JDD
Java Developers' Day 2010

Advanced HTTP session management with Oracle Coherence

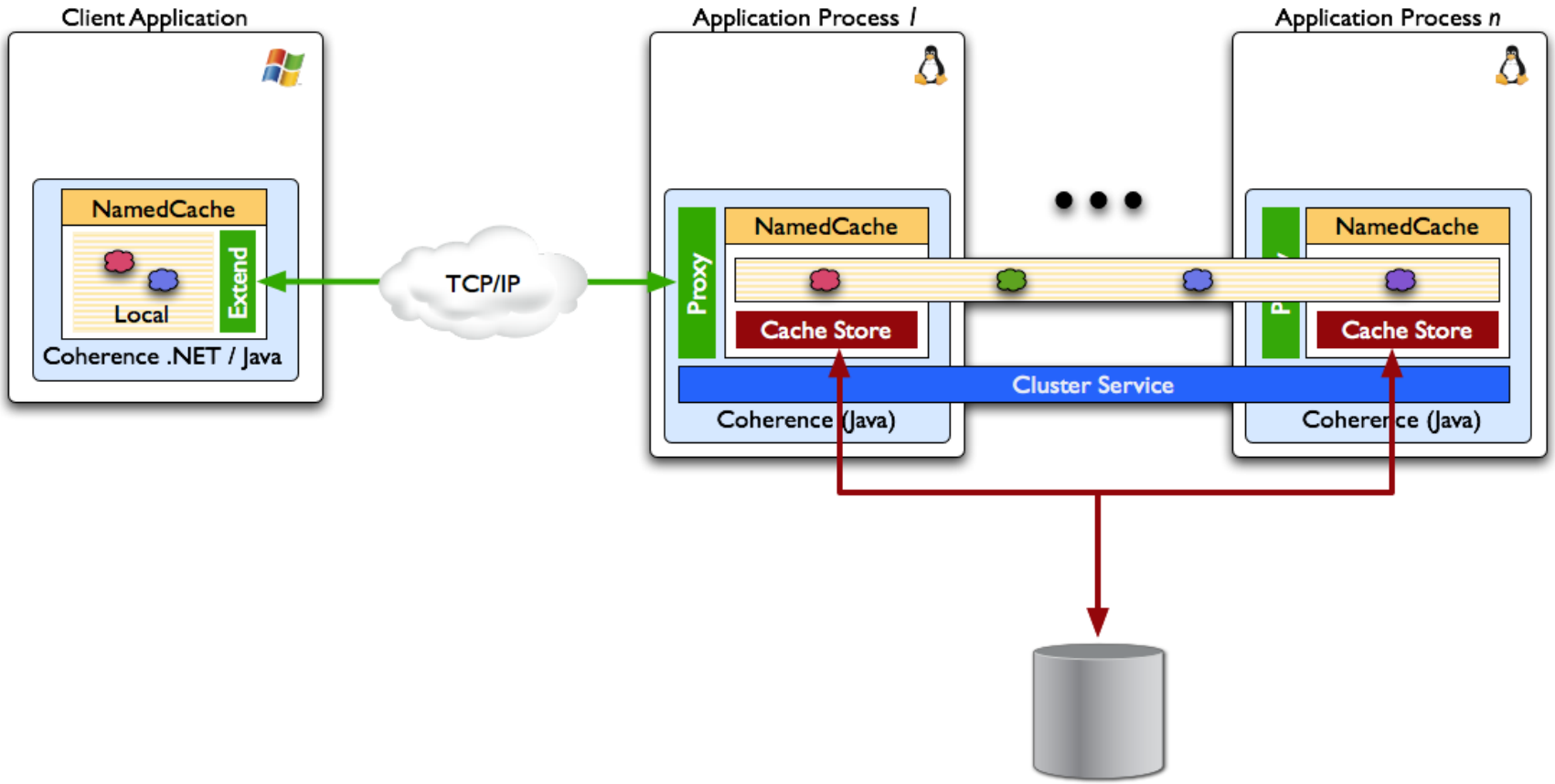
Michał Kuratczyk

principal solution architect, Oracle

Oracle Coherence

- distributed
- in-memory
- key-value
- NoSQL
- data grid
- for Java, .NET and C++ objects
- with distributed data processing
- and HTTP session management capabilities

Architecture



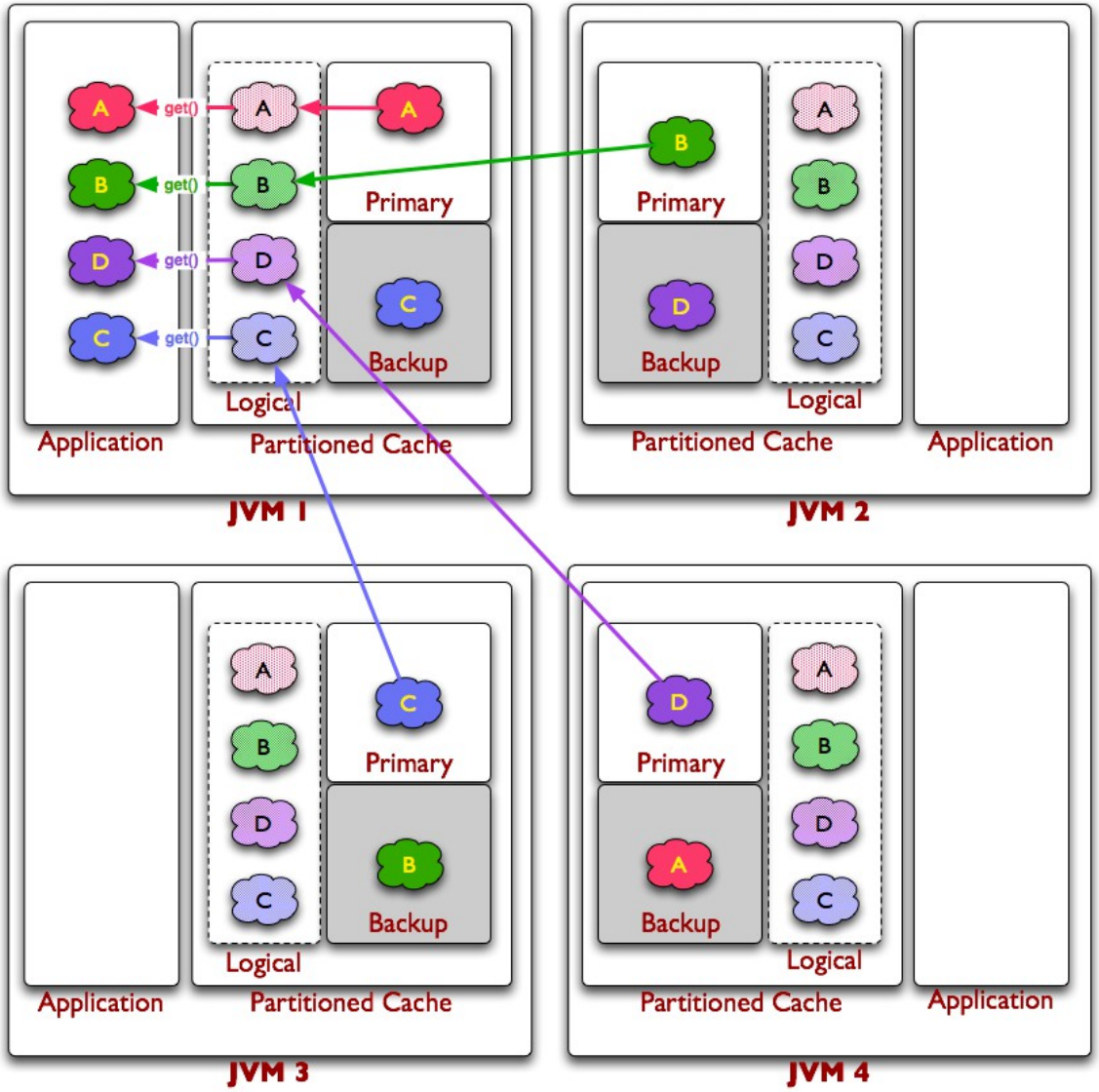
Partitioned Topology : Data Access

Data Access Topologies

- Coherence provides many Topologies for Data Management
- Local, Near, Replicated, Overview, Disk, Off-Heap, Extend (WAN), Extend (Clients)

Partitioned Topology

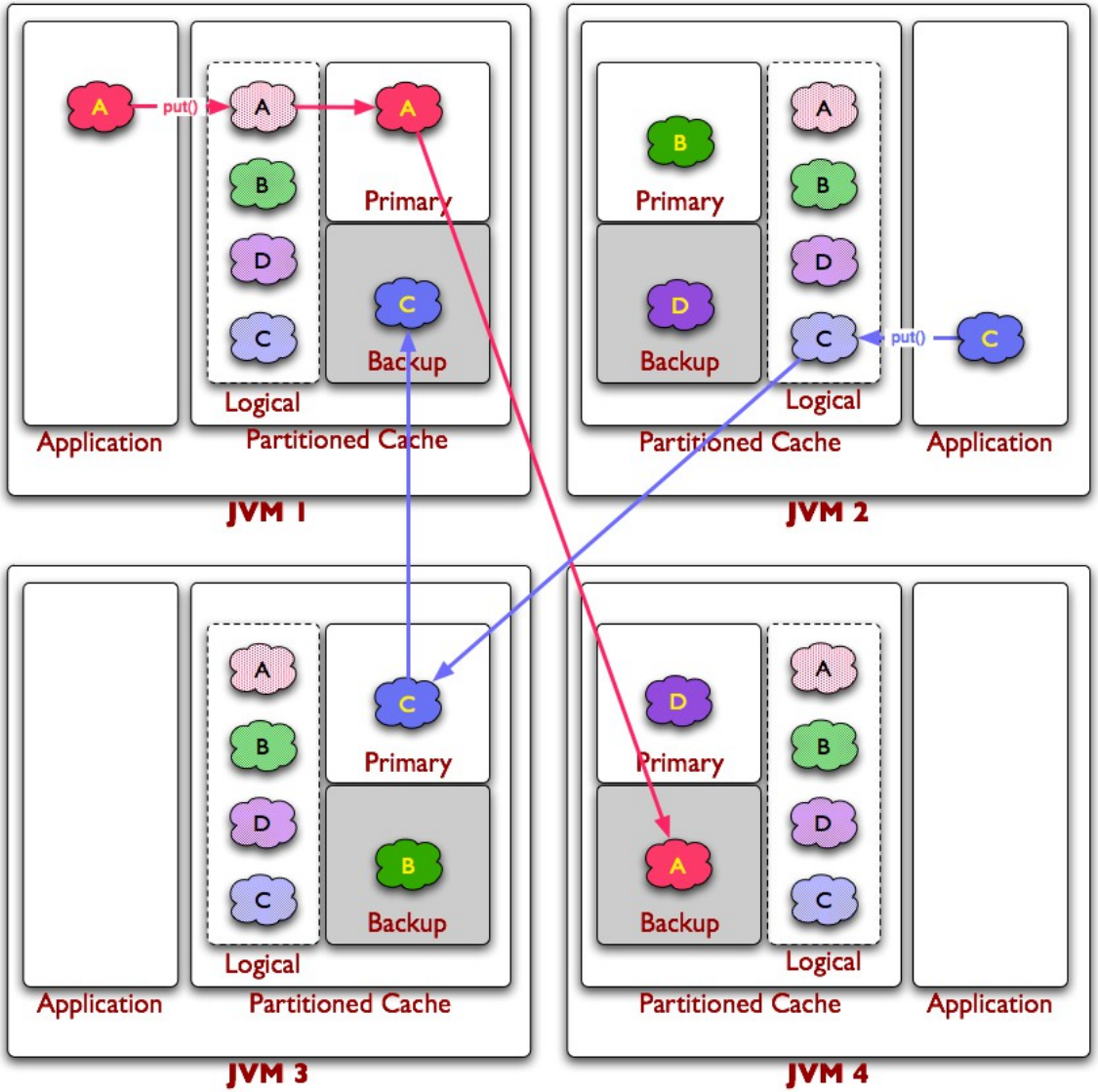
- Data spread and backed up across members
- Transparent to developer
- Members have access to all Data
- All Data locations are known – no lookup & no registry!



Partitioned Topology : Data Update

Partitioned Topology

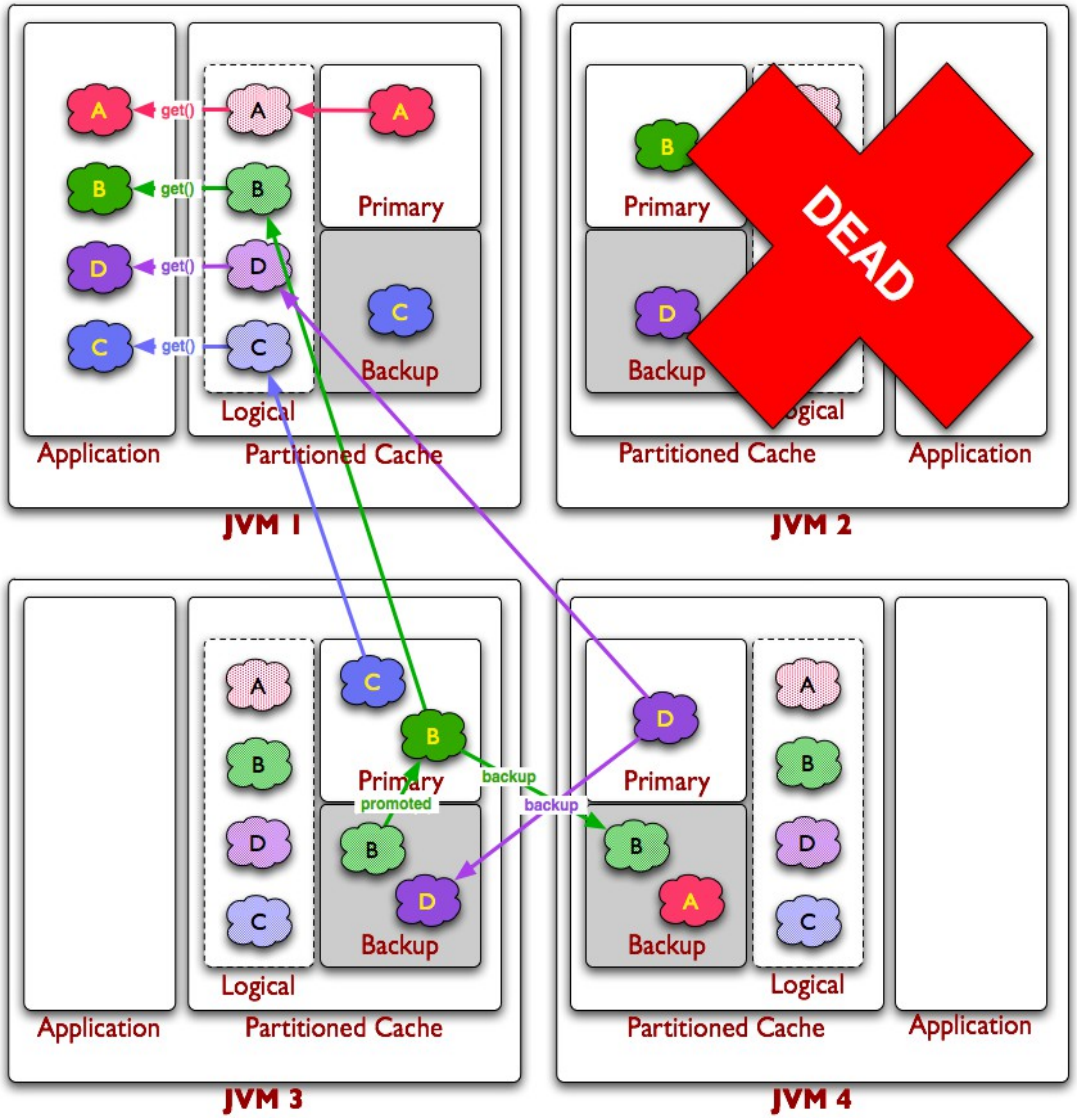
- Synchronous Update
- Avoids potential Data Loss & Corruption
- Predictable Performance
- Backup Partitions are partitioned away from Primaries for resilience
- No engineering requirement to setup Primaries or Backups
- Automatically and Dynamically Managed



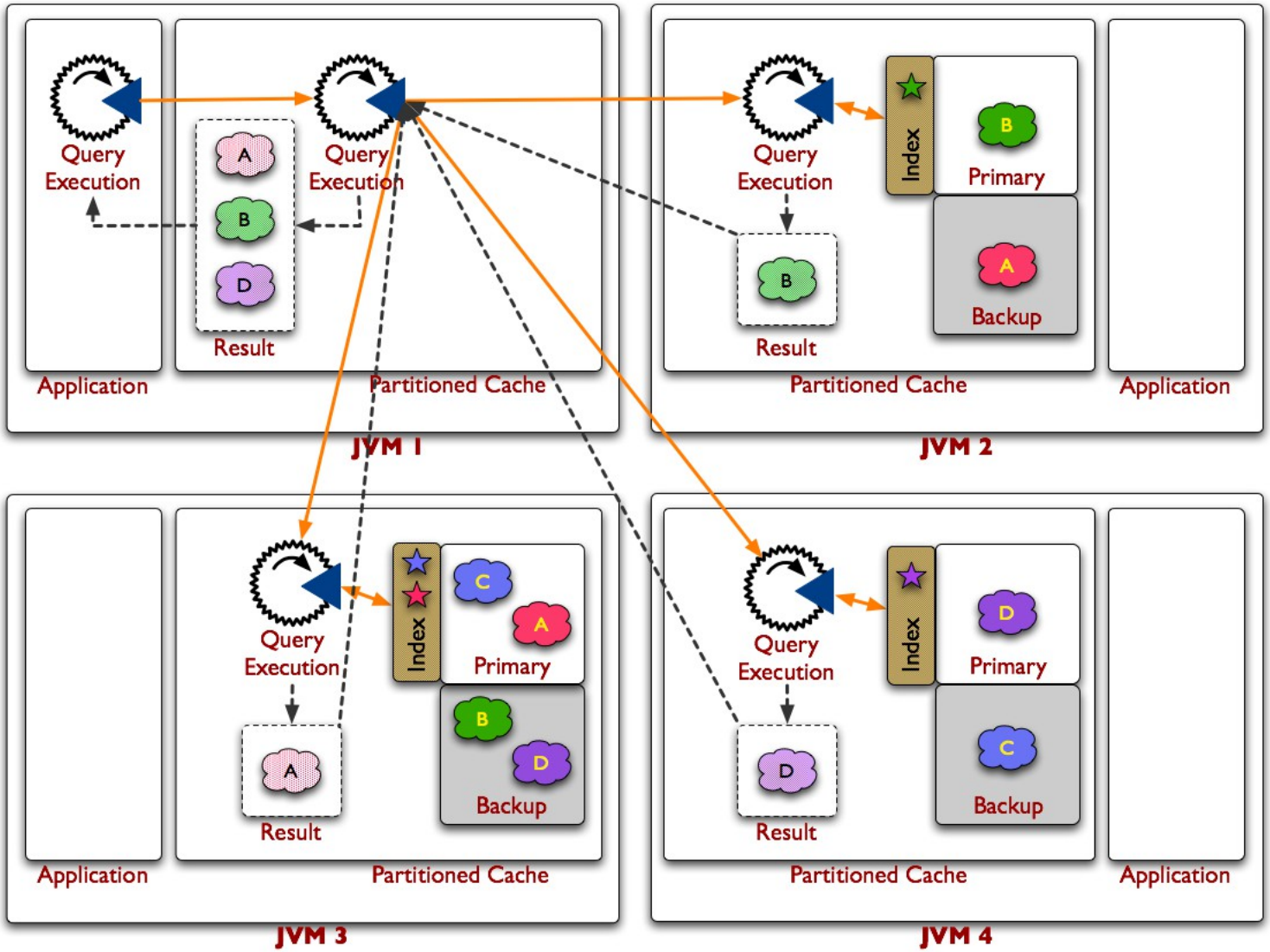
Partitioned Topology : Recovery

Partitioned Topology

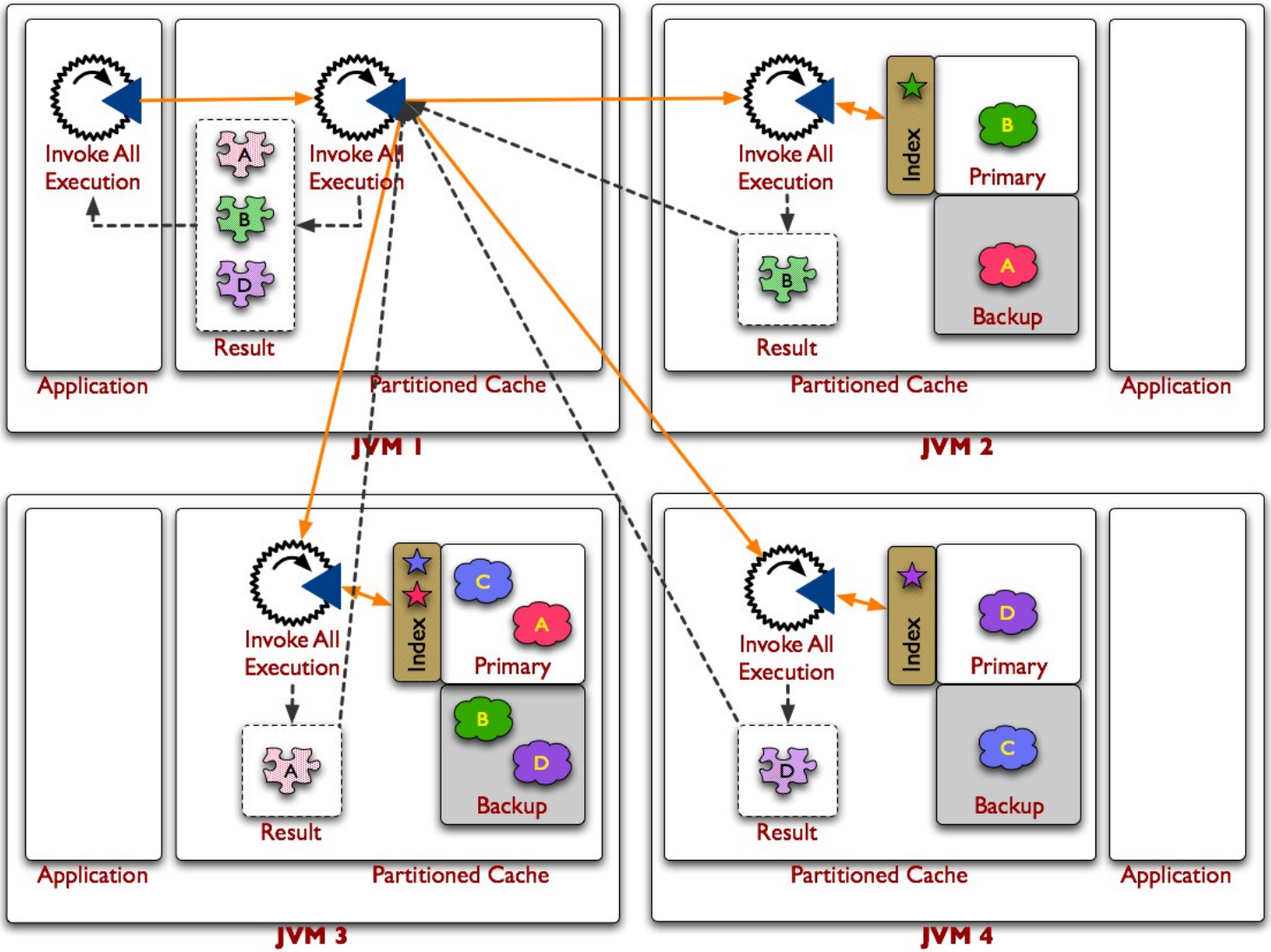
- Membership changes (new members added or members leaving)
- Other members, in parallel, recover / repartition
- No in-flight operations lost
- Some latencies (due to higher priority of recovery)
- Reliability, Availability, Scalability, Performance are the priority
- Degrade performance of some requests



Features : QueryMap Interface



Features : InvocableMap Interface



Coherence*Web

- Coherence*Web is an HTTP session management module
- It is a drop-in replacement for native container session management
- Works with all major web containers
  - WebLogic
  - WebSphere
  - OC4J
  - JBoss/Tomcat/Jetty
  - Glassfish
  - and others

Why Coherence*Web

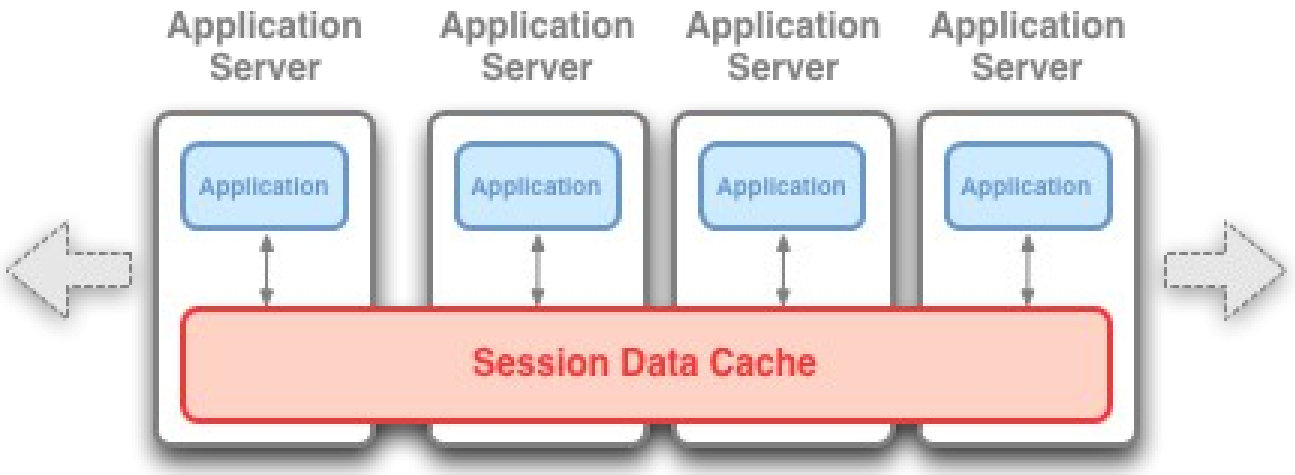
- Decouple session management from web container
- Application can handle more users without adding more application servers
- Applications/containers can be restarted/maintained without loosing sessions
- Handle very large sessions
- Keep session data coherent under heavy load
- Share session between applications
- Share session between application servers

How does Coherence*Web work?

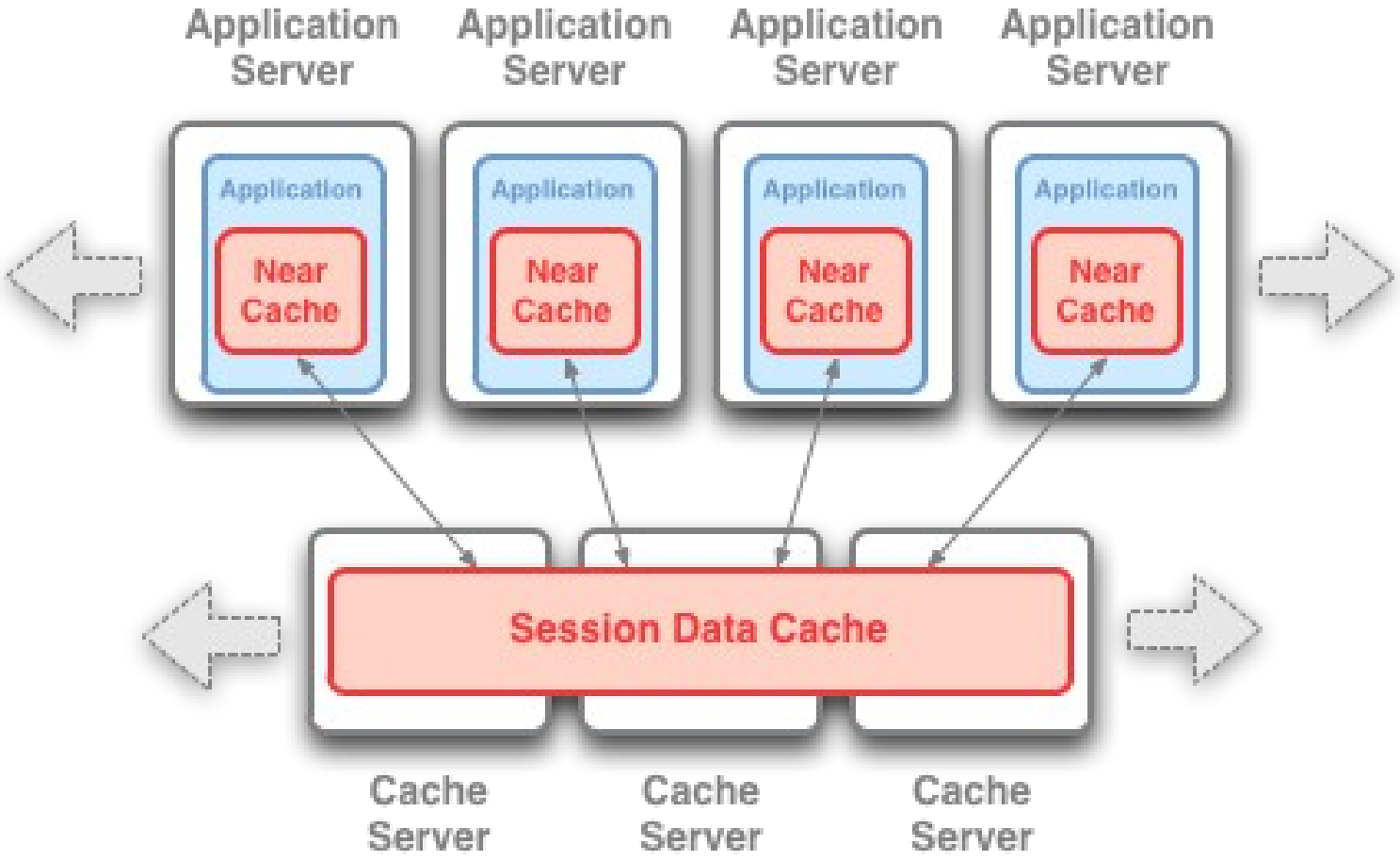
- Coherence*Web “wraps” existing web applications
  - No runtime byte code manipulation is done
- Any requests to use sessions (from servlets, JSPs, filters, etc) are intercepted by Coherence*Web wrappers
- Some web containers require patches to work with Coherence*Web
  - Coherence*Web extends container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

Deployment Models

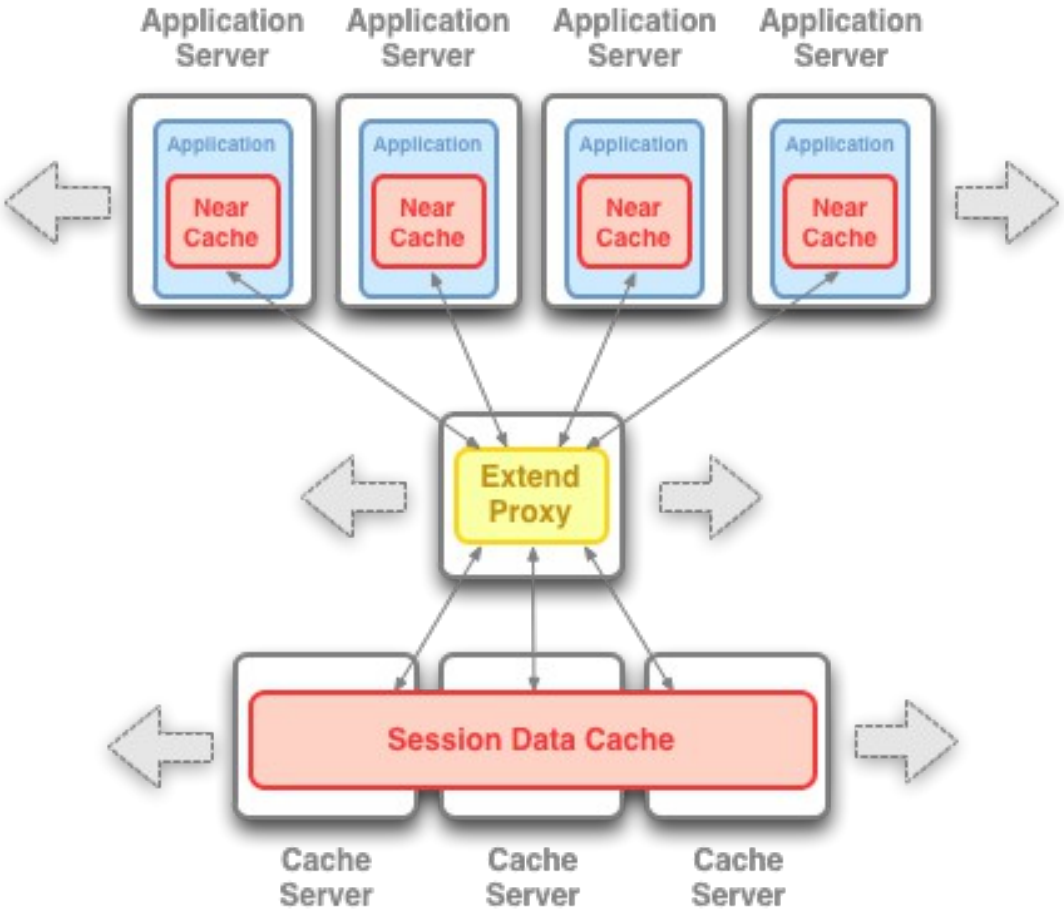
Deployment Models – In-Process



Deployment Models – Out-of-Process

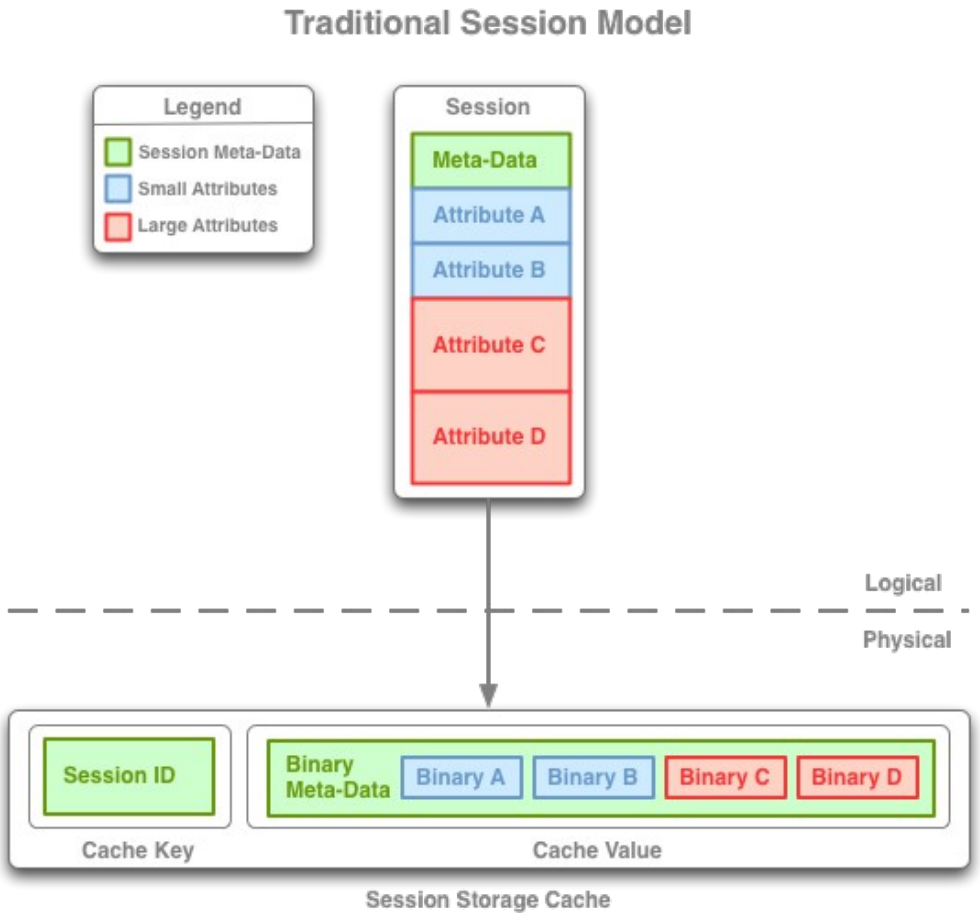


Deployment Models – Out-of-Process/Extend

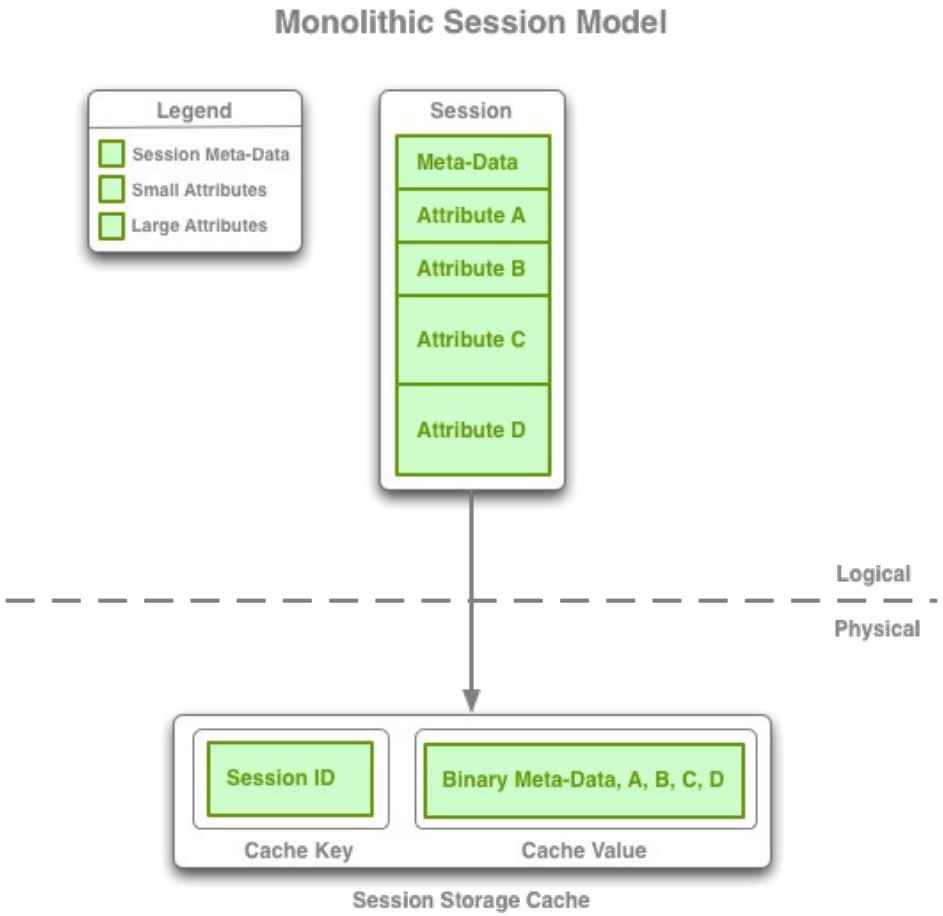


Session Models

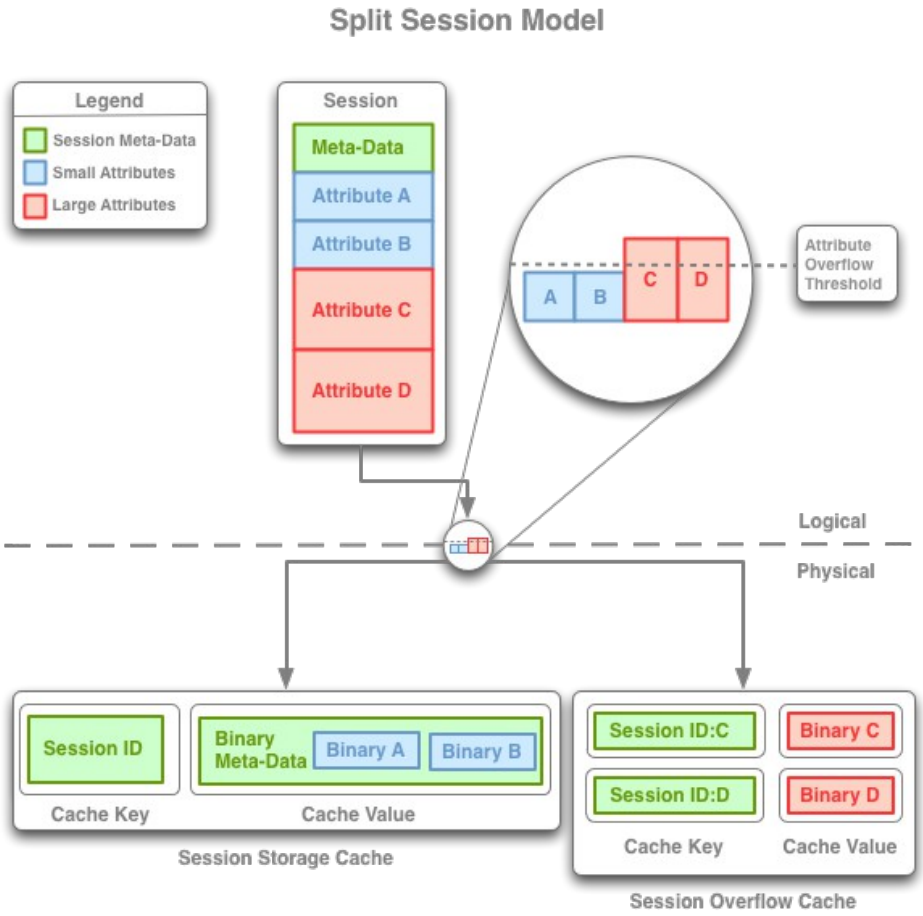
Session Models - Traditional



Session Models - Monolithic



Session Models - Split



Locking Models

Locking Models

- **Optimistc locking (default)**

  Allows multiple nodes in a cluster to access an HTTP session simultaneously; prohibits concurrent modification
- **Last Write Wins Locking**

  Like optimistc locking but the last write is allowed to win.
- **Member Locking**

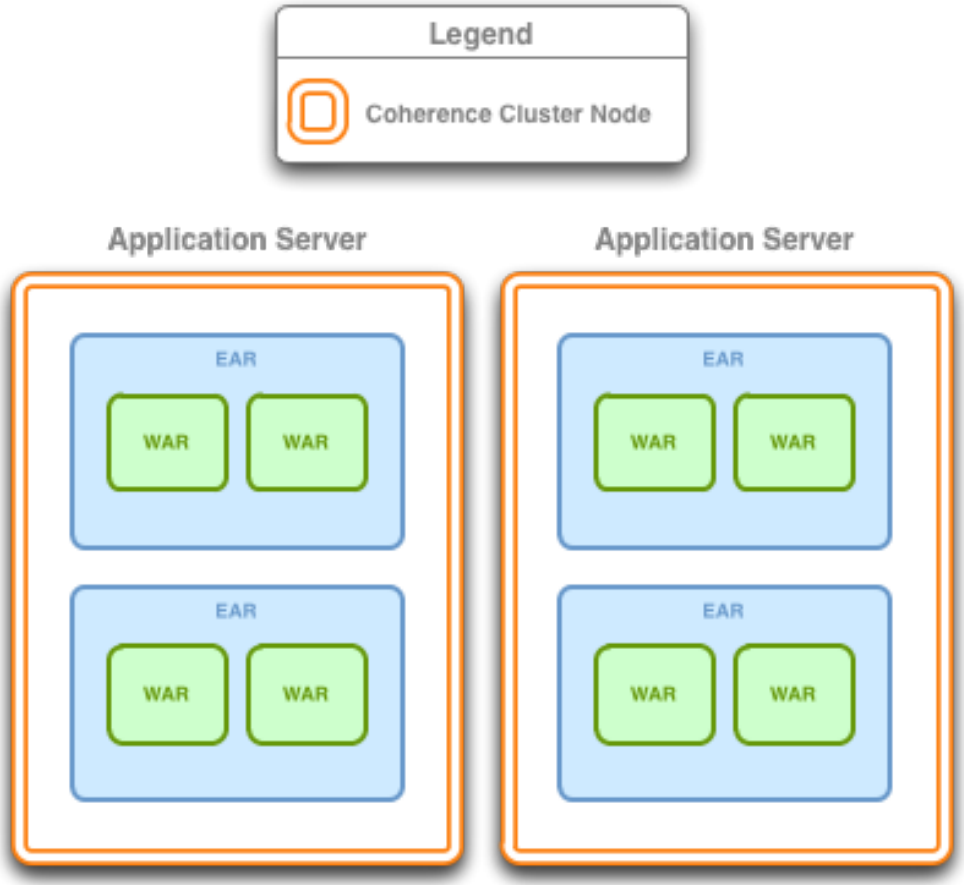
  Does not allow more than one node in the cluster to access HTTP session.
- **Application Locking**

  Multiple threads in the same web application allowed
- **Thread Locking**

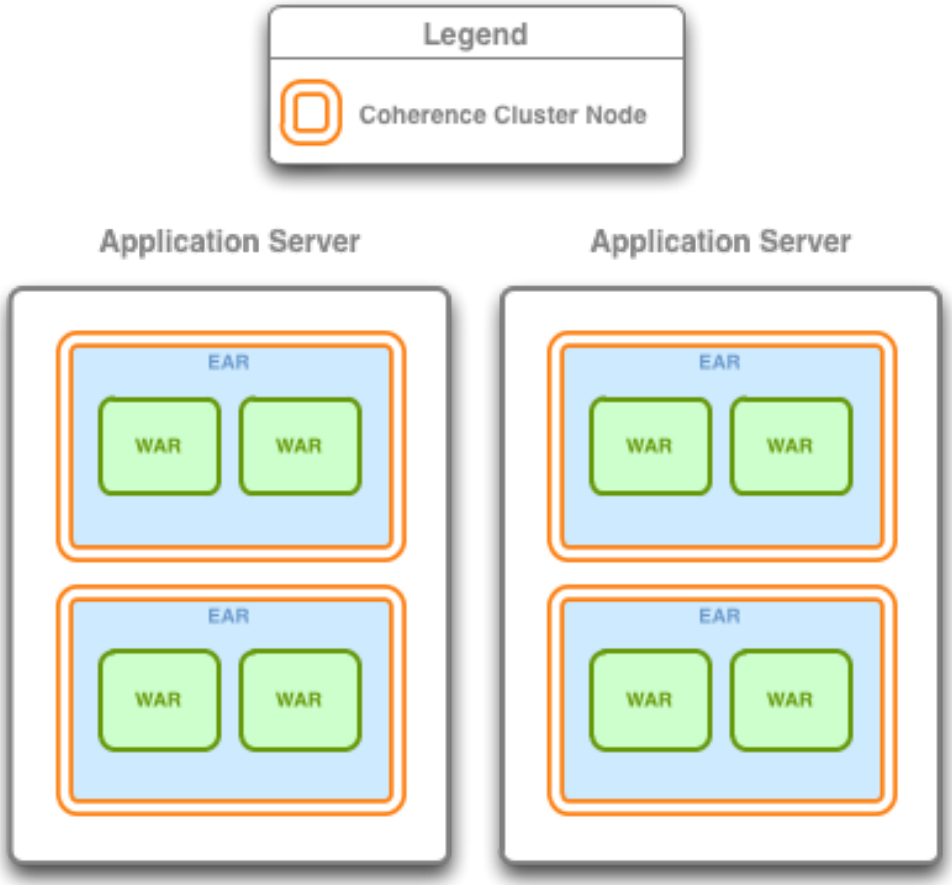
  Does not allow more than one thread in the cluster to access an HTTP session.

Cluster Node Isolation

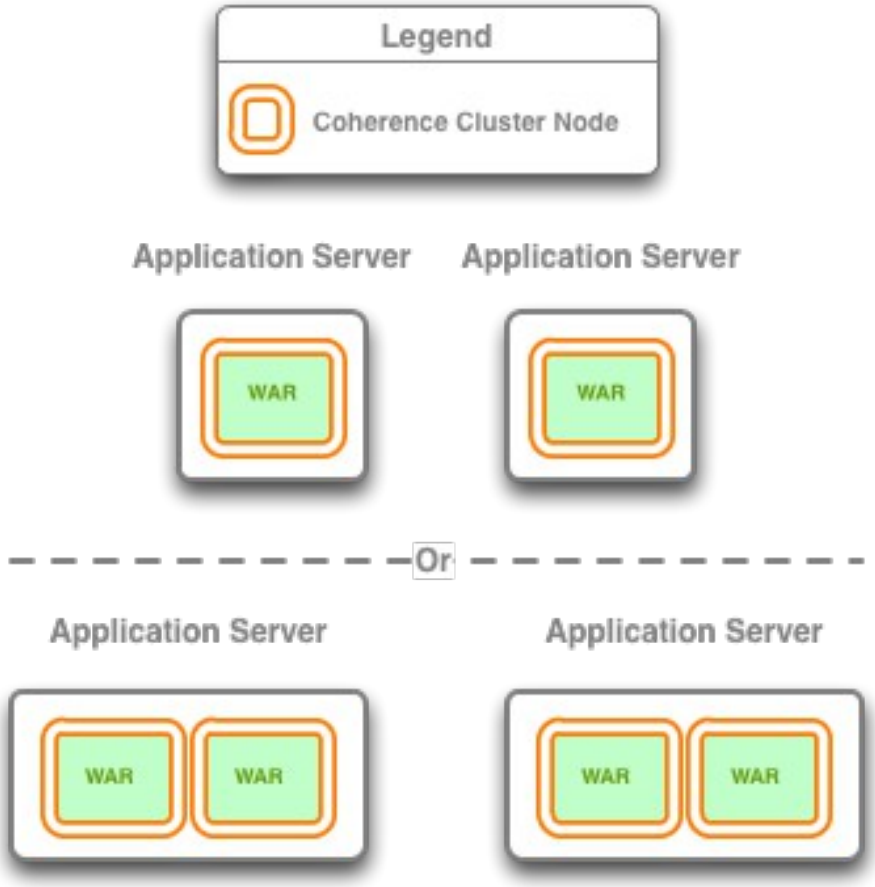
Cluster Node Isolation – App Server Scoped



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

Session Provider for the Microsoft .NET Framework

Overview:

The Coherence Session Provider for the Microsoft .NET Framework allows you to manage ASP.NET session state in a Coherence cluster in the same way Coherence*Web provides session management in JEE containers.

New Features:

- Supports all Coherence*Web Session Models (Traditional, Monolithic and Split)
- Pluggable Serializer Support
- .NET and POF Serialization Support
- Lockless exclusive session access via EntryProcessors
- Optimized Session_OnEnd Support

Demo

- sharing one session
- across three different applications
- deployed on three different app servers
  - Tomcat
  - Glassfish
  - WebLogic
- storing session data out-of-process

Prepare the WAR file

- Create the WAR file as usual
- For each container
  - inspect the WAR file

```
java -jar webInstaller.jar MyWar.war \
   -inspect -server:[CONTAINER]
```

  - review/edit generated file (coherence-web.xml)
  - modify the WAR file

```
java -jar webInstaller.jar MyWar.war
-install
```

  - deploy as usual

Different containers share the session

- http://localhost:7001/MyWar/DemoServlet
  username is currently set to null
  session.getId: 1En6Vk6qyYDz
- http://localhost:7001/MyWar/DemoServlet?username=foo
  username has been set to 'foo'
  session.getId: 1En6Vk6qyYDz
- http://localhost:8080/MyWar/DemoServlet
  username is currently set to 'foo'
  session.getId: 1En6Vk6qyYDz
- http://localhost:9090/MyWar/DemoServlet
  username is currently set to 'foo'
  session.getId: 1En6Vk6qyYDz