
ORACLE®

Oracle Coherence 12.1.3

Cloud Application Foundation

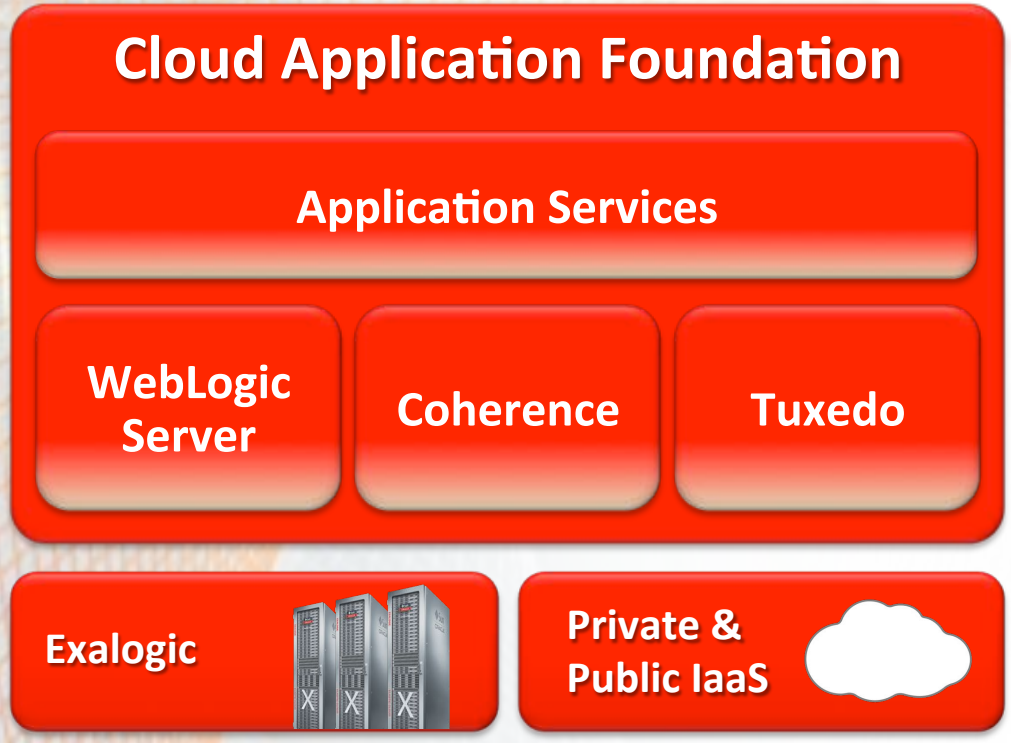
Dave Felcey
Product Management
Oracle Coherence
June, 2014

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Cloud Application Foundation



Deployment

Oracle Weblogic Server 12*c*

Release 12.1.3

Oracle Coherence 12*c*

Release 12.1.3

Oracle Tuxedo 12*c*

Productivity & Management

Oracle Enterprise Manager 12*c*

Oracle Development Tools 12*c*

Introduction to Memcached Adaptor

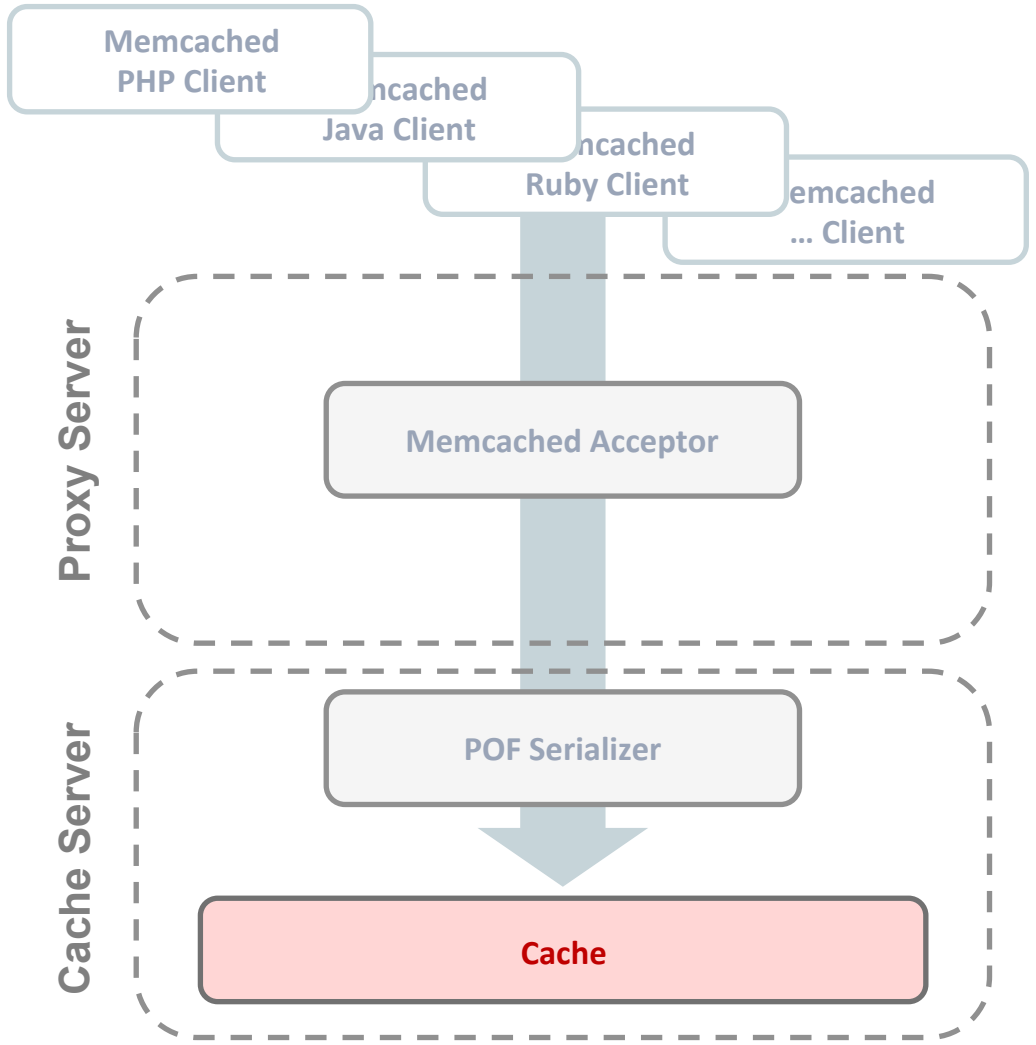
ORACLE

Motivation

- Allow Coherence to be a drop-in replacement for Memcached server.
- Leverage widely available Memcached client libraries to talk to Coherence from multiple languages.
- Provides an easy migration path from Memcached to Coherence.
- Take advantage of Coherence RASP features.

Memcached Adaptor

- Migrate from Memcached
- Use popular Memcached Clients
- Leverage Coherence benefits and features
  - Scalability, availability, and reliability
  - Data source integration including HotCache
  - Security
    - SASL PLAIN authentication mechanism using Coherence JAAS Identity Asserter
    - Integrates with the Coherence Proxy Security framework for custom authorization.



Implementation

- Memcached protocol support added to Coherence Proxy Service.
- Configured as an Acceptor in the proxy service similar to a tcp or http acceptor.
- Supports only memcached binary protocol.
- Access Coherence partitioned caches thru memcached clients.
- POF inter-operability between Coherence clients and memcached java clients that support pluggable serializers. (like spymemcached)

```
<proxy-scheme>
  <service-name>MemcachedProxyService</service-name>
    <acceptor-config>
      <memcached-acceptor>
        <cache-name>memcache</cache-name>
        <address-provider>
          memcached-provider
        </address-provider>
      </memcached-acceptor>
    </acceptor-config>
  <autostart>true</autostart>
</proxy-scheme>
```

Security

- Supports SASL PLAIN authentication mechanism using Coherence Identity Asserter

- Integrates with the Coherence Proxy Security framework for custom authorization.

Full-Lifecycle Monitoring and Management

DEV → **OPS**

JVisualVM Plugin

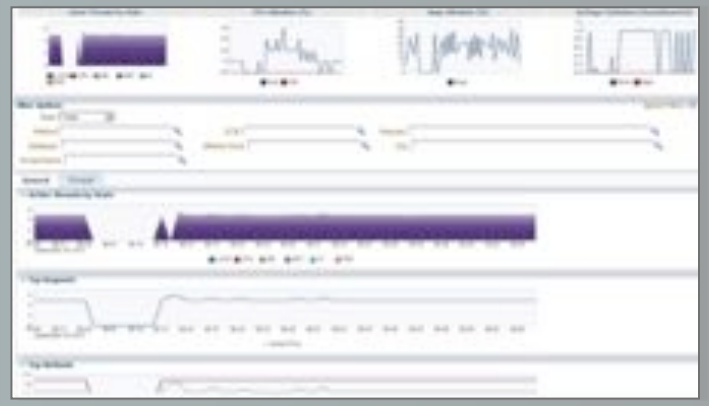
NEW

- Available now for 3.x on Coherence Community Website
- Lightweight plugin to JVM

Fusion Middleware Control

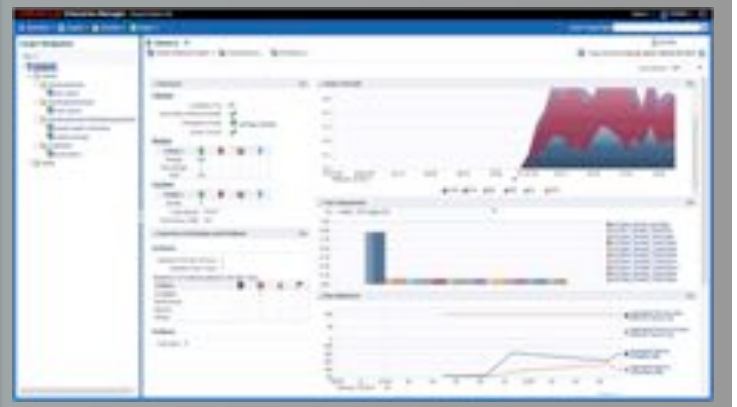
COMING SOON

- OOTB administration and monitoring for all FMW
- Dev/QA point-in-time insight into cluster



Coherence Management Pack for OEM

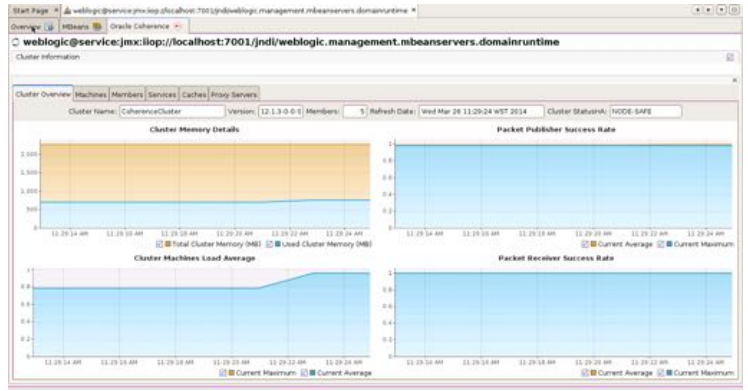
- Complete management and monitoring solution
- Store historical results
- Java diagnostics tooling

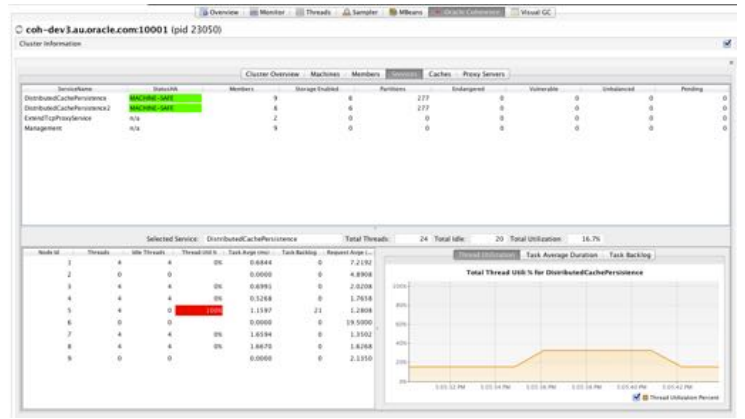


ORACLE

Coherence JVisualVM Plug-in Overview

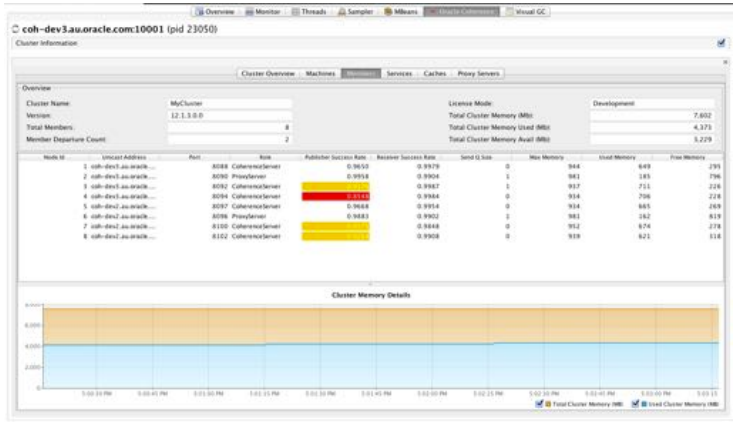
- A lightweight, developer focused plug-in for viewing general cluster information as well as aggregated Coherence MBeans
- Built as a standard NetBeans Module (NBM) utilizing the JVisualVM plug-in architecture to provide seamless integration with JVisualVM
- Information is presented in a tabular form or over time as graphs to allow real-time analysis and troubleshooting of your cluster
- Provides visual cues for pertinent information
- Supports Managed Coherence Servers (WebLogic Server) and traditional stand-alone clusters

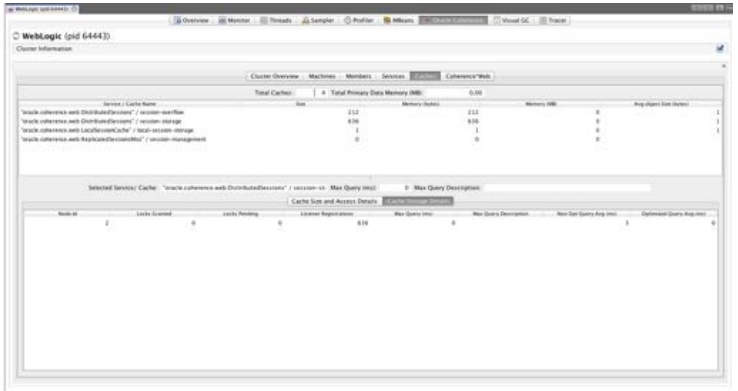




Coherence JVisualVM Plug-in Overview

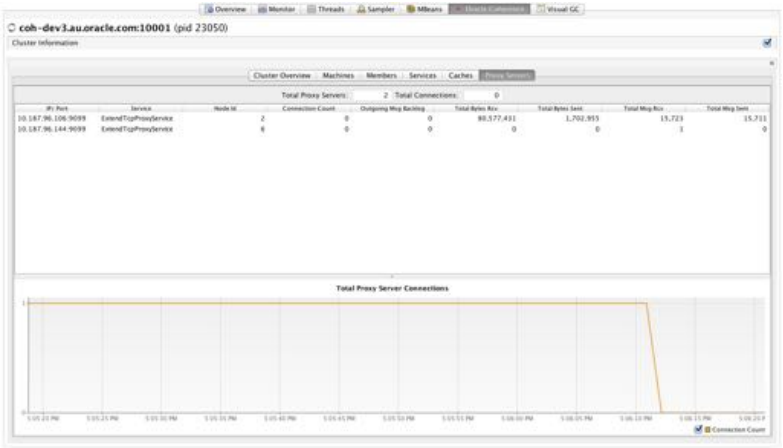
- Has been available as a community plug-in through “The Coherence Incubator”, but is now released as part of product in 12.1.3
- Displays information automatically based upon enabled functionality including:
  - General Cluster Information
  - Machines/ Nodes
  - Services
  - Caches
  - Proxy Servers
  - Coherence*Web
  - Elastic Data (available in a subsequent patch-set release)





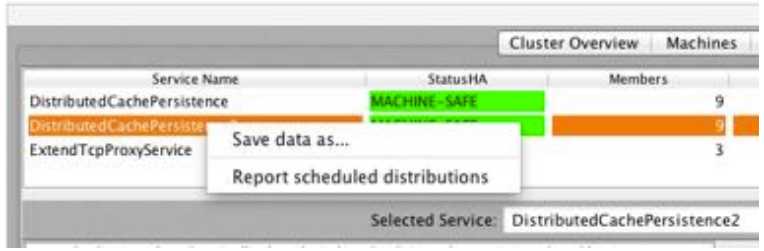
Coherence JVisualVM Plug-in Overview

- All data tables exportable as CSV for further analysis
- Configurable data refresh rate to ensure optimal Coherence cluster performance
- Supports Coherence versions 3.5.x and above via local or remote JMX connections to Coherence MBean Server
- Drill down capabilities for Services and Caches allowing per-node view of information such as thread utilization, task and request details.
- Available from JDK 1.7 u 40 and above
  (can connect to some clusters running older Java versions via remote JMX connection)



Coherence 12.1.3 and Beyond

- Part of Coherence product install from 12.1.3 - available in the "plugins" directory
  $COHERENCE_HOME/plugins/jvisualvm/coherence-jvisualvm.nbm
- Runs on any platform supported by the JDK. e.g. Windows, OSX, Linux, Solaris, etc.
- Utilizes new "runTabularReport" capabilities of Reporter to more efficiently gather JMX statistics
- Additional "Context Sensitive" information available via right-click on selected tables. This includes operations such as "Reporting scheduled distributions for service"
- Future Plans (subject to change)
  - Supporting new features, like Recoverable Caching & Federation
  - Additional context sensitive options will become available in subsequent patch-set release



Startup Options

- After installing the plug-in, there are a number of options you can pass to the *jvisualvm* command line to configure plug-in behavior:
  - Change the default time to refresh JMX statistics from 30 seconds:
    (This will add additional load to the cluster)

```
-J"-Dcom.oracle.coherence.jvisualvm.refreshtime=15"
```

  - Increase the timescale for each graph:
    (The default is 50,000 and represents approx. 12 hours. Increasing this will increase the amount of memory used by JVisualVM)

```
-J"com.oracle.coherence.jvisualvm.values.limit=100000"
```

  - Log diagnostic information regarding JMX query times:
    (To view the output, choose *About* menu and choose *Log File)*

```
-J"-Dcom.oracle.coherence.jvisualvm.log.query.times=true”
```

Example (Linux/Mac/Unix):

```
jvisualvm -J"-Dcom.oracle.coherence.jvisualvm.refreshtime=15" \
    -J"-Dcom.oracle.coherence.jvisualvm.log.query.times=true”
```

Demonstration

Memecached Client Support and Visual VM Plug-in

Introduction to JCache

ORACLE

Introduction to JCache

- **JCache defines the standard Caching API for the Java Platform**
  - A Cache is a Map-like data-structure that may be used to temporarily store keys and values

- Developed through the Java Community Process (JCP) (see: http://jcp.org)
  - Drives and manages enhancement requests to the Java Platform
  - Managed by the Java Executive Committee (EC)

- JCache also commonly known by the JCP specification number: **JSR-107**

Introduction to JCache

- JCache designed and developed by industry experts
  - Specification Leads: Greg Luck, Brian Oliver, Cameron Purdy
  - Large Expert Group:
    - Oracle, IBM, Red Hat, Fujitsu, Software AG...
    - Many individuals!

Introduction to JCache

- JCache consists of several parts:
  - JCache API (API)
  - JCache Reference Implementation (RI)
  - JCache Technology Compatibility Kit (TCK)
  - JCache Specification Document (SPEC)

Introduction to JCache

- JCache consists of several parts:
  - **JCache API (API)**
  - JCache Reference Implementation (RI)
  - JCache Technology Compatibility Kit (TCK)
  - **JCache Specification Document (SPEC)**

Introduction to JCache

- JCache Resources:
  - Project Information and Official Downloads (API, RI, TCK, SPEC):
    - https://jcp.org/en/jsr/detail?id=107
  - Source Code and Issue Tracking (API, RI, TCK and DEMO)
    - https://github.com/jsr107
  - Forums
    - https://groups.google.com/forum/?fromgroups#!forum/jsr107

Introduction to JCache

- Maven API Coordinates:
  - Group: javax.cache
  - Artifact: cache-api
  - Version: 1.0.0

Introduction to JCache

- **Providers produce and support implementations of the JCache API**
- Some Providers produce Compliant Implementations
  - Compliant Implementations must pass the TCK
  - Others may not…
- Compliance is very important!
  - Without compliance applications developed using JCache may not be portable

Simple Example

```
//acquire a previously configured named cache
Cache<Integer, String> cache = Caching.getCache("my-cache", Integer.class, String.class);

// put something in the cache
cache.put(123, "Hello World");

// retrieve it from the cache
String message = cache.get(123);
```

The JCache API

```
/**
 * A {@link Cache} is a Map-like data structure that provides temporary storage
 * of application data.
 */
public interface Cache<K, V> extends Iterable<Cache.Entry<K, V>>, Closeable {

    V get(K key);

    Map<K, V> getAll(Set<? extends K> keys);

    boolean containsKey(K key);

    void loadAll(Set<? extends K> keys,
                 boolean replaceExistingValues,
                 CompletionListener completionListener);
```

The JCache API

```
void put(K key, V value);

V getAndPut(K key, V value);

void putAll(java.util.Map<? extends K, ? extends V> map);

boolean putIfAbsent(K key, V value);

boolean remove(K key);

boolean remove(K key, V oldValue);

V getAndRemove(K key);
```

The JCache API

```
boolean replace(K key, V oldValue, V newValue);

boolean replace(K key, V value);

V getAndReplace(K key, V value);

void removeAll(Set<? extends K> keys);

boolean remove(K key);

void removeAll();

void clear();
```

The JCache API

```
<C extends Configuration<K, V>> C getConfiguration(Class<C> clazz);


<T> T invoke(K key,
             EntryProcessor<K, V, T> entryProcessor,
             Object... arguments) throws EntryProcessorException;


<T> Map<K, EntryProcessorResult<T>> invokeAll(Set<? extends K> keys,
                                              EntryProcessor<K, V, T> entryProcessor,
                                              Object... arguments);


String getName();


CacheManager getCacheManager();


void close();


boolean isClosed();
```

The JCache API

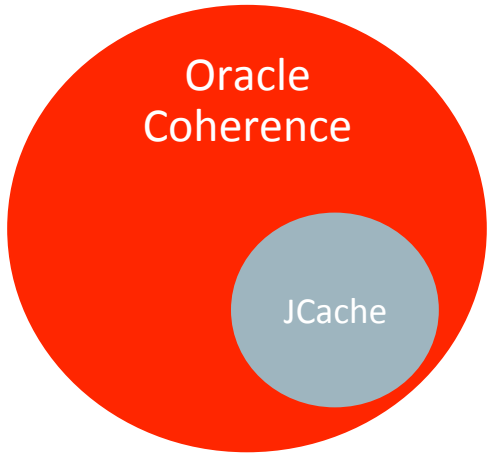
```
<T> T unwrap(java.lang.Class<T> clazz);

void registerCacheEntryListener(CacheEntryListenerConfiguration<K, V> configuration);

void deregisterCacheEntryListener(CacheEntryListenerConfiguration<K, V> configuration);
```

Introduction to JCache

- There's actually a lot more to JCache
  - Support for Multiple Named Caches
  - Support for Multiple Caching Providers
  - Support for Annotation-based Caching (when using Compliant Containers)
  - Support for JMX-based Monitoring
- The specification document contains a detailed overview

Getting Started with JCache and Coherence

- Oracle Coherence 12.1.3 is a **compliant** JCache Provider
  - Supports all JCache APIs and configurations
  - Additionally numerous compliant Caching Topologies (eg: local and partitioned/distributed)
  - Provides “pass-through” configurations to support accessing existing Coherence-native Caches as JCache Caches
- “Essentially JCache is a sub-set of Oracle Coherence”



Getting Started with JCache and Coherence

- **Step 1: Application Dependencies and Class Path Requirements**
  - cache-api.jar (The JCache API)
  - coherence.jar (Oracle Coherence)
  - coherence-jcache.jar (Provides Oracle Coherence JCache Support)

Getting Started with JCache and Coherence

- **Step 2: Configuring JCache for use with Coherence**
  - All applications must specify a Coherence Cache Configuration that enables JCache
  - Approach 1: Use the example Coherence Cache Configuration for JCache

```
-Dtangosol.coherence.cacheconfig=coherence-jcache-cache-config.xml
```

  - Approach 2: Introduce the JCache Namespace into an existing Coherence Cache Configuration

```
<cache-config …
   xmlns:jcache="class://com.tangosol.coherence.jcache.JCacheNamespace">
```

Getting Started with JCache and Coherence

- **Step 3: Acquire a Cache Manager from the Caching Provider**

```
// acquire the default CachingProvider
CachingProvider provider = Caching.getCachingProvider();

// acquire the default CacheManager
CacheManager manager = provider.getCacheManager();
```

Getting Started with JCache and Coherence

- **Step 4: Configure JCache Caches**

```
// define a standard JCache Cache Configuration
MutableConfiguration<String, Integer> config = new MutableConfiguration<>();

config.setStoreByValue(true)
      .setTypes(String.class, Integer.class);


// create the cache
Cache<String, Integer> cache = manager.createCache(”Ages", config);
```

Getting Started with JCache and Coherence

- **Step 5: Use JCache Caches**

```
// acquire a previously configured named cache
Cache<String, Integer> ages = manager.getCache("Ages", String.class, Integer.class);

// put values into the cache
ages.put("Talker", 132);
ages.put("Tonto", 2);

// retrieve a value from the cache
int age = ages.get("Tonto");
```

Comparison of JCache and NamedCache Features

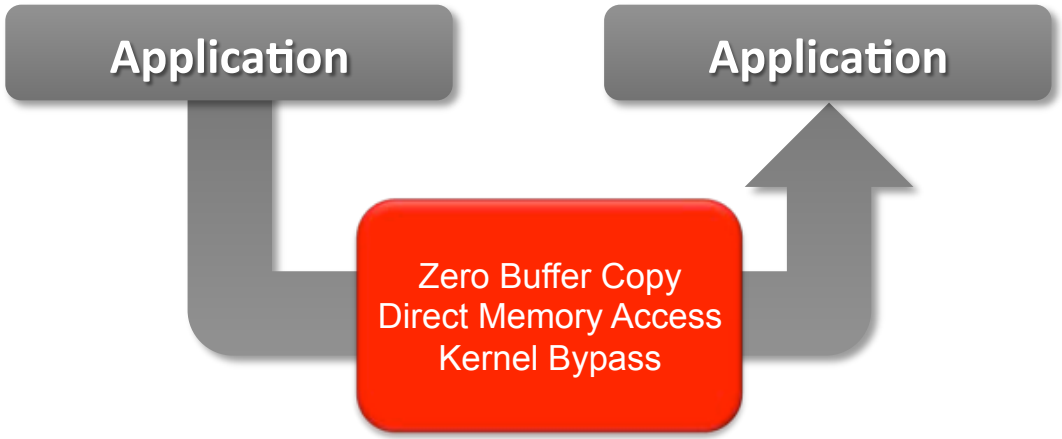
Feature	JCache	NamedCache
Local Cache	X	X
Partitioned (Distributed) Cache	X	X
Replicated Cache		X
Optimistic Cache		X
Near Cache		X
Read Through	X	X
Write Through	X	X
Events	X	X
Query Filters		X
Indexes		X
Entry Processors	X	X
Aggregation		X

More Information

- Oracle Coherence JCache Information:
  - Consult the “Developing Applications with Oracle Coherence” Book
    - http://www.oracle.com/technetwork/middleware/coherence/overview/index.html

Infiniband Message Bus 2.0

- Second generation implementation of the Infiniband MessageBus API for Exalogic
- MQL libraries shared with database and eventually other products
  - Leverages support and tuning investment across ExaData and ExaLogic platforms, and other product suites
- More latency and scalability improvements



4X Throughput
6X Lower Latency

Functionality Improvements

- Added SSL support for reliable transport tmb and sdmb.
- Asynchronous EntryProcessors
  - Better control over parallel execution of EntryProcessors without need to create new threads.
- Elastic Data Eviction Support via high/low watermark
- Async index creation
  - Reduce Mean Time To Recovery by rebuilding the index data structures after partition transfer. Note: any requests dependent on indices will not execute until the indices are rebuilt.
- Added a custom index implementation to avoid repetitive value deserialization.
  - See Javadoc for com.tangosol.util.extractor.DeserializationAccelerator class.

Ease of Use Improvements

- Enhanced NameService so it runs on all cluster nodes (backported to 12.1.2.0.1)
  - Removes requirement to start proxy servers before cache servers on a given machine.
  - http://coherencedownunder.wordpress.com/2014/03/21/using-the-nameservice-in-coherence-12-1-2/

Compatibility Improvements

- Extend clients support both forward and backward compatibility with cluster proxies.
  - That is, extend clients can connect to cluster proxies that have lower or higher version numbers (within a major release).
  - For example, a 12.1.3 extend client can connect to a 12.1.2.0.1 proxy.
  - Extend client backward compatibility is not supported on proxy versions prior to 12.1.2.0.1, including 12.1.2.0.0 and proxy versions 3.7.1 or earlier.

Join the Coherence Community

@OracleCoherence

/OracleCoherence

/OracleCoherence

Oracle Coherence Users

blogs.oracle.com/ OracleCoherence

Visit us at: coherence.oracle.com

ORACLE

ORACLE

FUSION MIDDLEWARE

The Foundation for Innovation

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®