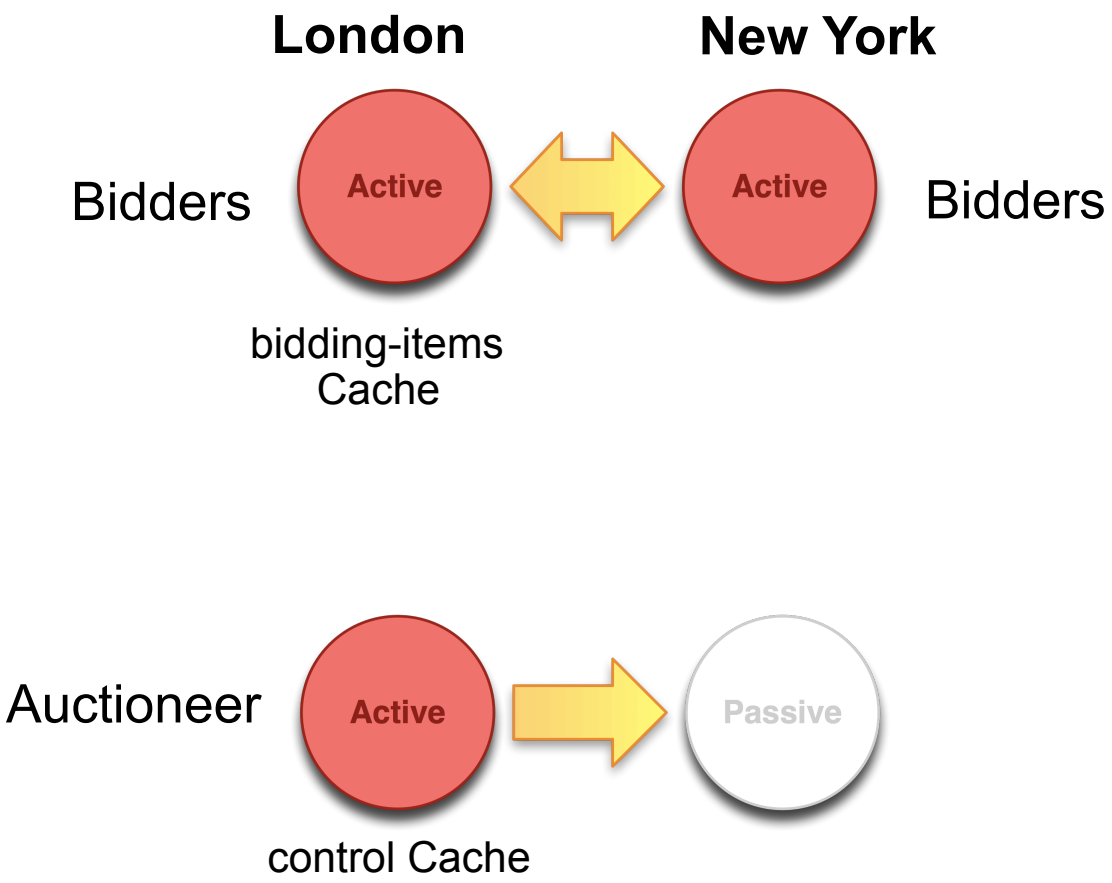
Real-World Usecase

What is Going On?

- Two separate Coherence clusters are running in New York and London operating against two caches (i.e. the bidding-cache and the control-cache)
- The clusters are using Coherence Incubator Push Replication to push bidding activity to New York from London and vice versa (active-active replication).
- It is also using Push Replication to push a single control object to both clusters (active-passive replication).
- Concurrent bidding is happening both within a cluster and between clusters.

Real-World Usecase

What is Going On?

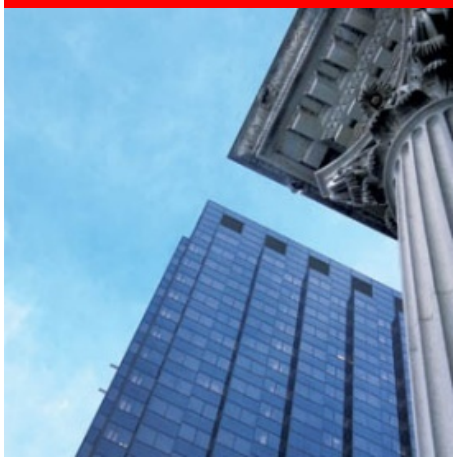


Real-World Usecase

What is Going On?

- Within a cluster, standard Coherence Entry Processors are used to reconcile concurrent bids between competing bidders in the same cluster.
- When bids are replicated to either New York or London, a registered Conflict Resolver object reconciles bids across the pond.
- Logic in both the Conflict Resolver and the Entry Processor is the same: is the bidding price higher than the existing price in the cache? If it is,then it becomes the current high bid in the cache. If it isn't, the bid is dropped.

Agenda



- What is a DataGrid?
- Why one site isn't enough…
- Multi-Site Challenges
- The Push Replication Pattern
- Deployment Models
- Real-World Use Case
- **Demonstration**

Globally Distributed Auction Demonstration

- Multiple 4 x virtual core servers (high CPU)
- Amazon European Cloud (west)
- Amazon United States Cloud (east)
- Fedora 32-bit base build
- Standard Java JDK 6
- Coherence 3.5.1
- Coherence Incubator Auction Example
- SWT-based GUIs

Other uses

- Global Session Management
  - Using Coherence and Push Replication to permit highly available multi-continent seamless availability
- Coherence Global System of Record
  - Trades / Shopping Carts
  - Integrating Multi Domain Systems
- Messaging
  - Replacing traditional message-based systems
  - Systems become “state based” not message-based.

- “If you can cache it, Coherence can distribute it”

For More Information

search.oracle.com

Coherence

or

oracle.com/products/middleware/coherence

ORACLE®