Coherence Architectural and Implementation Patterns

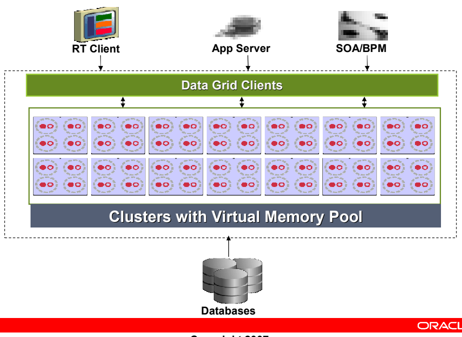
Oracle Coherence Workshop

Agenda

- Infrastructure and Configuration
  - Clustered Named Cache
  - Clustering Application Servers
- Data Source Integration and Access
  - Coherence To the Side, Behind, On-Top
  - Coherence Cache Topologies
- Coherence & Other Oracle Solutions
  - Coherence & Times Ten
  - Coherence & RAC
- Coherence & SOA Grid (Optional)
  - Coherence & Fusion Middleware
- Q&A

Oracle Coherence

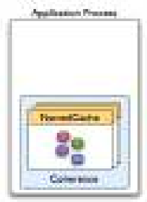
Reliable, Coherent, In-Memory Data Grid



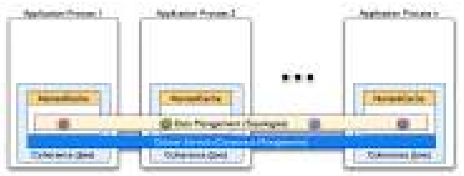
Coherence Architectural Patterns

Infrastructure & Configuration

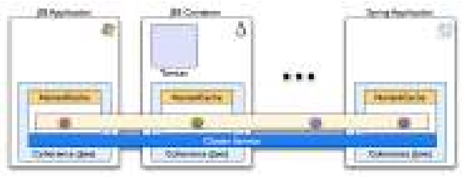
Single Application Process



Clustered Named Cache

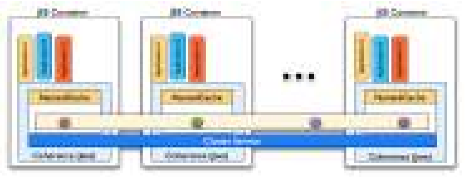


Clustered across Multiple Platforms



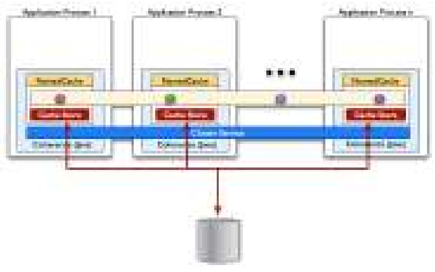
ORACLE

Clustering Application Servers



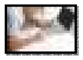
ORACLE

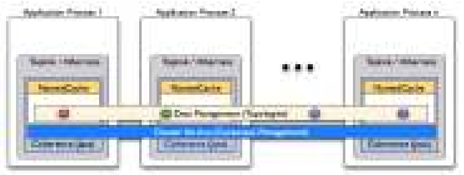
With Data Source Integration (Cache Stores)



ORACLE

Clustered Second Level Cache
(for Hibernate)

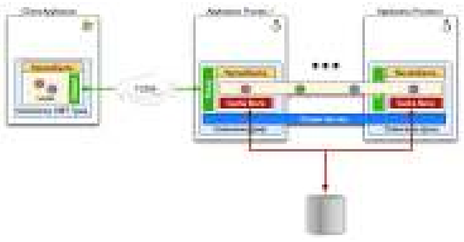




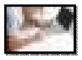
Coherence*Extend

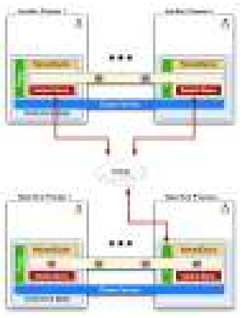
- Supports “fat client” real-time applications such as trading desks, as well as other server tiers
- Provides near caching capability within “fat client” app, and other server tiers connected to the cluster remotely (through firewall)
- Connection to the cluster is over TCP
- Continuous query can be used to maintain real-time query results on the desktop!

Remote Clients connected to Coherence Cluster



Interconnected WAN Clusters





Coherence Architectural Patterns

Data Source Integration and Access

Architectural Integration Possibilities!

Direct Data Integration:

Oracle Coherence Behind: Use Oracle Coherence as L2 cache for OR/M (Hibernate)

Oracle Coherence To-The-Side: Application manages Data CRUD in Oracle Coherence next to OR/M

Oracle Coherence On-Top: Oracle Coherence is System of Record. Use CacheLoaders and CacheStores to integrate with Data Sources

Cache-Aside Architecture

- Cache-Aside refers to an architecture in which the application developer manages the caching of data from a data source
- Adding cache-aside to an existing application:
  - Check the cache before reading from the data source
  - Put data into the cache after reading from the data source
  - Evict or update the cache when updating the data source

Cache-Through Architecture

- **Cache-Through** places the cache between the client of the data source and the data source itself, requiring access to the data source to go through the cache.
- **A Cache Loader** represents access to a data source. When a cache is asked for data, if it is a cache miss, then any data that it cannot provide it will attempt to load by delegating to the Cache Loader.
- **A Cache Store** is an extension to Cache Loader that adds the set of operations generally referred to as Create, Read, Update and Delete (CRUD)

Data Source Integration

- Coherence supports transparent read-write caching of any datasource, including databases, web services, packaged applications and filesystems, databases are the most common use case
- Effective caches must support both intensive read-only and read-write operations, and in the case of read-write operations, the cache and database must be kept fully synchronized.
- To accomplish this, Coherence supports Read-Through, Write-Through, Refresh-Ahead and Write-Behind caching.

Persisting Data to a Database

- So far, we have written data only to memory of the Coherence JVMs
  - It's persistent because of the backups
  - However, this is not permanently persisted to disk or a database
- Integration with databases is done with the Coherence *CacheStore*
  - Writes to the cache can persist to the database
  - Reads to the cache can obtain data from the database automatically
  - Any backing data source can be used: RDBMS, Mainframe, Disk File, Berkeley DB, etc.

ORACLE

Persisting Data – The mechanics

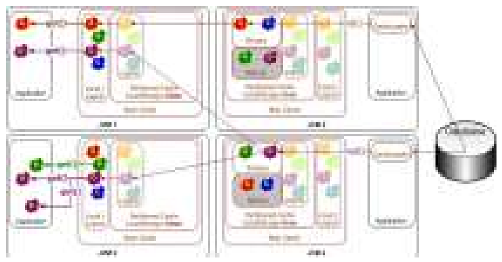
- Backing Maps are the method by which a NamedCache persists data
- Memory is the default implementation that we have been using
- This is achieved by using a different Backing Map to persist to databases, files ,etc

ORACLE

Read Through

Read from CacheLoader when data not in grid

- If data is not present in the cache, then the back end data source implementation is used to read the data and place it in the cache

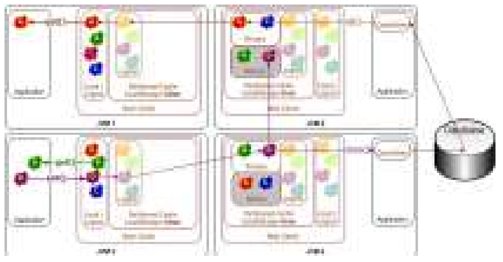


ORACLE

Write Through

Write to CacheStore when data inserted, updated, removed in grid

– When writing data, the "put" method will not return until the data is written the back end data source. E.g. synchronous

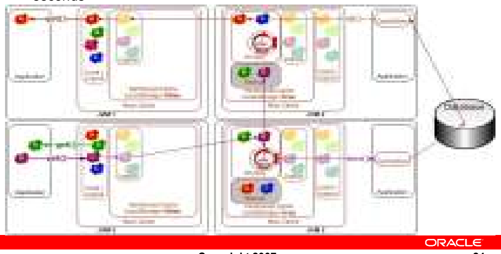


ORACLE

Write Behind

Asynchronous and coalesced updates to CacheStore when data inserted, updated, deleted in grid

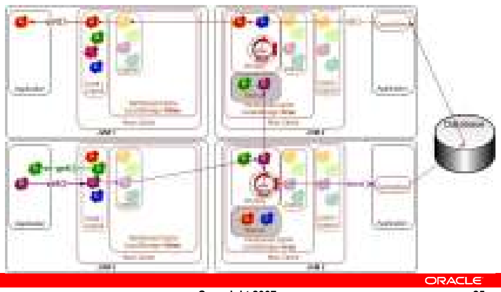
– Data is written asynchronously to the back end data source with a configurable delay. E.g. ensure that the data is written by a max of **n** seconds



ORACLE

Refresh Ahead

Data that is about to expire will be refreshed before its expiry time, so as to not delay any reads



ORACLE

Data Source Integration

There are a number of out of the box integrations:

- Hibernate
- Toplink Essentials
- Java Persistence Architecture (JPA)
- Simple JDBC
- File system

- You can also write your own ORM (Object-Relational Mapping) code using JDBC and implementing the CacheStore interface
  - http://wiki.tangosol.com/display/COH32UG/Sample+CacheStore

Architectural Integration Possibilities!

Session Management

Oracle Coherence Web = drop-in replacement to reliably cluster and scale out session management (Java and .NET) across a grid

Service Integration:

Oracle Coherence WorkManager: Use Oracle Coherence to resiliently manage and execute “tasks” across the members.

Invocation Service: Directly use Oracle Coherence Invocation Service to execute tasks on individual, sets or all members (sync or async)

Architectural Integration Possibilities!

Provide:

Push / Pull data model based on subscription and event notification

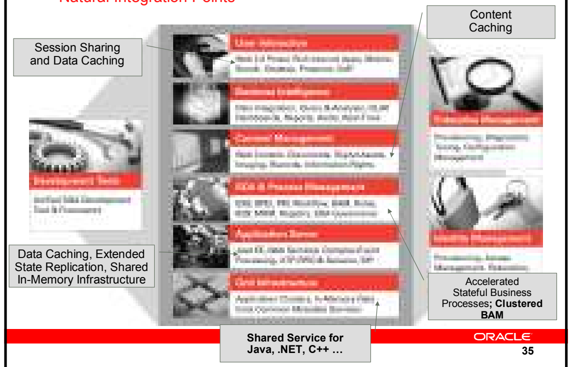
Client / Data Grid model where clients connect to Oracle Coherence for data and services

Coherence Architectural Patterns

Coherence & Other Oracle Offerings

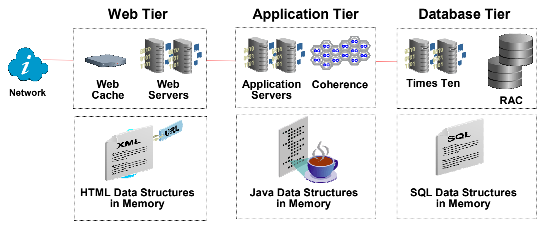
Coherence and FMW

Natural Integration Points


Content Caching
Session Sharing and Data Caching
Data Caching, Extended State Replication, Shared In-Memory Infrastructure
Accelerated Stateful Business Processes; Clustered BAM
Shared Service for Java, .NET, C++ ...

Coherence & Other Oracle Products

RAC, Times Ten, Coherence, Web Cache



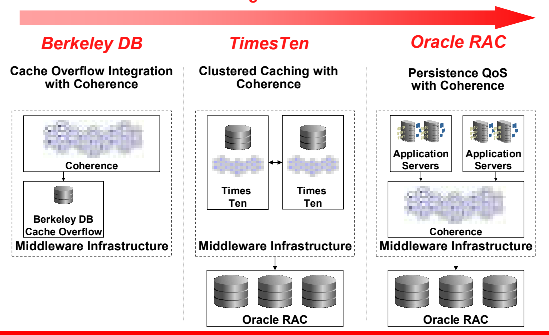
Web Cache offloads Web Servers, Improves Network Performance via Compression

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

Times Ten & RAC provide Scalability to Database Data improving Query & Transaction Write Performance

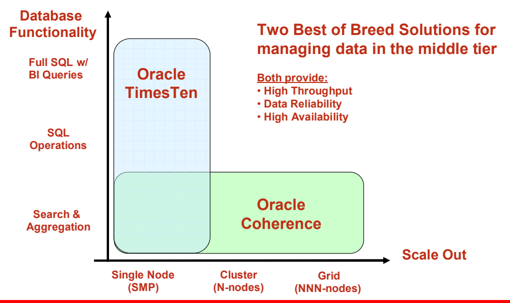
Oracle RAC, Times Ten, Berkeley DB

Coherence has Natural Integration Points



Berkeley DB
Cache Overflow Integration with Coherence
Coherence
Berkeley DB Cache Overflow
Middleware Infrastructure
TimesTen
Clustered Caching with Coherence
Times Ten
Times Ten
Middleware Infrastructure
Oracle RAC
Oracle RAC
Persistence QoS with Coherence
Application Servers
Application Servers
Coherence
Middleware Infrastructure
Oracle RAC

Coherence & TimesTen Strengths



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

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

Coherence & Other Oracle Products

RAC, Times Ten, Coherence, Web Cache

- Many different bottlenecks in Transaction Applications
  - Three Tiers in Architecture: Web Server, App Server, DBMS
  - Three Shapes to Data: SQL, Java, HTML
  - Three Types of Access to Data: Query, Transaction, Fetch
  - Different Types of Bottleneck: Network; CPU; Memory; I/O/Storage
- Key Performance & Scalability Considerations
  - RAC: Improve Scalability of DBMS to Manage Transactions by clustering Nodes Together – Data shaped as SQL
  - Times Ten: Offload DBMS while Caching Data In-Memory & Providing Queryability on Data – Data shaped as SQL
  - Coherence: Offload DBMS while Caching Data In-Memory within Java VM & close to Application – Data shaped as Java
  - Web Cache: Offload Application & Web Servers; outside Firewall & and caches Data In-Memory – Data shaped as HTML
- Oracle’s Solutions work together – Complete, Integrated

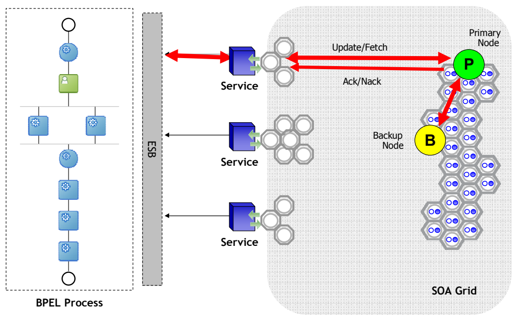
Coherence Architectural Patterns

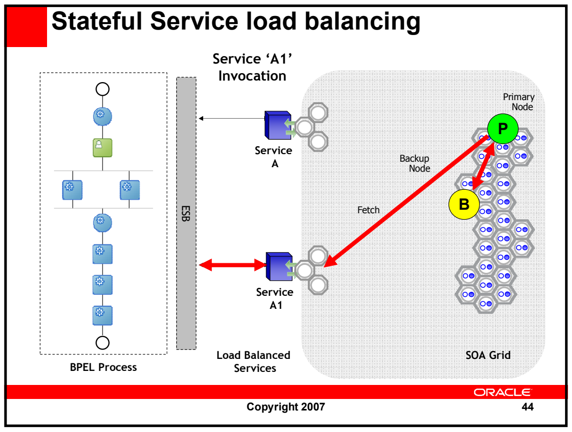
SOA Grid (Future)

SOA Grid – Advanced Capabilities

- Co-locate service code with grid data
  - Load balance and dispatch requests appropriately
- Availability and failover of Stateful services
- State Passing Model Redefined
  - BPEL dehydration into the grid
- Relocatable BPEL processes

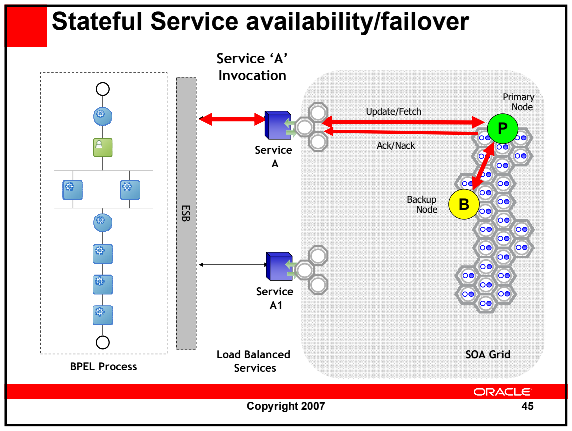
SOA Grid





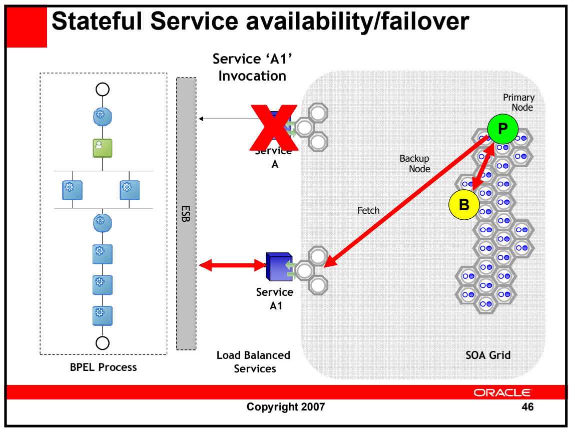
Stateful Service load balancing

Copyright 2007



Stateful Service availability/failover

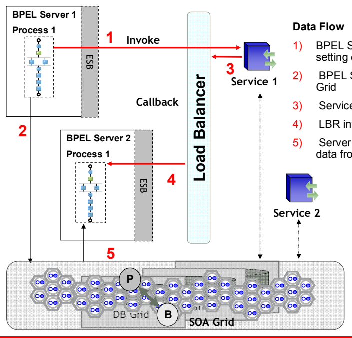
Copyright 2007



Stateful Service availability/failover

Copyright 2007

BPEL Dehydration Example



Data Flow

1) BPEL Server 1 Process 1 invokes Service 1 setting callback URL to LBR
2) BPEL Server 1 Process 1 dehydrates to Grid
3) Service 1 Invokes LBR
4) LBR invokes BPEL Server 2 Process
5) Server 2 Process 1 rehydrates process data from grid and continues

Service 1 and 2 can store their own data in the SOA Grid but they do not require access to the BPEL dehydration store in this example.

ORACLE

Q&A

ORACLE

ORACLE IS THE INFORMATION COMPANY

ORACLE®