Data as a service - building reliable Data Grid with Oracle Coherence

Anatoli Atanasov,
Sales Consultant
Agenda

- Motivation for Oracle Coherence
- Coherence Technical Features
- Coherence Cache Topologies
- Q & A
- Demonstration
Motivation for Oracle Coherence
XTP Defined

An application style aimed at supporting secure, large-scale, high-performing transactions across a distributed environment on commodity hardware and software

Gartner Group
XTP Evolution

**Problem**
- Explosive growth of user bases overloading system capacity
- Rate of change faster than IT’s ability to re-architect systems

**Challenge**
- Do or Die Market Pressure to Meet User Expectations
  - Changing Service Level Agreements
  - Constant pressure to add new features and services
- Intense Pressure on Cost: Rapid infrastructure growth

**Solution**
- Unlimited, predictable scalability with capacity on demand
- Extreme performance with zero latency
- Dynamic by nature, able to easily change
- 100% transactional integrity and data reliability
- Continuous availability

*For data intensive high volume mission critical applications*
Application Scalability

- Scaling the Application-Tier is difficult

- If it was easy it would be an IDE option

- Scalability is a design option
  - Requires knowledge, care and experience
  - Developers have the “option” to consider building it in!
  - It’s not an IDE option

- Coherence is scalability infrastructure for the application-tier

Not possible!
## Scalability Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>How</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical “scaling-up”</td>
<td>Increase resources in existing server(s)</td>
<td>❖ Relatively simple process (can be achieved overnight)</td>
<td>❖ Comparatively expensive hardware (niche)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>❖ Transparent to system architecture and development</td>
<td>❖ Limited Scalability (physical limits typically encountered)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>❖ Increases cost of failure</td>
</tr>
<tr>
<td>Horizontal “scaling-out”</td>
<td>Add more servers</td>
<td>❖ Comparatively inexpensive hardware (commodity)</td>
<td>❖ Applicable only when a system is designed to “scale-out”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>❖ Virtually unlimited scalability possible (typically greater than scale-up approach)</td>
<td>❖ May require months of rework to achieve</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>❖ Scalability may be limited by “network”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>❖ Requires additional administration</td>
</tr>
</tbody>
</table>
Why Go Outside the Database to Scale Java Applications?

A **HUGE** performance bottleneck:
Volume / Complexity / Frequency of Data Access
Performance Bottleneck Between Tiers

Solution:
Move relevant data to middle tier

Java
Application

- The Solution is to keep the object data in object form in high-speed distributed memory cache
- Database remains the system of record (persistence)
The Solution ....

Oracle Coherence
Oracle Coherence
Reliable, Coherent, In-Memory Data Grid

Clusters with Virtual Memory Pool

Databases

RT Client

App Server

SOA/BPM

Data Grid Clients
Oracle Extreme Transaction Processing (XTP)

- Data Demand outpacing Data Supply
- Rate of growth outpacing ability to cost effectively scale applications
Oracle Extreme Transaction Processing (XTP)

- Oracle Coherence brokers Data Supply with Data Demand
- Scale out Data Grid in middle tier using commodity hardware
Oracle Grid Computing: Enterprise Ready

- Common Shared Application Infrastructure (Application Virtualization)
- Data Virtualization (Data as a Service)
- Middle tier scale out for Grid Based OLTP
- Massive Persistent scale out with Oracle RAC
How Does Oracle Coherence Data Grid Work?

- Data load-balanced in-memory across a cluster of servers
- Data automatically and synchronously replicated to at least one other server for continuous availability
- Single System Image: Logical view of all data on all servers

- Servers monitor the health of each other
- In the event a server fails or is unhealthy, other servers cooperatively diagnose the state
- The healthy servers immediately assume the responsibilities of the failed server
- Continuous Operation: No interruption of service or loss of data due when a server fails
Coherence: A Unique Approach

• In Coherence…
  – Members **share** responsibilities (health, services, data…)
  – Completely Peer-to-Peer
  – No Single Points of Bottleneck (SPOB\text{s})
  – No Single Points of Failure (SPOF\text{s})
  – Linearly scalable to thousands of servers **by design**

• Servers form a full “mesh”
  – No Masters / Slaves etc.
  – Data Grid members work together as a team
  – Communication is almost always point-to-point
    • Designed for commodity switched infrastructures
    • Scalable throughput up to the limit of the backplane
Architectural Integration Approaches

• Architect Solutions with Coherence
  – Simple Java programming API for J2EE
  – .NET integration using C#
  – Read-Through / Write-Through / Write-Behind or Cache Aside

• Plug into Existing Applications
  – Hibernate/TopLink Integration
  – Session state scaling with Coherence Web
    • .NET or Java EE session state

• Pluggable Integration with Oracle (roadmap)
  – SOA Suite
  – WebCenter
  – Business Intelligence
  – Content Management
  – …
Oracle Coherence broad integration

• **Hot pluggable**
  – Broad support for leading App Servers: Oracle iAS, IBM Websphere, JBoss, Sun, etc…

• **Helps any back end DB environment**
  – Oracle, Sybase, DB2, SQL Server

• **Any vertical, Any application**
  – (java, .net, soon C++) that needs **Predictable Scalability**
Oracle Coherence Advantage

- **Protect the Database Investment**
  - Ensure DB does what it does best, limit cost of re-architecture

- **Scale as you Grow**
  - Cost effective: Start small with 3-5 servers, scale to hundreds of servers as business grows

- **Enables business continuity**
  - Providing continuous data availability
What is Coherence (Finally)
Coherence Is ..

- An object-oriented data manager for the grid
  - Data is managed in memory
  - Runs inside Java Virtual Machines (JVMs)
  - Client applications can be Java or C#/.NET
  - Distributed/partitioned across potentially hundreds of JVMs and dozens of servers
- Automatic “scale out” / horizontal scaling
  - If you add new servers, they automatically join the cluster and re-distribute the data evenly
  - Hardware is usually inexpensive commodity servers
- Automatic high availability
  - Every object in memory has a backup on another server
  - If the primary server fails, the backup takes over and makes another backup
  - A “Consensus” algorithm keeps track of which object is the “primary” (more later)
What is Coherence?

- Coherence (deployment perspective)
  - Single Library*
  - Standard Java Archive “JAR” for Java
  - Standard Dynamically Linked Library “DLL” for .NET connectivity (.Net 1.1 and 2.0)
  - *Other libraries for integration (Databases, Spring…)
  - No 3rd party dependencies!
  - Minimal “invasion” on standard code*
  - Configurable implementations of standard Map / Dictionary interfaces (NamedCache)
  - Provides Predictable Scalable Caching
  - “RemoteException” free distributed computing
What is Coherence?

- Coherence (architectural perspective)
  - Scale-out Applications State
  - Reliable Data Management / Data Abstraction Layer
  - Effortlessly Cluster Applications (clustering infrastructure)
  - Web (session management)
  - Front, Middle, Back Tiers
  - Thick Clients (AWT, Swing, Console, RCP…)
  - JSE or JEE
  - Remote Connectivity
  - Business Continuity and Disaster Recovery
  - Provide a Data Grid
Different Needs Require a **Different Data Manager**

<table>
<thead>
<tr>
<th>APPLICATION TIER</th>
<th>DATABASE TIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manage data as objects</td>
<td>• Manage data as rows/columns</td>
</tr>
<tr>
<td>• Manage data in memory – or the applications won’t scale</td>
<td>• Manage data on disk</td>
</tr>
<tr>
<td>• Scale horizontally</td>
<td>– Long-term persistent store</td>
</tr>
<tr>
<td>– Add new, inexpensive nodes when more capacity is needed</td>
<td>• Scale vertically</td>
</tr>
<tr>
<td>• Query by object ID, meet the immediate needs of the app</td>
<td>– Usually SMP, big boxes, big storage devices, etc.</td>
</tr>
<tr>
<td></td>
<td>• Ad-hoc query, data warehouse query, SQL</td>
</tr>
</tbody>
</table>

Very different needs = different data management
Coherence Cache Topologies

Topologies to Ensure Cache Coherency
Local Caching

Coherence Local Cache Scheme
The Local Scheme

- Non-Clustered Local Cache
  - Contains a local references of POJOs in Application Heap

- Why:
  - Replace in-house Cache implementations
  - Compatible & aligned with other Coherence Schemes

- How:
  - Based on SafeHashMap (high-performance, thread-safe)
  - Size Limited (if specified)

- Configurable Expiration Policies
Replicated Caching

Coherence Replicated Cache Scheme
The Replicated Cache Scheme

• Bruce-force implementation of Clustered Caching

• Challenge
  – Need **Extreme Performance** (read)

• Solution
  – Replicate and maintain copies of all entries in all Members
  – Zero latency access as all entries are local to Members
  – Replication and syncing process transparent to developer

• Configurable Expiration Policies:
  – LFU, LRU, Hybrid (LFU+LRU), Time-based, Never, Pluggable
The Replicated Cache Scheme
The Replicated Cache Scheme

So What Is the “Cache”…ahem…Catch?
The Replicated Cache Scheme (Updates)
The Replicated Scheme

- **Cost Per Update**: Updating a replicated cache requires pushing the new version of the data to all other cluster members, which will limit scalability if there are a high frequency of updates per member.
  - Each Member must be updated!
  - Not scalable for heavy writes!

- **Cost Per Entry**: The data is replicated to every cluster member, so Java heap space is used on each member, which will impact performance for large caches. Cost Per Entry
  - Each Entry consumes Nx memory (N = #Members)
  - 1x for each Member
  - Not scalable for large caches!
The Replicated Caching Scheme

So How to Solve the Scalability Issue?
Distributed [Partitioned] Caching

The Disturbed...ahem...Distributed Scheme
Distributed Cache Scheme

- Sophisticated approach for Clustered Caching
- Challenge:
  - Need **Extreme Scalability**
- Solution:
  - Transparently partition the Cache Data; distribute the load across all cluster members, including backup cache entries
  - Often referred to as ‘Partitioned Topology’
  - **Linear Scalability**: By partitioning the data evenly, the per-port throughput (the amount of work being performed by each server) remains constant.

- Configurable Expiration Policies:
  - LFU, LRU, Hybrid (LFU+LRU), Time-based, Never, Pluggable
The Distributed Cache Scheme
The Distributed Cache Scheme
Distributed Cache Scheme

- Each Member has logical access to all Entries
  - At most 2 network-hop for Access
  - At most 4 network-hops for Update
  - Regardless of Cluster Size
  - This is why Coherence Scales!

- Linear Scalability
  - Cache Capacity and Processing Power Increases with Cluster Size
  - Coherence Load-Balances Partitions across Cluster
  - Point-to-Point Communication
  - No multicast required (sometimes not allowed)
  - **Ownership**: Exactly one node in the cluster is responsible for each piece of data in the cache.
The Distributed Cache Scheme
The Distributed Cache Scheme

• Distribution is invisible to application
  – The application does not need to know the physical location of the data
• Recovery occurs in Parallel
  – Not 1 to 1 like Active + Passive architectures
• Any Member can fail without data loss
• Configurable # backups
• No Developer or Infrastructure intervention
The Distributed Cache Scheme

• Benefits:
  – Deterministic Access and Update Latency (regardless of Cluster Size)
  – Seamless, automatic failover and failback
    • Backups ‘promoted’ to be Primary
    • Primary ‘makes’ new Backup(s)
  – Cache Capacity Scales with Cluster Size Linearly
  – Dynamically scalable without runtime reconfiguration
  – Automatic recovery and failover if any JVM or server failure occurs

• Constraints:
  – Cost of backup (but less than Replicated Topology)
  – Cost of serialization
  – Cost of non-local Entry Access (across the network)
    • (use Near Scheme – Discussed Next)
  – **Existence of Latency**
The Distributed Cache Scheme

• Lookup-free Access to Entries!
  – Uses sophisticated ‘hashing’ to partition and load-balance Entries onto Cluster Resources
  – No registry is required to locate cache entries in Cluster!
  – No proxies required to access POJOs in Cluster!

• Master / Slave pattern at the Entry level!
  – Not a sequential JVM-based one-to-one recovery pattern

• Cache still operational during recovery!
The Distributed Caching Scheme

So How to Solve the Latency Issue?
Local & Distributed Caching Combination

The Near Cache Scheme
The Near Cache Scheme

- A composition of pluggable **Front** and **Back** schemes
  - Provides L1 and L2 caching (cache of a cache)
- Challenge:
  - **Scalable Performance**
  - Partitioned Topology may always go across the wire
  - Need a local cache (L1) over the distributed scheme (L2)
  - Best option for scalable performance!
- Solution:
  - Add in-memory performance to distributed cache scalability.
  - Configure ‘front’ and ‘back’ topologies

- Configurable Expiration Policies:
  - LFU, LRU, Hybrid (LFU+LRU), Time-based, Never, Pluggable

- **Coherency**: Provides a number of cache-invalidation strategies, including simple expiry and event-based invalidation.
The Near Cache Scheme
The Near Cache Scheme
The Near Cache Scheme (+ Storage Option)
Clustered Caching & Scalability Performance Summary

- Clustering provides reliability through redundancy, and scalability by horizontal scale

- Applications that delegate all state management to the database will not scale well

- Clustered caching can significantly reduce the back-end load, resulting in scalable performance

- Decoupling the application from the back end (using caching, clustered data, write-behind and JMS) can help make applications Highly Available
Coherence Architectural Patterns

Coherence & Other Oracle Offerings
Coherence and FMW
Natural Integration Points

User Interaction
- Web 2.0 Portal, Rich Internet Apps, Mobile, Search, Desktop, Presence, VoIP

Business Intelligence
- Data Integration, Query & Analysis, OLAP, Dashboards, Reports, Alerts, Real-Time

Content Management
- Web Content, Documents, Digital Assets, Imaging, Records, Information Rights

SOA & Process Management
- ESB, BPEL PM, Workflow, BAM, Rules, B2B, MDM, Registry, SOA Governance

Application Server
- Java EE, Web Services, Complex Event Processing, XTP, RFID & Sensors, SIP

Grid Infrastructure
- Application Clusters, In-Memory Data Grid, Common Metadata Services

Session Sharing and Data Caching

Development Tools
- Unified SOA Development Tool & Framework

Data Caching, Extended State Replication, Shared In-Memory Infrastructure

Content Caching

Enterprise Management
- Provisioning, Diagnostics, Tuning, Configuration Management

Identity Management
- Provisioning, Access Management, Federation,

Accelerated Stateful Business Processes; Clustered BAM

Shared Service for Java, .NET, C++ …
Coherence & Other Oracle Products
RAC, Times Ten, Coherence, Web Cache

Web Tier
- Web Cache
- Web Servers

Application Tier
- Application Servers
- Coherence

Database Tier
- Times Ten
- RAC

Network

HTML Data Structures in Memory
Web Cache offloads Web Servers, Improves Network Performance via Compression

Java Data Structures in Memory
Coherence caches Java Structures in Memory; Very Fast Access to Java Data in Memory across Mid-Tier Grid

SQL Data Structures in Memory
Times Ten & RAC provide Scalability to Database Data improving Query & Transaction Write Performance
Oracle RAC, Times Ten, Berkeley DB
Coherence has Natural Integration Points

**Berkeley DB**
Cache Overflow Integration with Coherence

**Times Ten**
Clustered Caching with Coherence

**Oracle RAC**
Persistence QoS with Coherence

- Berkeley DB Cache Overflow with Coherence
- Times Ten with Coherence
- Oracle RAC Persistence QoS with Coherence
Coherence & TimesTen Strengths

Two Best of Breed Solutions for managing data in the middle tier

Both provide:
- High Throughput
- Data Reliability
- High Availability

Oracle TimesTen

Oracle Coherence

Single Node (SMP)
Cluster (N-nodes)
Grid (NNN-nodes)

Scale Out
Q&A
For More Information

http://search.oracle.com

Coherence

or
