ORACLE®

Scaling Java EE Applications with Coherence*Web

Dave Felcey Coherence Product Manager
Michael Braeuer Principal Sales Consultant

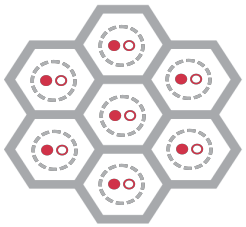
Agenda

- What is Coherence?
- What is Coherence*Web and how does it work?
- What is ActiveCache?
- How can it help to scale Java EE applications?
- Are there any other benefits of Coherence*Web?

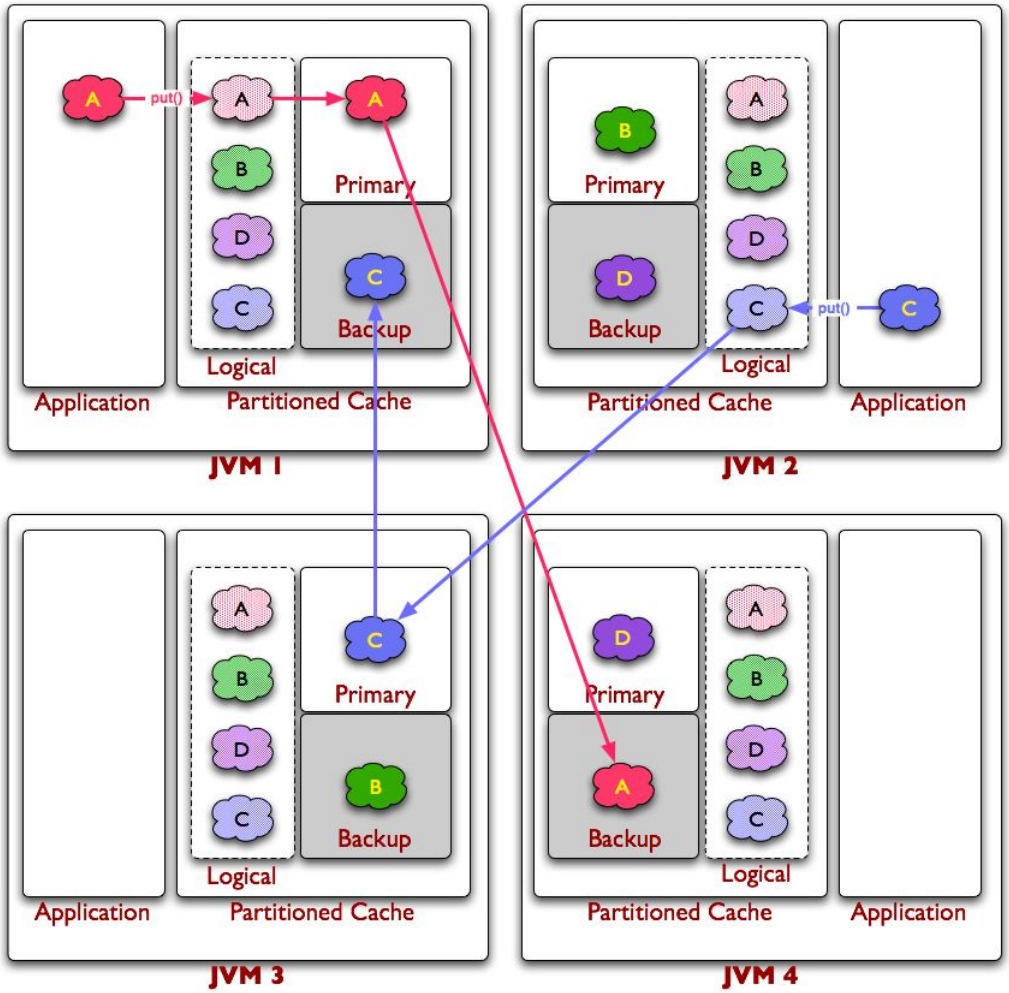
What is Coherence?

What is Coherence?

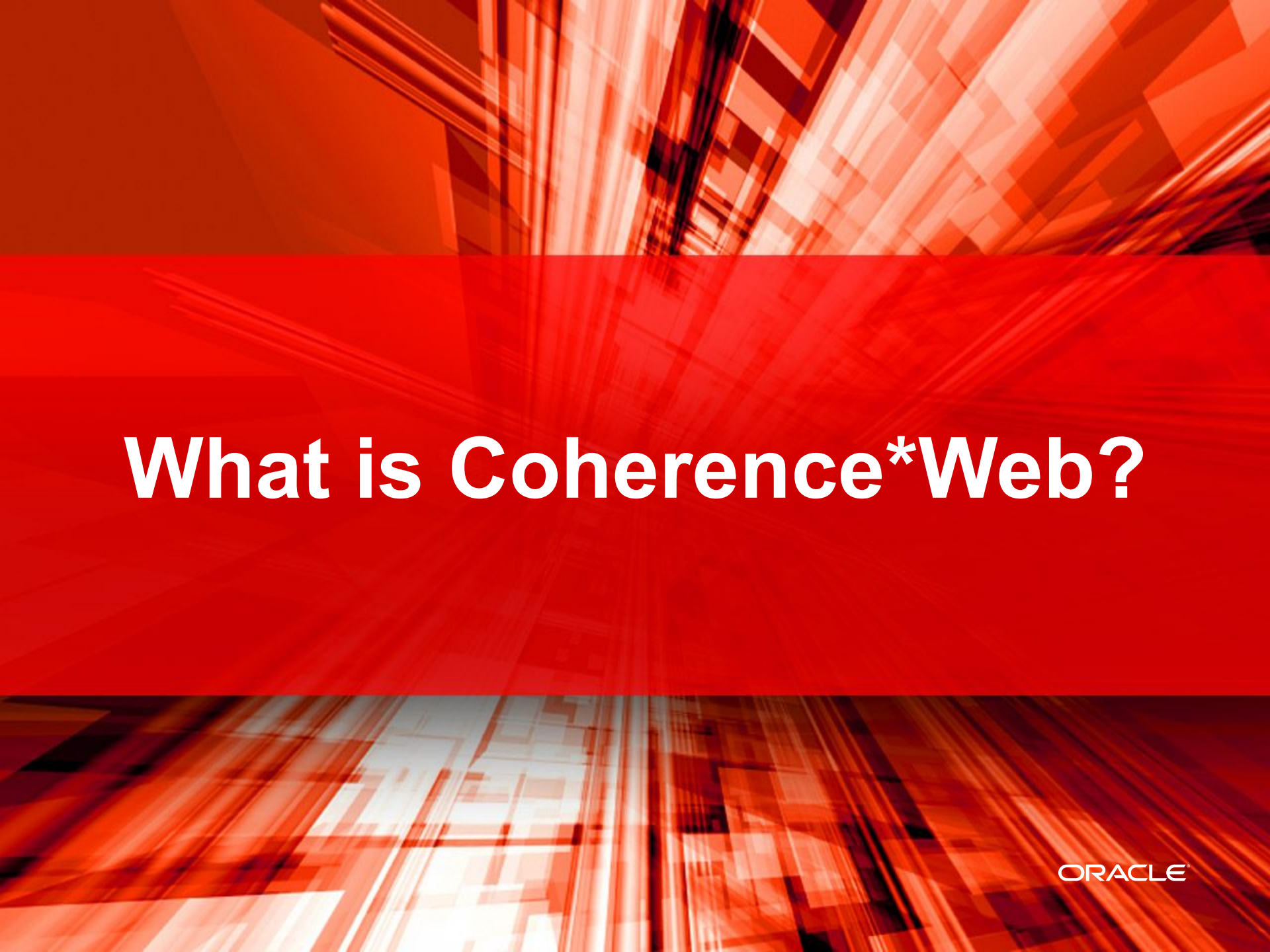
- Java Application
- Mesh Architecture



- Key/Value memory store
- No SPOF or SPOB
- Peer-to-Peer communications
- Java, .NET and C++ API's
- 'Self-Healing'
- Can support 100's of nodes
- Optimized transaction processing
- Supports events, queries and aggregations



What is Coherence*Web?



What is Coherence*Web?

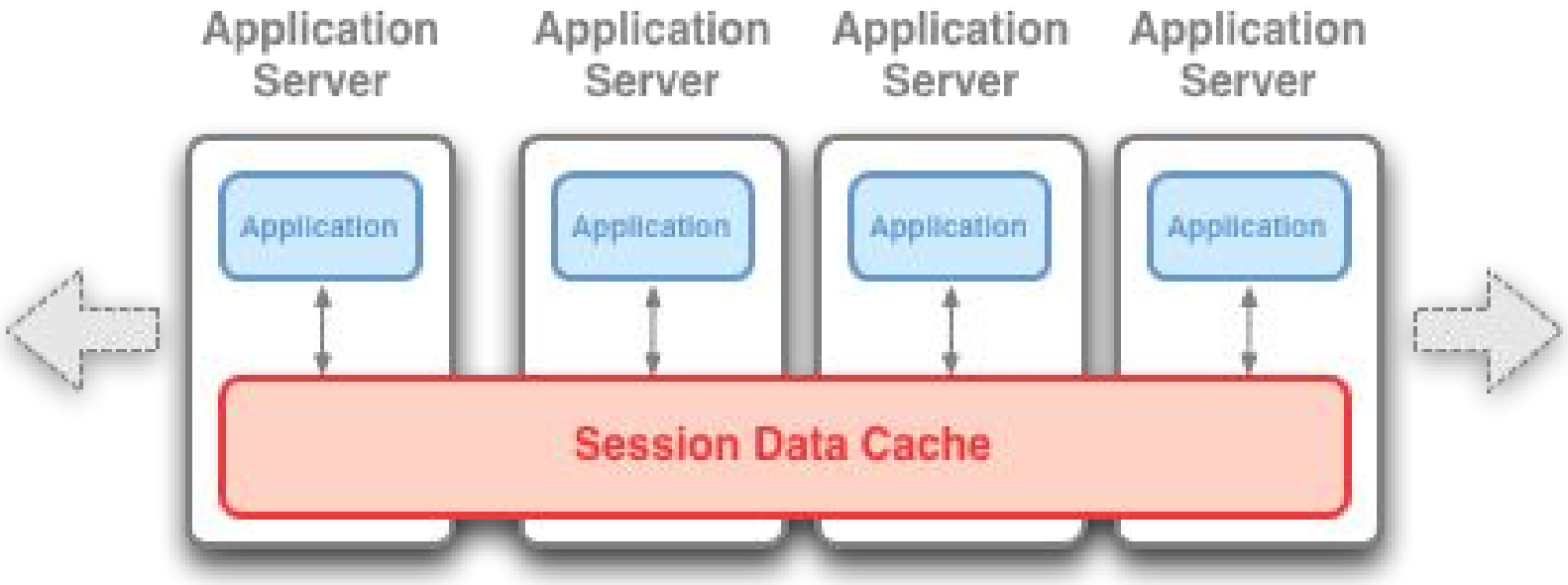
- Coherence*Web is an HTTP session management module
- It is a drop-in replacement for native container session management
- Tested and certified for all major web containers
  - WebLogic – Native Coherence*Web SPI installation
  - Glassfish – Native Coherence*Web SPI installation
  - WebSphere
  - OC4J
  - JBoss/Tomcat/Jetty
  - And others!

How does Coherence*Web work?

- Coherence*Web “wraps” existing web applications
  - No runtime byte code manipulation is done
- Any requests to use sessions (from servlets, JSPs, filters, etc) are intercepted by Coherence*Web wrappers
- Some web containers require patches to work with Coherence*Web
  - Coherence*Web extends container classes, which sometimes are declared final
  - These patches modify internal container classes to make them extendable by Coherence*Web

Deployment Models

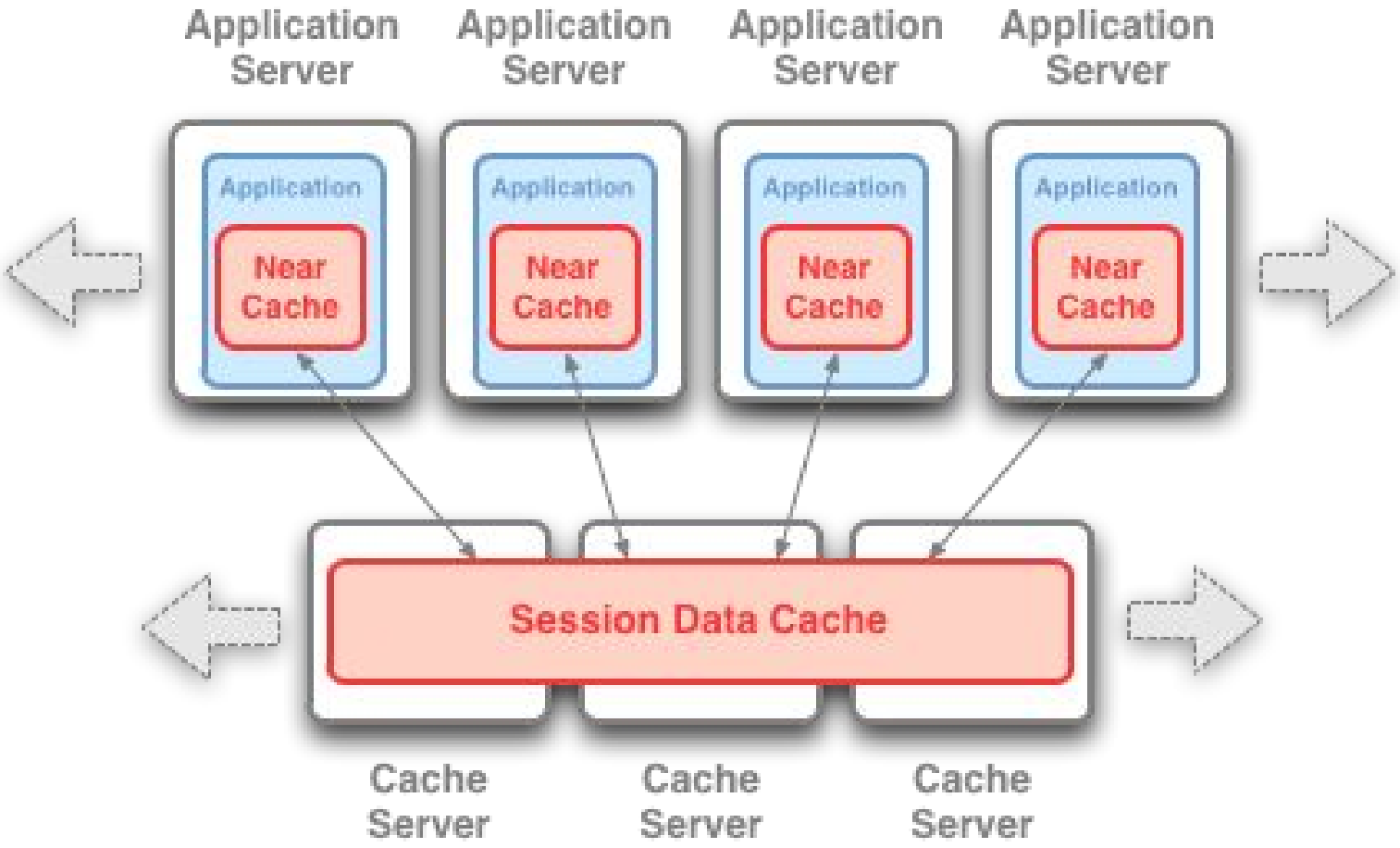
Deployment Models – In-Process



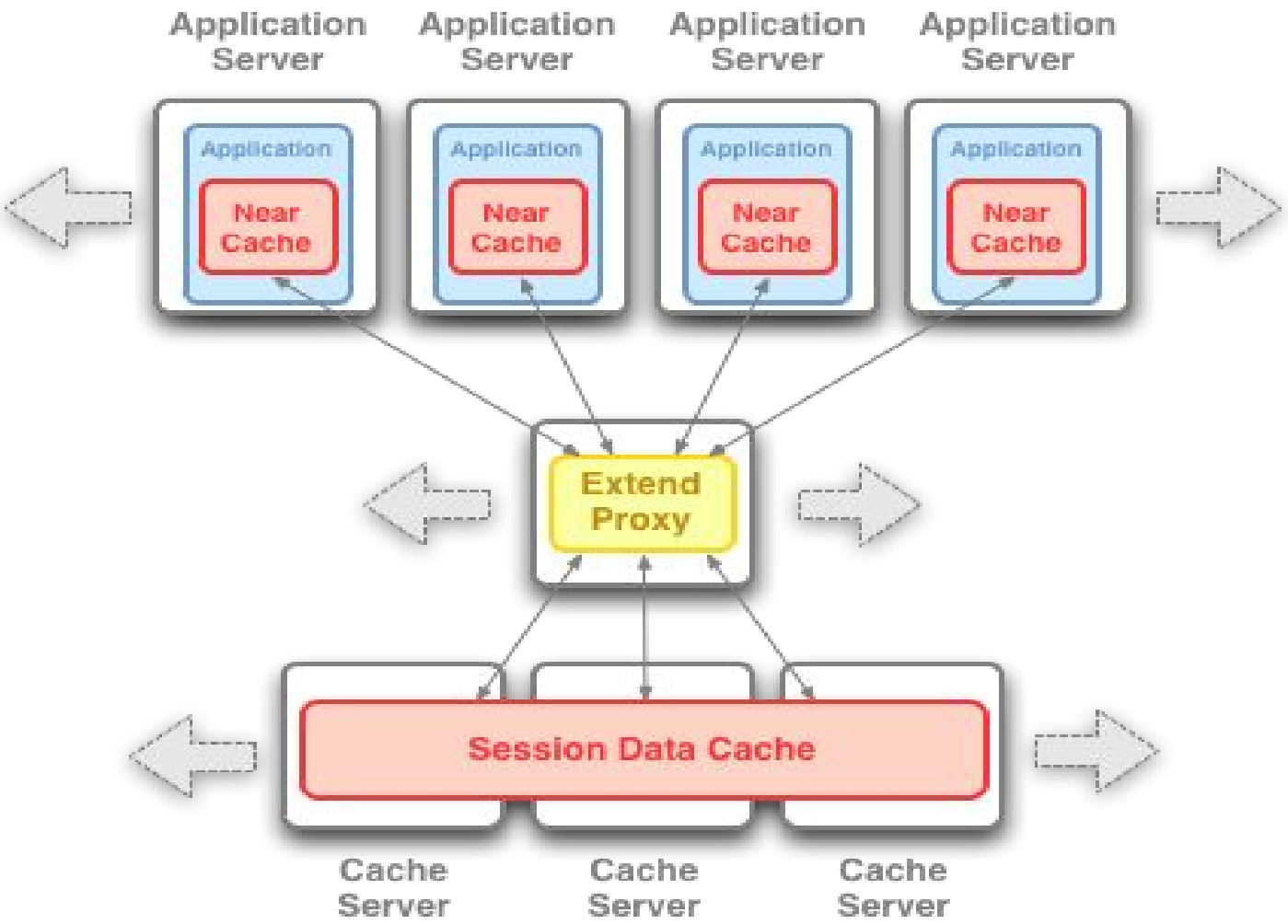
Default ONLY for ease of eval, demonstration and smoke-testing.

DO NOT USE IN PRODUCTION!

Deployment Models – Out-of-Process

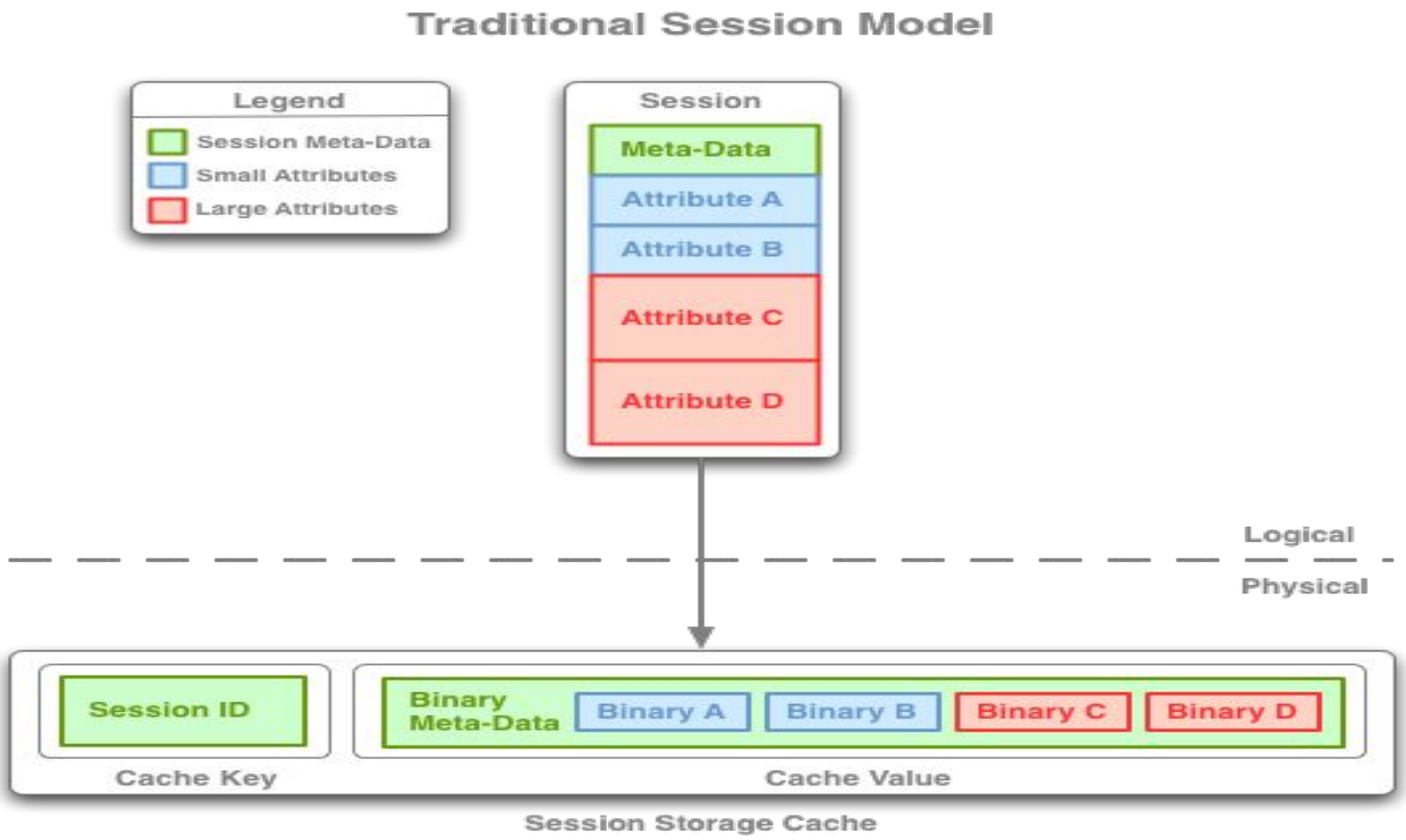


Deployment Models – In-Process

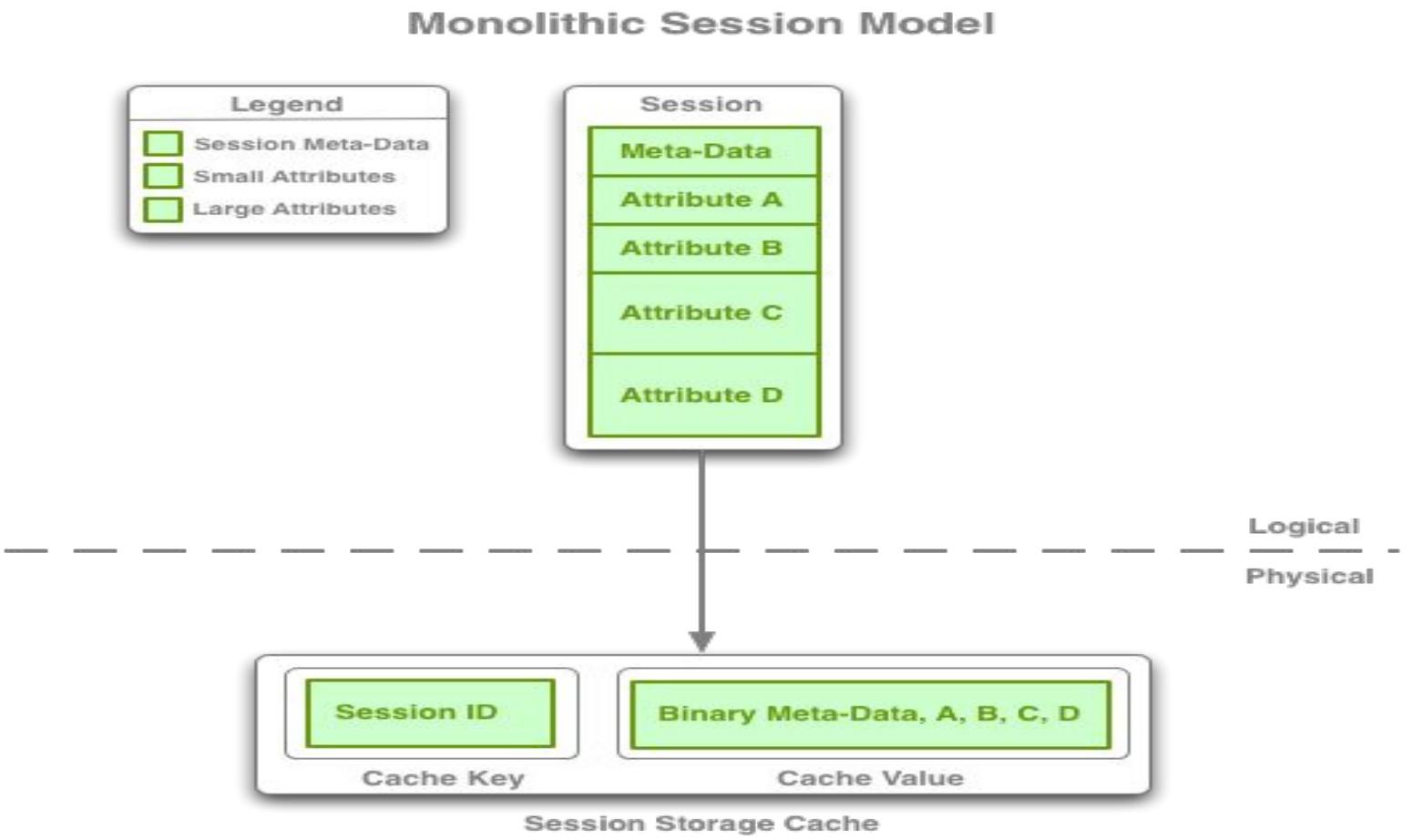


Session Models

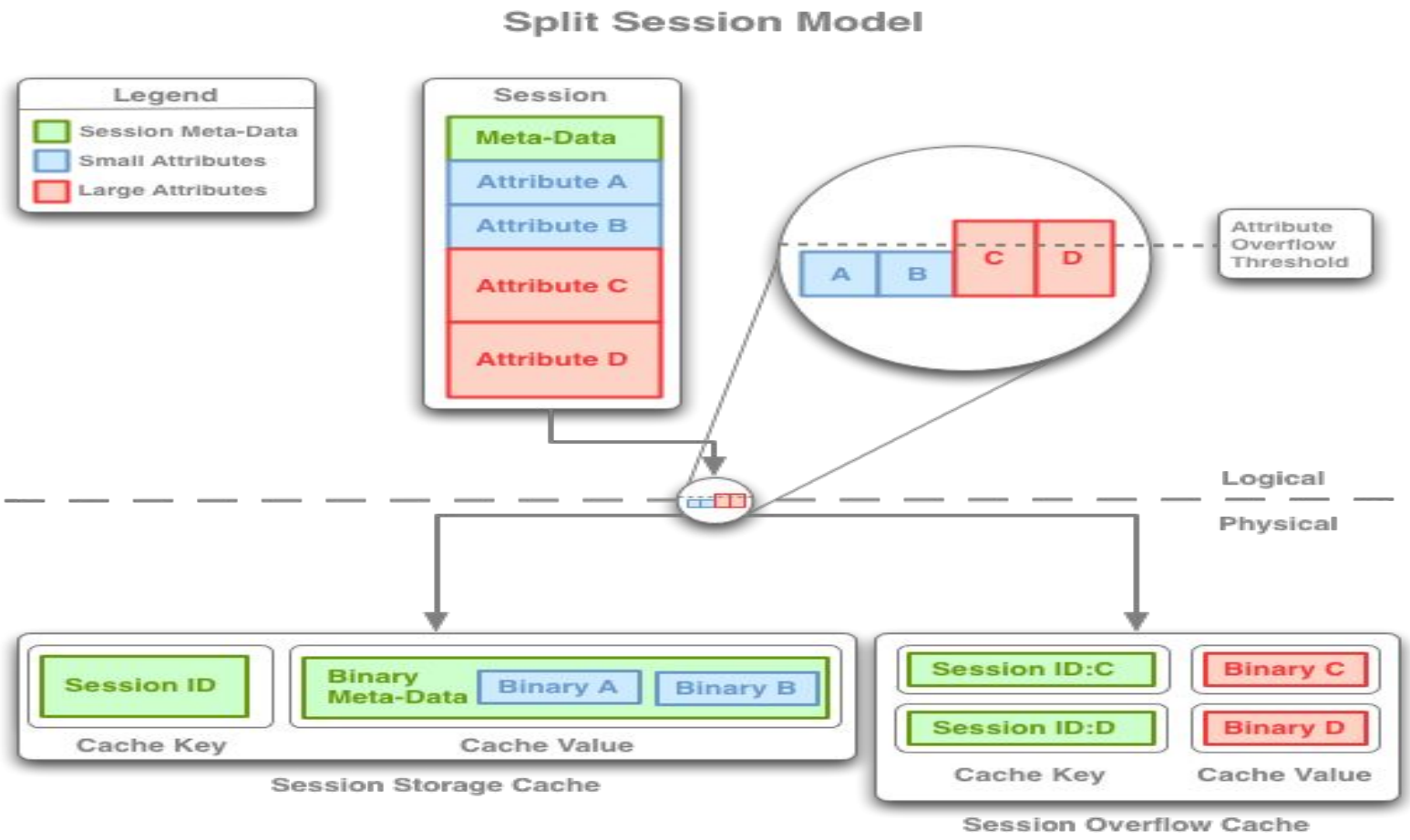
Session Models - Traditional



Session Models - Monolithic



Session Models - Split



Locking Models

Locking Models

- **Lock Free (default)**

  Allows multiple nodes in a cluster to access an HTTP session simultaneously.

- **Member Locking**

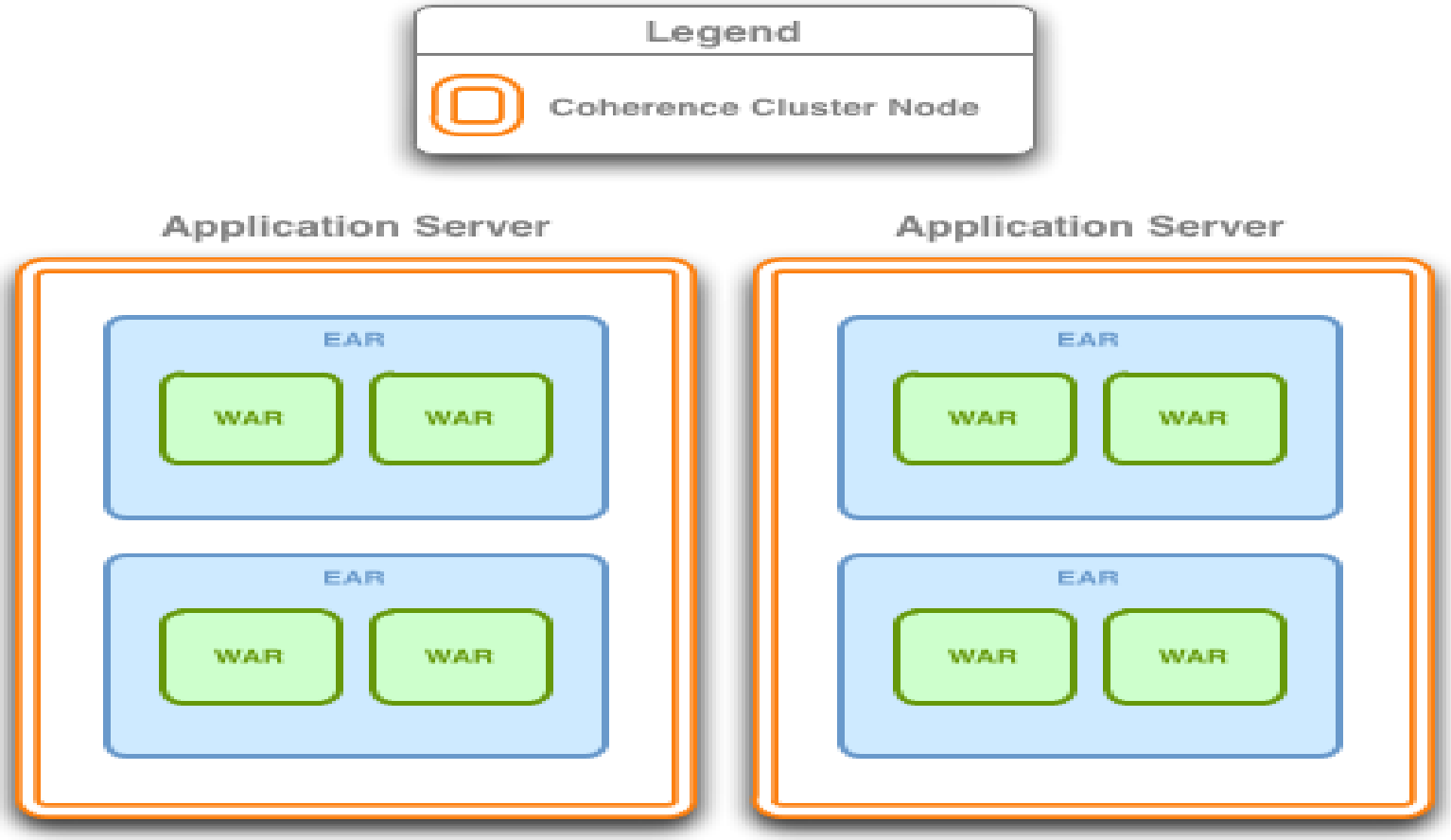
  Does not allow more than one node in the cluster to access an HTTP session.

- **Thread Locking**

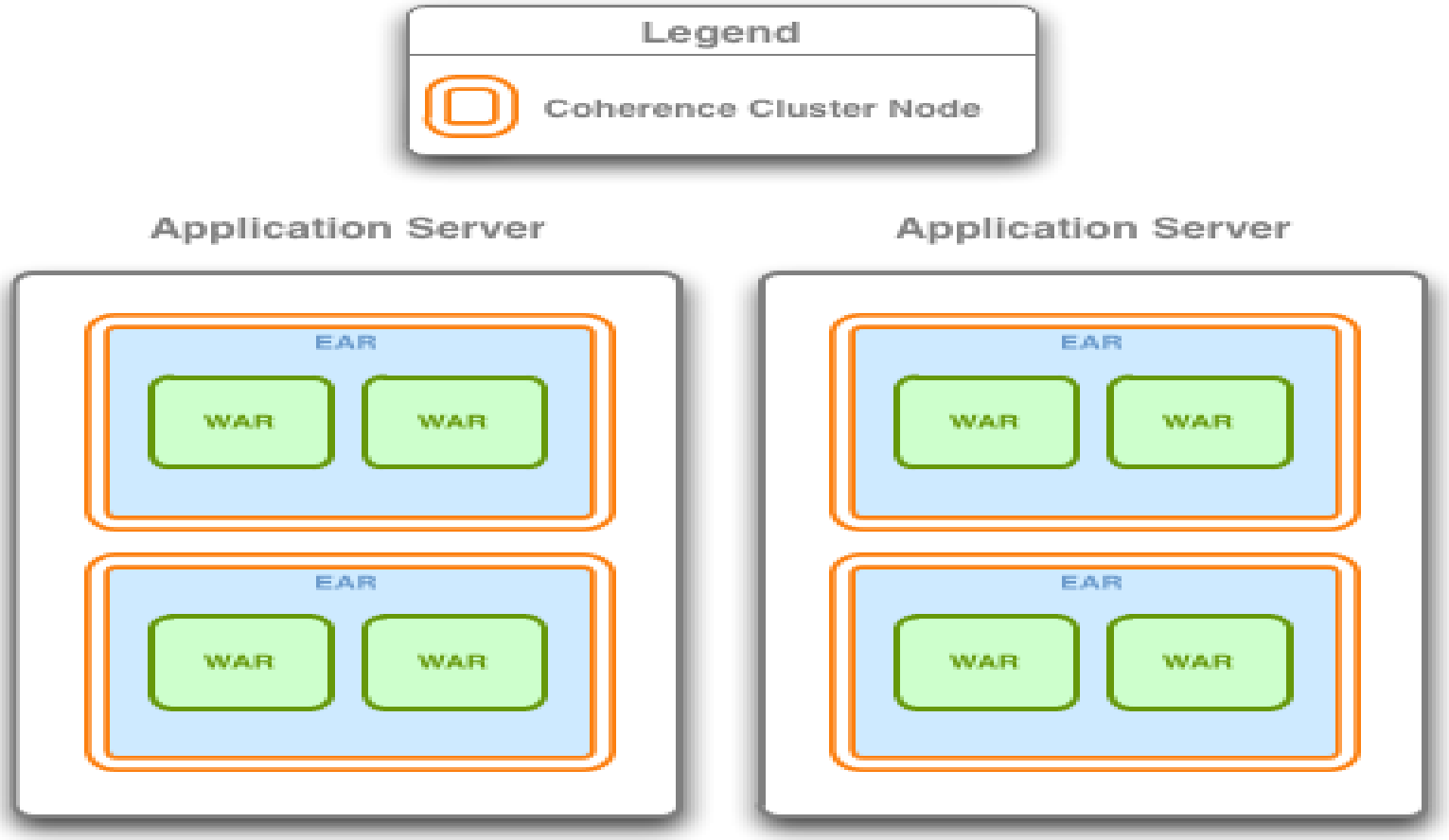
  Does not allow more than one thread in the cluster to access an HTTP session.

Cluster Node Isolation

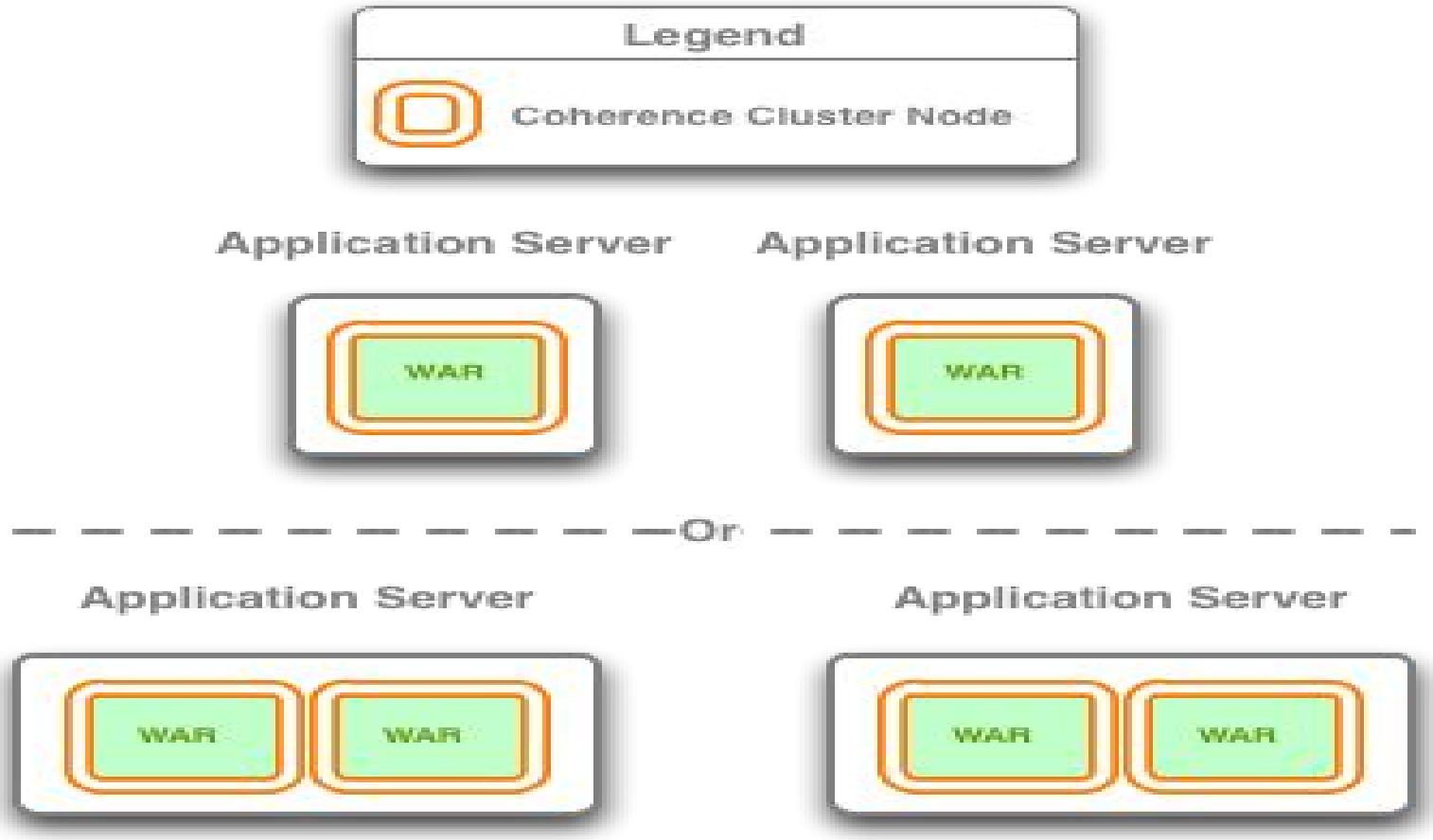
Cluster Node Isolation – App Server Scoped



Cluster Node Isolation – EAR Scoped



Cluster Node Isolation – WAR Scoped



Session and Session Attribute Scoping

Session and Session Attribute Scoping

- Session Scoping
  - Coherence*Web allows session data to be shared by different Web applications deployed in the same or different Web containers.
- Session Attribute Scoping
  - Extension of Session Scoping allowing for scoping of individual session attributes so that they are either globally visible or scoped to an individual web application
  - Behavior is controllable via the AttributeScopeController interface.
    - Two out of the box implementations: ApplicationScopeController and GlobalScopeController

Installing Coherence*Web

Web Installer

- Make sure patch is installed for your web container if it is required
- Run the inspector on the existing WAR/EAR file
  - This generates a coherence-web.xml configuration file
  - This file wraps all servlets, filters, etc with Coherence implementations
  - It also contains configuration settings for Coherence*Web
- Inspect and (if any changes are required) modify the coherence-web.xml file
- Run the installer process on the existing WAR/EAR
  - This generates a new WAR/EAR and backs up the original
  - Deploy the WAR/EAR

What is ActiveCache?

ActiveCache and WebLogic

- Integration of Coherence and WebLogic Server
  - Incremental progress since launch of 11g
- The whole is greater than sum of its parts!
  - Coherence*Web SPI Support for HTTP Sessions
  - Dependency Injection
  - Configuration, Lifecycle and and Monitoring of Coherence Clusters and Servers

active-cache-1.0.jar

- Shipped with WebLogic Server distribution
- Required for advanced WebLogic Server and Coherence integration
- Manage Coherence configuration with WLS MBeans and WLS Console
- Dependency Injection in Java EE modules
- Manages Coherence Lifecycle in Java EE modules
- How to reference from applications:
  - EAR scope - Import into the Java EE application as a shared library jar in weblogic-application.xml
  - WAR scope - Import as an optional package via META-INF/manfiest.mf
  - Server scope – reference this jar from the system classpath

active-cache-1.0.jar (continued)

- META-INF/manifest.mf file uses relative paths to refer to other WLS libraries, so refer to this jar in the location: $WLS_HOME/common/deployable-libraries
- If you copy it out of that location it will not work

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.7.1
Created-By: R28.0.2-3-133398-1.6.0_20-20100512-1453-windows-ia32 (Orac
 le Corporation)
Specification-Title: active-cache
Specification-Version: 1.0
Implementation-Title: active-cache
Implementation-Version: 1.0
Extension-Name: active-cache
Class-Path:
@BEA_HOME@/modules/features/weblogic.server.modules.coherence.integration_10
.3.4.0.jar:
../../../modules/features/weblogic.server.modules.coherence.integration_10.3
.4.0.jar
```

coherence.jar

- Shipped with Coherence distribution in $COHERENCE_HOME/lib
- Contains core Coherence classes
- How to reference from applications:
  - EAR scoped - reference as a shared library in weblogic-application.xml (recommended) or embed in APP-INF/lib
  - WAR scoped - embed in an application in WEB-INF/lib
  - Server scoped - can be put on the system classpath

coherence-web-spi.war

- Shipped with Coherence distribution in $COHERENCE_HOME/lib
- Contains WebLogic Server SPI implementation for HTTP Session storage in Coherence
- Reference as a shared library in web module WEB-INF/weblogic.xml
- Contains default cache configuration for session storage WEB-INF/classes/session-cache-config.xml (can be overridden via classpath)
- Local storage defaults to false by default

Node Manager and Coherence Servers

- Coherence Servers need classpath to include $MW_HOME/modules/features/weblogic.server.modules.coherence.server_10.3.4.0.jar
- META-INF/manifest.mf contains relative references to other WebLogic Server libraries, so refer to it in the default location and do not copy it
- If no classpath entries are specified in Startup tab, this jar and coherence.jar are added by default implicitly by Node Manager

ActiveCache and Oracle GlassFish Server

- Integration of Coherence and GlassFish
  - For commercial distribution, not open source edition
- Coherence*Web support for HTTP Sessions

+

Getting Started

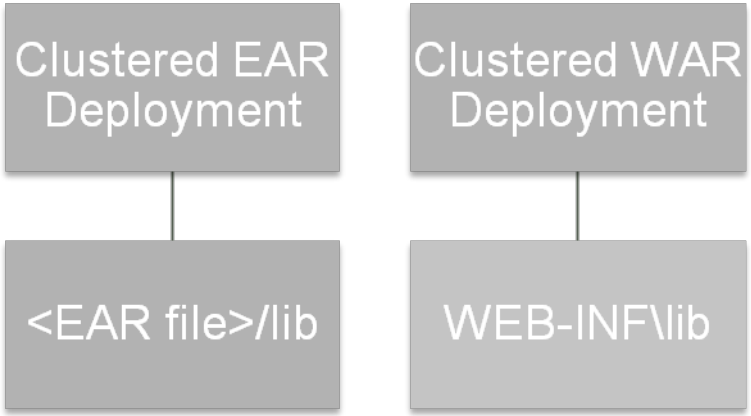
- Configure your session to use Coherence*Web
  - Edit glassfish-web.xml

```
<glassfish-web-app error-url="">
<Session-config>
  <session-manager persistence-type="coherence-web" />
</session-config>
…
</glassfish-web-app>
```

  - Advanced configuration options available in glassfish-web.xml
    - Defaults will handle most web applications
    - Additional <context-params> elements will override or add new configuration
  - In web.xml: use <distributable/>, otherwise Glassfish's standard session management (1 JVM, in-memory) is applied

Preparing your GlassFish application(s)

- Configuration changes do require restart of Web App
- Extract coherence-web.jar and session-cache-config.xml from coherence\lib\webInstaller.jar/web-install
- Copy session-cache-config.xml to WEB-INF\classes
- If clustered, copy coherence.jar to appropriate location



Example: preparing your GlassFish Server

- Example: clustered, in-process deployment model
- Create/configure domain
  - Use asadmin at command line to create and start
  - Use GF console to add appropriate JVM options
    - coherence –D JVM arguments
- Example JVM options
    - Well known addresses (as opposed to multicast)
    - If multi-homed system that has multiple IPs, which IP to bind to
    - Tell glassfish to use cache servers for storage, not the in the GlassFish JVM, e.g.
    - Session.localstorage=false (ensuring use of coherence cache server, not GF)

Example (continued)

- Example: clustered, in-process deployment model
- Configure load balancer (ex. Apache)
  - Configure JVM for cluster or standalone servers to use
  - Configure / Enable mod_jk.conf load balancer plug-in
    - Mount paths defined in workers.properties
  - Edit workers.properties list to accommodate cluster members
  - Enable mod_jk for GlassFish
    - –Djvmroute option for routing to LB
    - JVM -D option for location of workers.properties &
    - In GF console, enable JK network listener for Apache
      - for AJP protocol

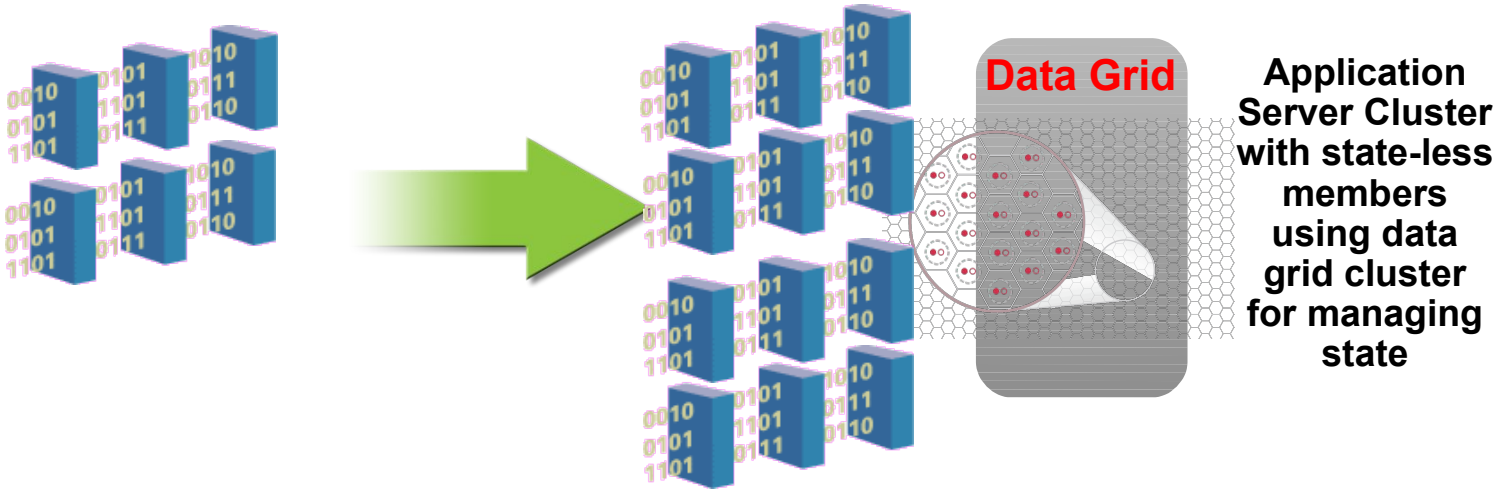
How can it help scale J2EE applications?

How does Coherence*Web help scale J2EE Applications?

- Application can handle more users without adding more application servers
- Handle very large sessions – Elastic Data
- Keep session data coherent under heavy load
- Reduce application server heap size and so impact of JVM GC's or pauses – for instance if have a lot of abandoned sessions

Seamlessly Scale on-demand

Application Server Cluster with multiple state-full JVM's



Application Server Cluster with state-less members using data grid cluster for managing state

Challenges

- Application load can be very variable
- Adding more application servers can result in lost-sessions
  - Lost sessions can mean lost sales
  - Scaling cluster needs to be planned

Benefits

- Keep application state outside the application server;
  - Prevents no service interruptions
  - Reduces effort in adding more capacity
  - Dynamically add capacity with no outages

ON24

- Supplier of Enterprise WebCasting and hosts Virtual Events
- Supplier to wide range of large organizations
- Provides global distributions

Scenario	Problem	Solution
· Live WebCasting needs to be very reliable · Performance and scalability key · Maintenance hard to manage	· Challenge providing the highest quality uninterrupted service · Live traffic occasionally need to be switched from primary cluster into a backup cluster	· Coherence*Web provided common cache for HTTP state across cluster for seamless failover · Coherence*Web reduced application server pauses and heap fragmentation

Are there any other benefits of Coherence*Web?

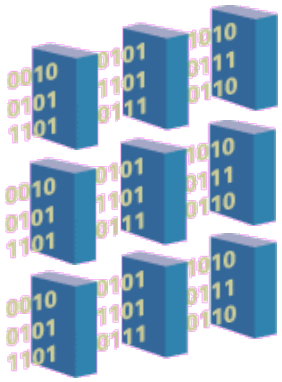
ORACLE

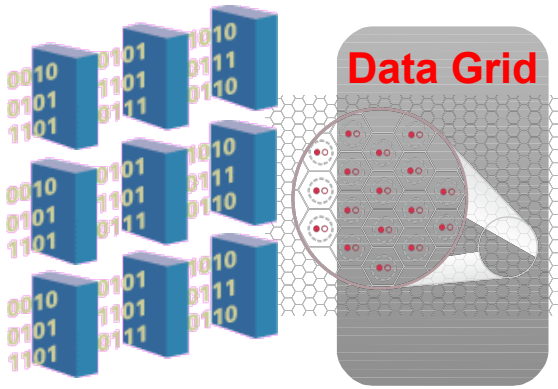
What are the other benefits of Coherence*Web?

- Seamless integration of separate applications using common session state
- Applications/containers can be restarted/maintained without loosing sessions
- Keep session data coherent under heavy load
- Decouple session management from web container

Seamless Integration

Application Server Cluster with multiple JVMs / Managed





Application Server Cluster with less members and data grid cluster connected

Challenges

- Application data can overwhelm available server memory.
- Adding more application servers only compounds problem.
  - Each app server requires fixed overhead.
  - Memory management issues.

Benefits

- Keep application data outside the application server.
- Instant 25% increase of users per WLS application.
- Scale application data linearly thru commodity hardware.
- Increase availability thru data redundancy.

GAP

- One of the largest clothing retailer
- 3100 Stores
- FY2006 revenue $16 Billion
- Four nationally recognized brand names

Scenario	Problem	Solution
· Online retail store · High availability of the web site, shopping cart 	· Loss of database connections means lost shopping carts · Customers needed to re-enter lost items after failure, impacting customer satisfaction · Slow on-line store performance increased customer frustration	· Coherence eliminates lost connections to the back-end systems · Customers can shop between different brands using a single unified check out experience · Customer data now available in memory for innovative features

Summary

- Coherence*Web can enhance J2EE application scalability without application changes
- Coherence*Web is a proven and mature technology
- Coherence*Web provides a host of additional benefits – like session sharing, transparent session failover etc.

Q&A