

Oracle Coherence In-Memory Data Grid

Arquitecturas en tiempo real

Miguel García Lorenzo

Solution Architect at Oracle

miguel.garcia.lorenzo@oracle.com

Índice

- 1 ➤ Introducción
- 2 ➤ Patrones de Arquitectura
- 3 ➤ Oracle Coherence Data Grid
- 4 ➤ Provisión y Monitorización de la Plataforma
- 5 ➤ Desarrollo e Integración Continua
- 6 ➤ Práctica de Laboratorio



Introducción

Principios Fundamentales

Introducción

In-Memory Data Grid @ IMDG

Los **IMDGs** son productos que han sido diseñados para proporcionar:

- ✓ Un gran **rendimiento** soportando miles de operaciones por segundo
- ✓ Arquitecturas de **alta disponibilidad**
- ✓ Grandes capacidades de **escalabilidad**

Este tipo de productos se engloban dentro de las **soluciones de Data Management** por lo que en fase de diseño de la solución debemos valorar tanto las **características ACID** y como el teorema de **CAP** en función de las necesidades de la solución.

Introducción

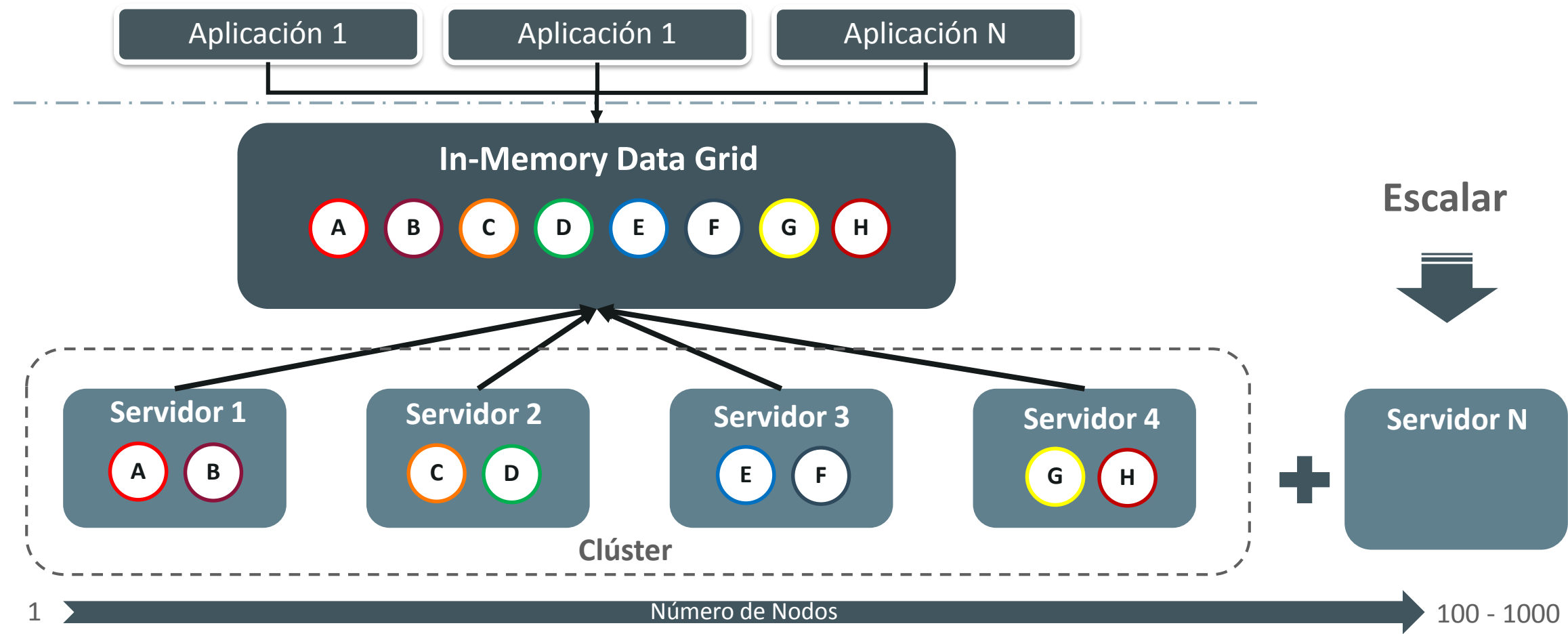
In-Memory Data Grid @ Características

Las principales características técnicas que define a los IMDG:

- **Arquitectura de datos distribuida** a través de los servidores que componen el IMDG
- Los datos se encuentran en la memoria **RAM** de los servidores
- Habitualmente implementan una **estructura de datos** del tipo <clave,valor>
- Implementan modelos **MapReduce**
- Permiten la **persistencia** en plataformas clásicas y **no volátiles**

Introducción

In-Memory Data Grid @ Diagrama Arquitectura Alto Nivel



Introducción

Teorema CAP – Genera Debate

Consistency: En un mismo momento del tiempo todas las lecturas obtienen el mismo dato, el más reciente.

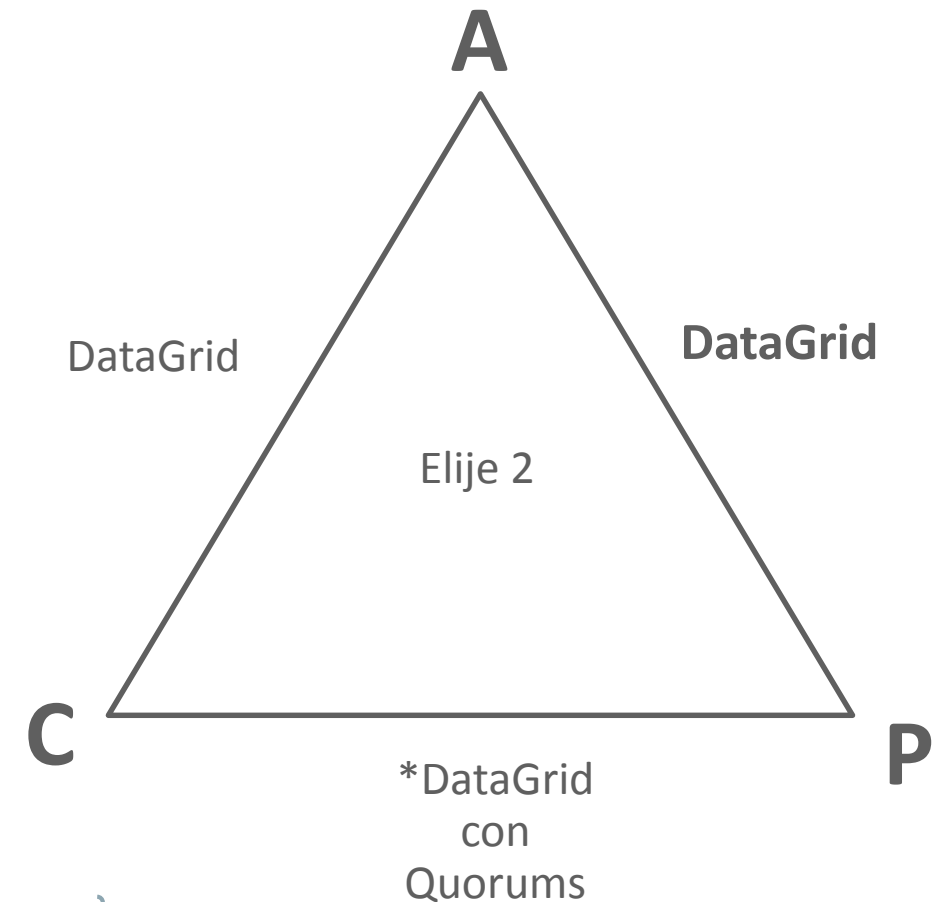
- Garantiza que no se obtiene información desactualizada.

Availability: Los clientes pueden realizar operaciones aunque se hayan caído alguno de los nodos.

- La respuesta a las peticiones deben realizarse en un tiempo razonable.

Partition-tolerance: El sistema deben seguir funcionando aunque las particiones, conjunto de nodos, no puedan comunicarse.

- Ambas particiones pueden seguir trabajando { ★ Split-Brain }

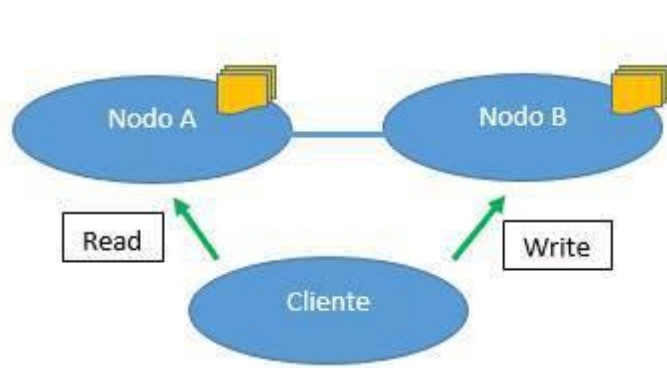


Introducción

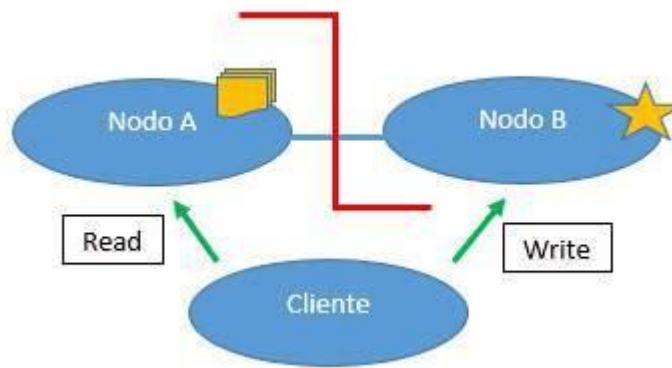
Teorema CAP - Escenarios

Un cliente escribe datos en el nodo A y los lee desde el nodo B.

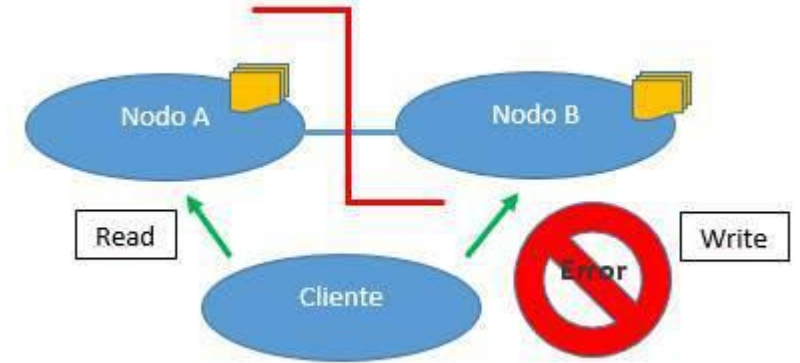
- Los datos se replican desde el nodo A al nodo B.
- El nodo B los recupera desde el nodo A.



- Cuando la comunicación es estable
- El sistema se comportará accesible y consistente, no tiene la necesidad de ser tolerante a particiones.



- Si existe una falla en la red entonces el nodo B no puede recuperar la información del nodo A
- El nodo B devuelve datos desactualizados.
- En este caso **el sistema no es consistente**.



- Si existe una falla en la red entonces el nodo A trata de conectarse al nodo B sin éxito,
- Esto ocasiona que ocurra un error de timeout o que no se responda la petición en un monto de tiempo razonable.
- En este caso **el sistema no es accesible**.

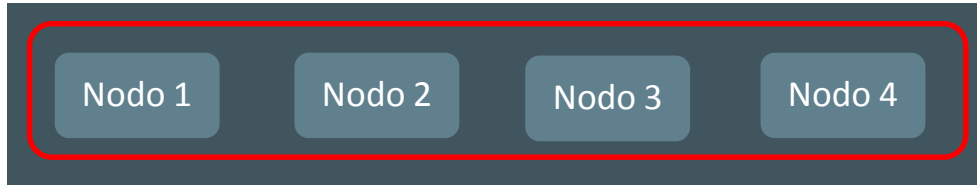
*Referencia: <https://code2read.com/2015/05/12/sistemas-distribuidos-teorema-cap/>

Introducción

Teorema CAP – Preguntas y Respuestas

AP

Data Clúster Grid



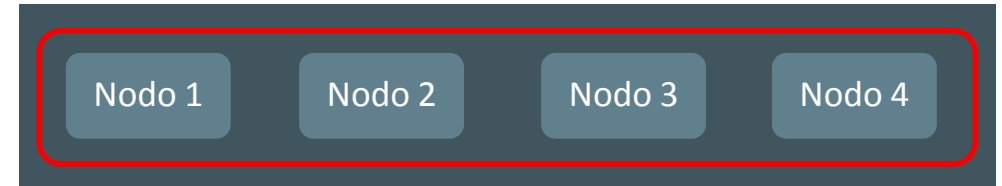
✗



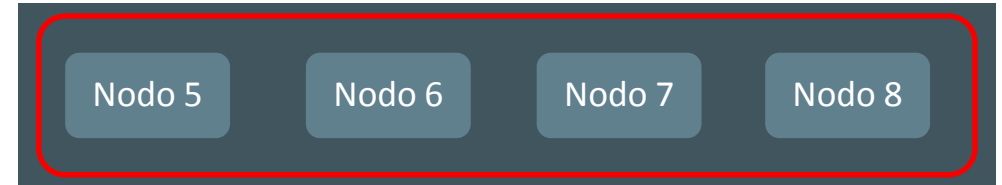
Seguimos dando servicio pero sin consistencia

CP




Quorum = 8



✗



Consistencias pero no hay servicio

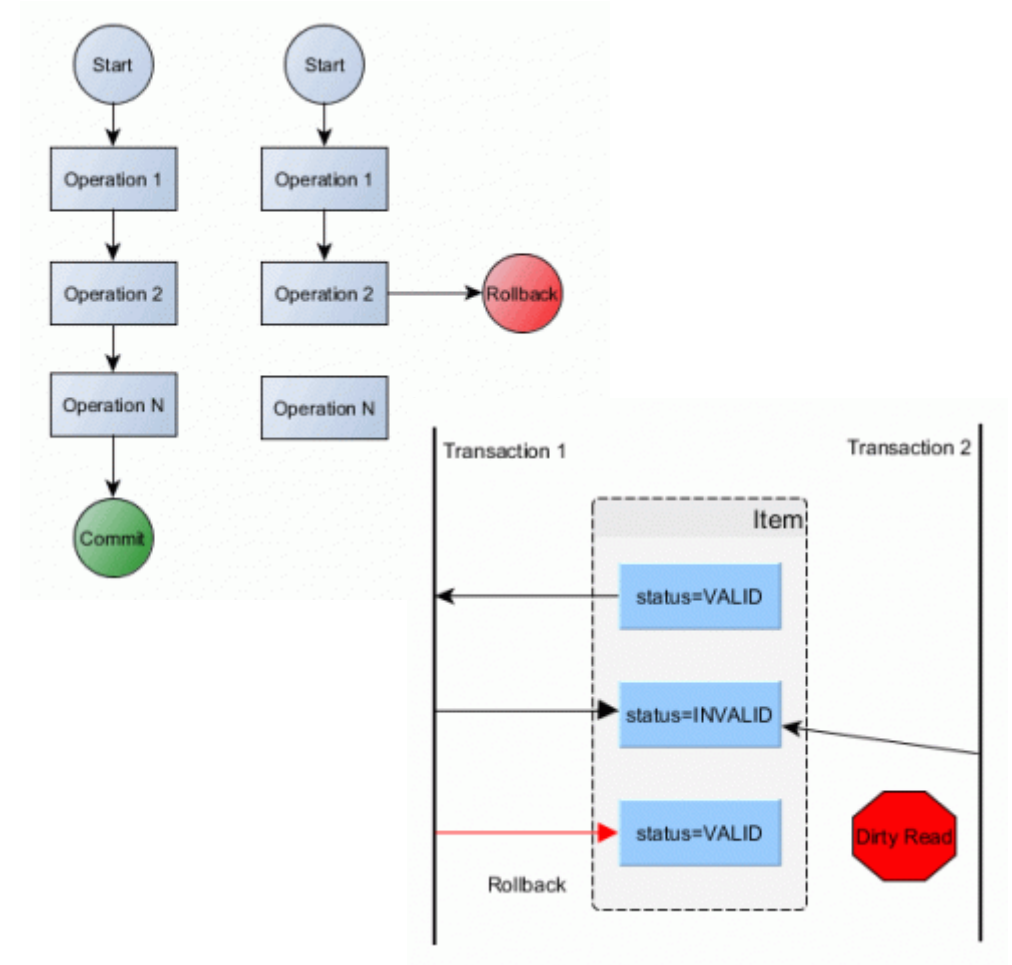
-  Clúster Data Grid Inicial
-  Clúster Data Grid generados por Split Brain
-  Problema de red

* Quorum: Número mínimos nodos en el clúster para dar servicio

Introducción

ACID – Características de las Transacciones

- **Atomicity:** Una transacción completa totalmente o no se realiza.
- **Consistency o Integridad:** Los datos están en un estado consistente cuando la transacción comienza y finaliza.
- **Isolation:** El aislamiento proporciona transacciones independientes y oculta los cambios intermedios para evitar la corrupción de datos.
 - En un momento dado del tiempo y para un conjunto de datos únicamente se ejecuta una transacción de forma concurrente.
- **Durability:** El cambio realizado por una transacción finalizada correctamente será persistido y no podrá perderse por un fallo del sistema.





Patrones de Arquitectura

Descripción de patrones de arquitecturas y casos de uso donde se puede utilizar productos In-Memory Data Grid

Introducción

Arquitectura @ Patrones y Casos de Uso



Arquitecturas Real-Time & Near Real-Time (Fast Data)



Arquitecturas Lambda



Arquitectura Command Query Responsibility Segregation



Otros Patrones: Consolidación, Web, etc..

Arquitectura Real-Time & Near Real-Time

Casos de Uso @ Ejemplos de soluciones implantadas

Near Real-Time

- No es necesario disparar una acción en el mismo momento en el que se recibe un evento o se detecta un patrón
- Recepción de ventas
- Cálculo de stock

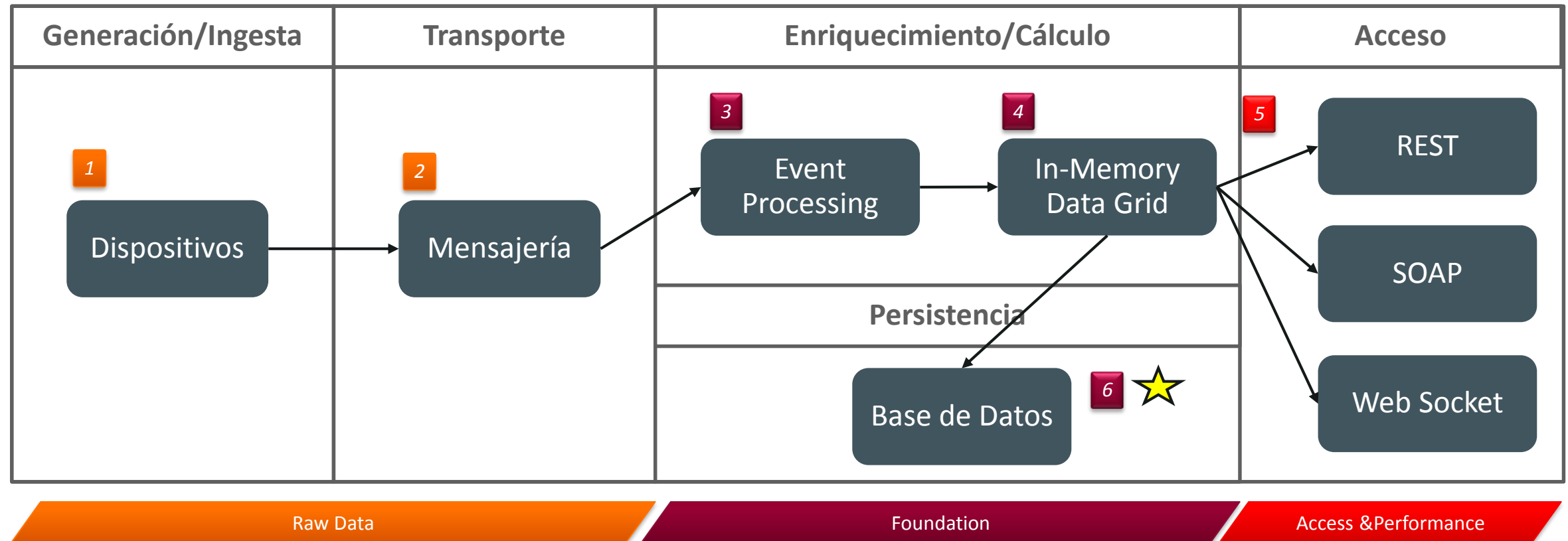
Real-Time

- Es necesario disparar una acción de forma inmediata a recibir un determinado evento o detectar un patrón
- Recepción de alertas
- Detección de fraudes
- Redes sociales o casas de apuesta

★ En esta presentación no abordamos arquitecturas de tiempo real clásicas del tipo RTOS

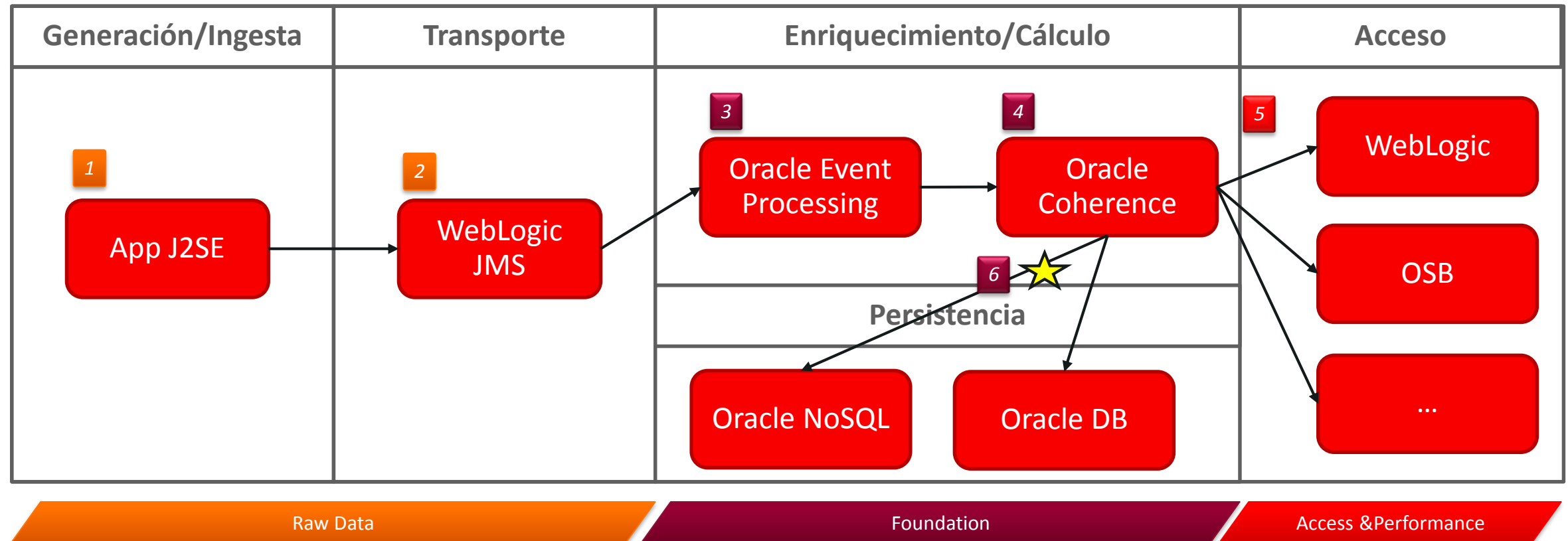
Arquitectura Real-Time & Near Real-Time

Diagrama de N1 @ Fast Data



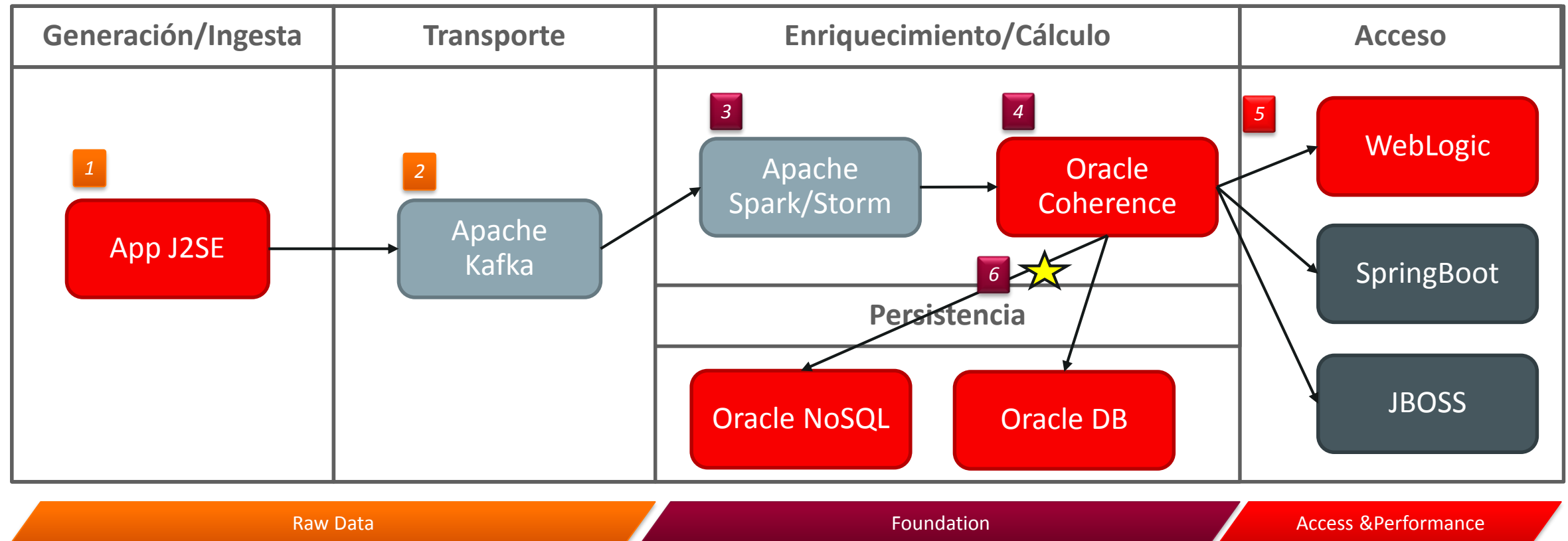
Arquitectura Real-Time & Near Real-Time

Diagrama de N2 @ Fast Data – Portafolio Oracle



Arquitectura Real-Time & Near Real-Time

Diagrama de N2 @ Fast Data



Arquitectura Lambda

Casos de Uso @ Ejemplos de soluciones implantadas

Este patrón de arquitectura se utiliza tanto para soluciones de negocio como para soluciones de pura analítica IT

Capa Batch

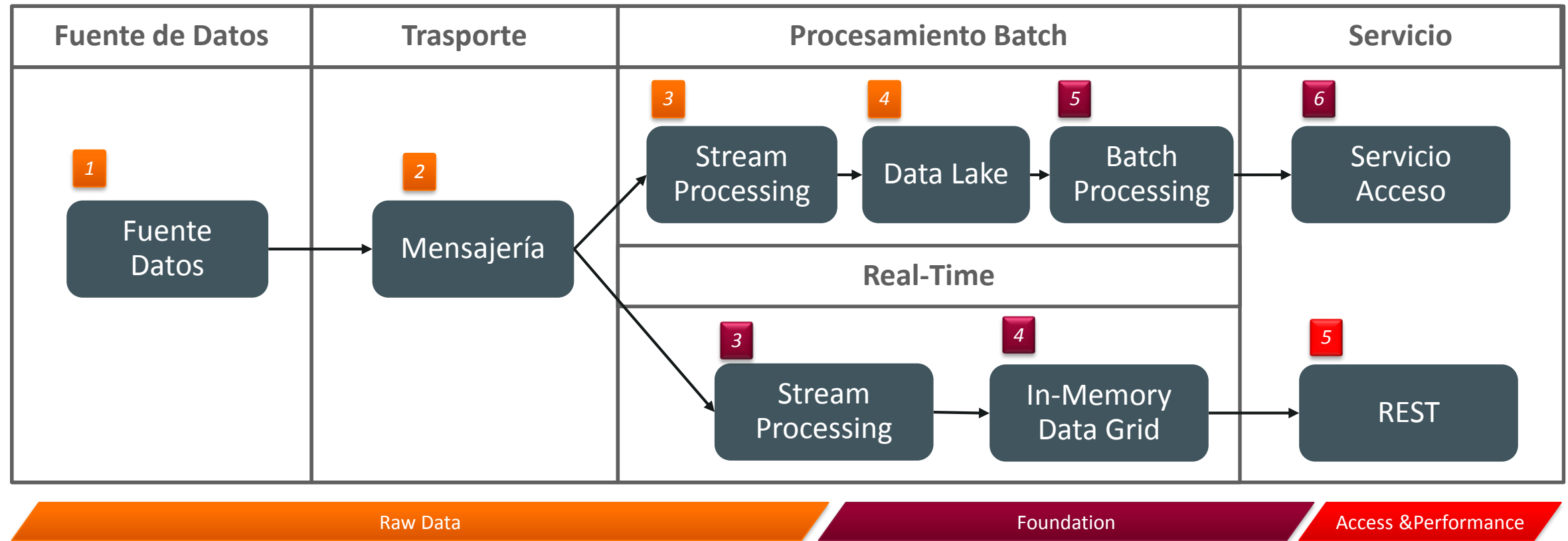
- Clásica solución Big Data
- Se almacena dato “RAW” y en ocasiones “Foundation”
- Se utiliza para para
 - Análisis de históricos
 - Búsqueda de nuevos patrones
 - Verificación resultados real-time
 - Informes no cubiertos por el real-time

Capa Real-Time

- En soluciones de monitorización IT se puede utilizar para:
 - Implementar patrones predictivos
 - Detección de incidencias

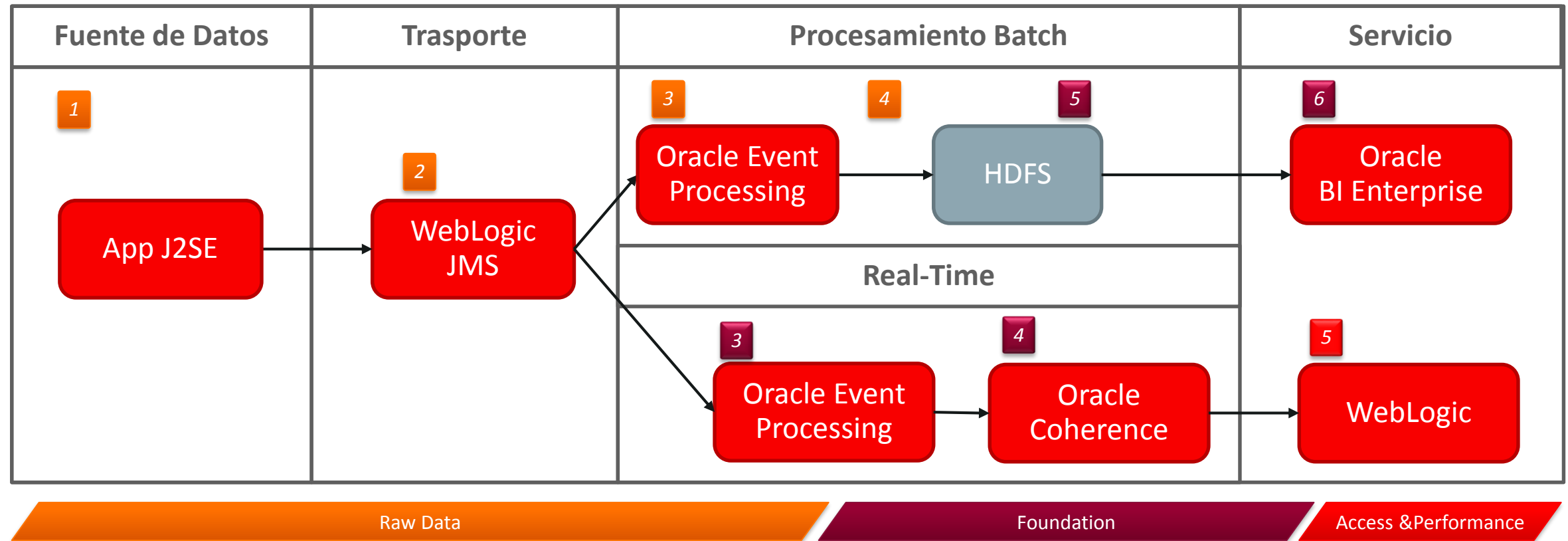
Arquitectura Lambda

Diagrama de N1 @ Arquitectura Asíncrona



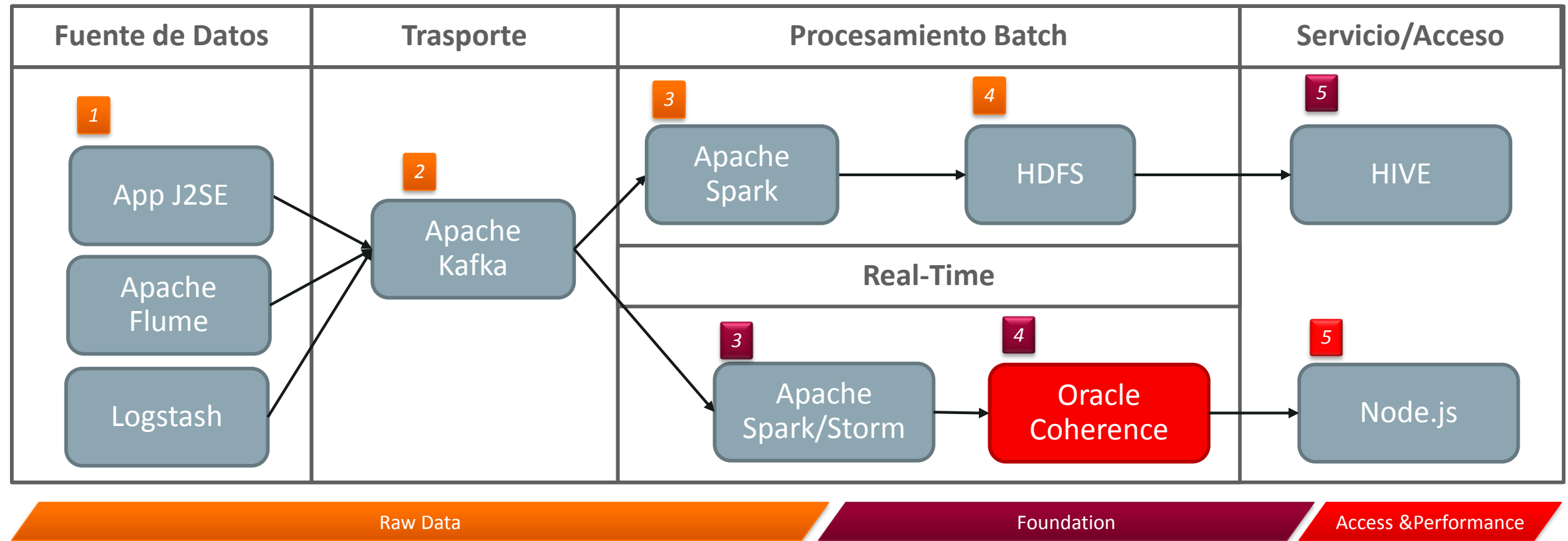
Arquitectura Lambda

Diagrama de N2 @ Arquitectura Asíncrona – Portafolio Oracle



Arquitectura Lambda

Diagrama de N2 @ Arquitectura Asíncrona



Arquitectura Command Query Responsibility Segregation

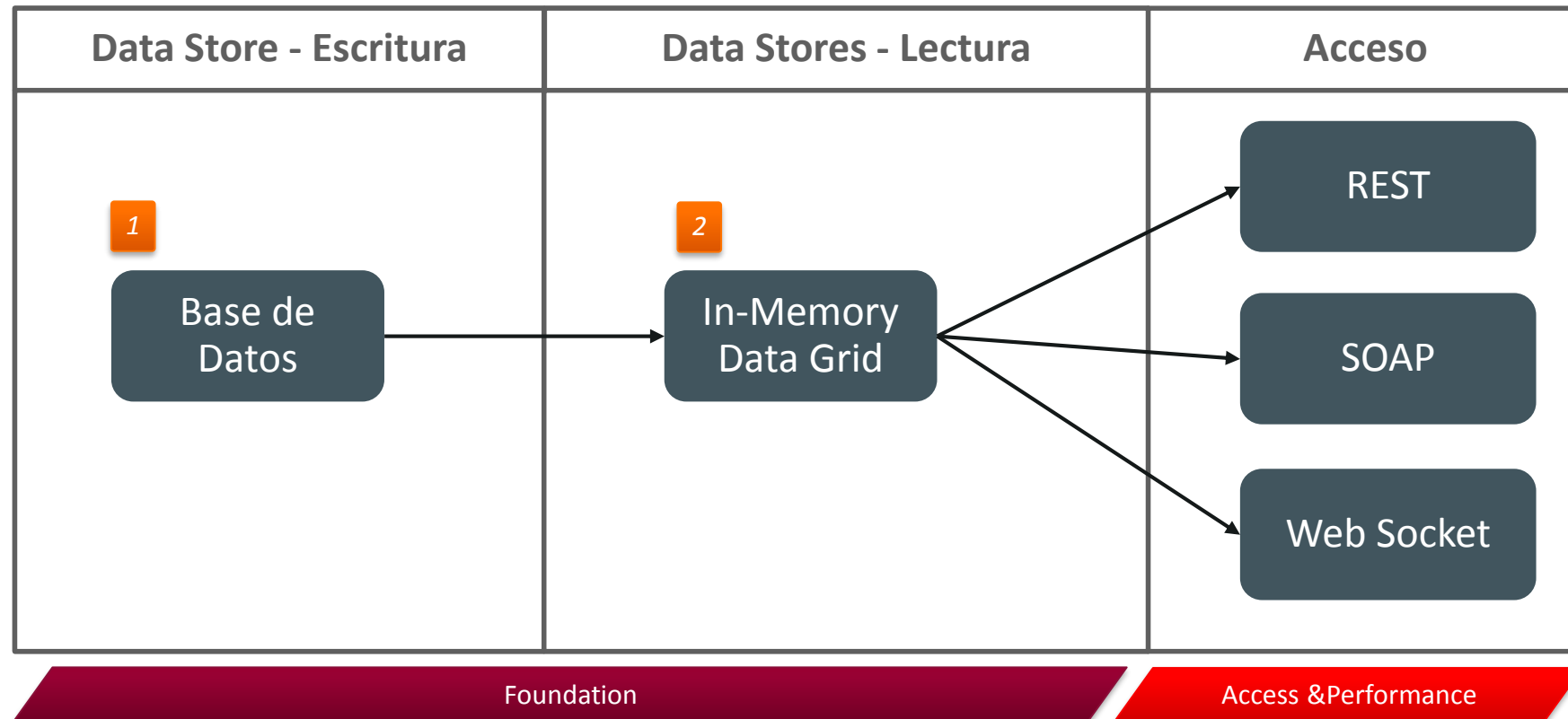
Casos de Uso @ Ejemplos de soluciones implantadas

Este patrón implementa una segregación de las operaciones de escritura y lectura, permitiendo proporcionar diferentes modelos de datos lógicos de lectura optimizados para cada caso de uso:

- Consolidación de información de STOCK para eCommerce
- Consolidación de información de ventas para BI
- Diferentes agregados de información

Arquitectura Lambda

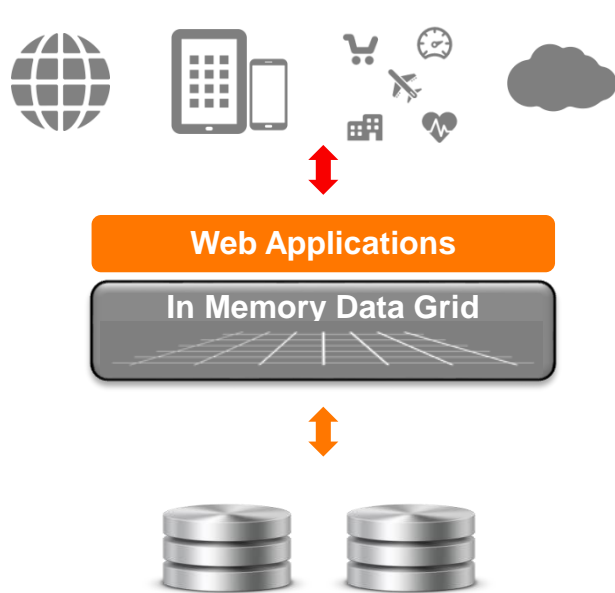
Diagrama de N1 @ Arquitectura Asíncrona



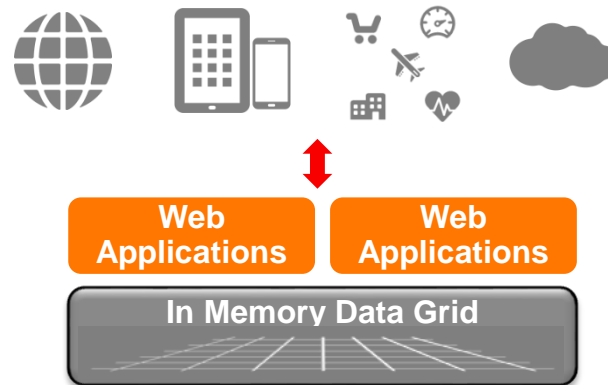
Otros Patrones

Casos de Uso @ Otros Ejemplos de soluciones implantadas

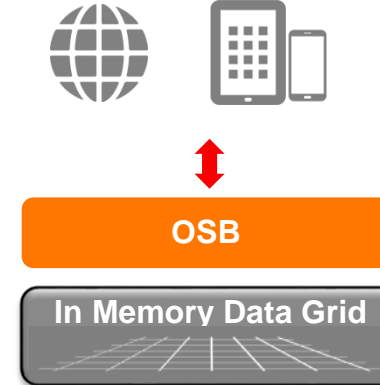
Escalar Plataformas Web (High Read Only; Medium Write)



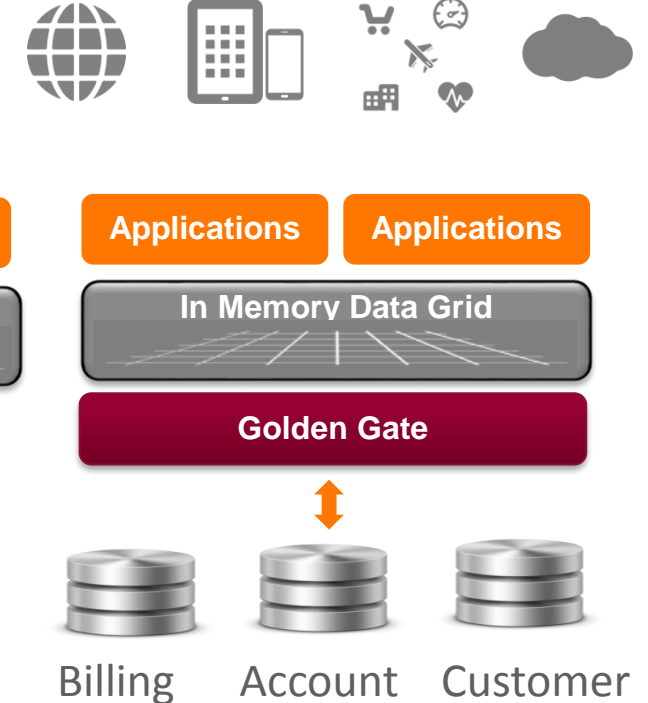
Compartir Sesión Web



Cachés Técnica Servicios



Consolidación



A woman with long brown hair and glasses is sitting at a wooden table in a cafe. She is wearing a brown leather jacket over a blue patterned scarf. She is holding a black smartphone to her ear with her left hand and looking down at an open book or magazine on the table with her right hand. The background is a blurred interior of a cafe with other tables and chairs.

Oracle Coherence Data Grid

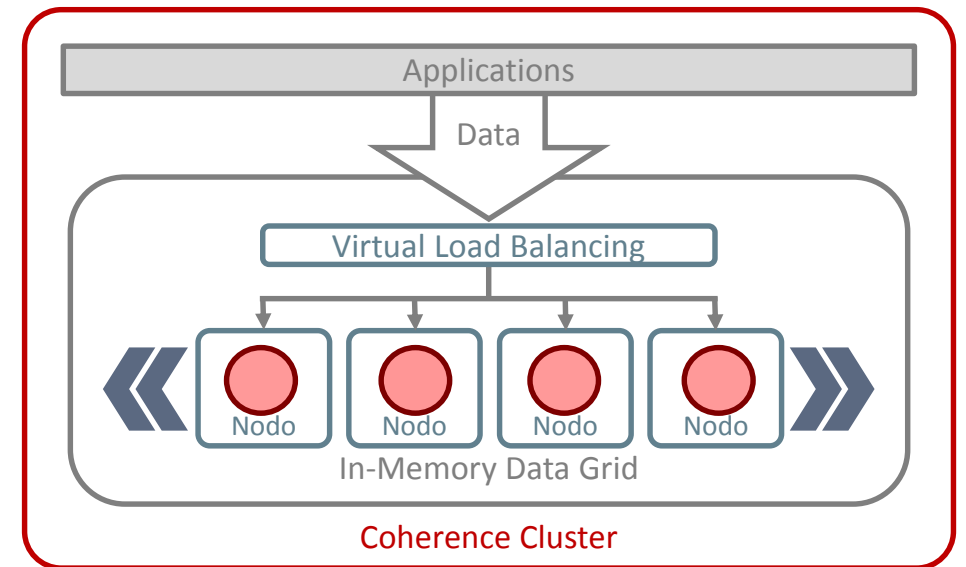
Descripción técnica del producto Coherence 12cR2

Introducción

In-Memory Data Grid @ Oracle Coherence

Coherence es el producto de **Oracle** para soluciones In-Memory Data Grid:

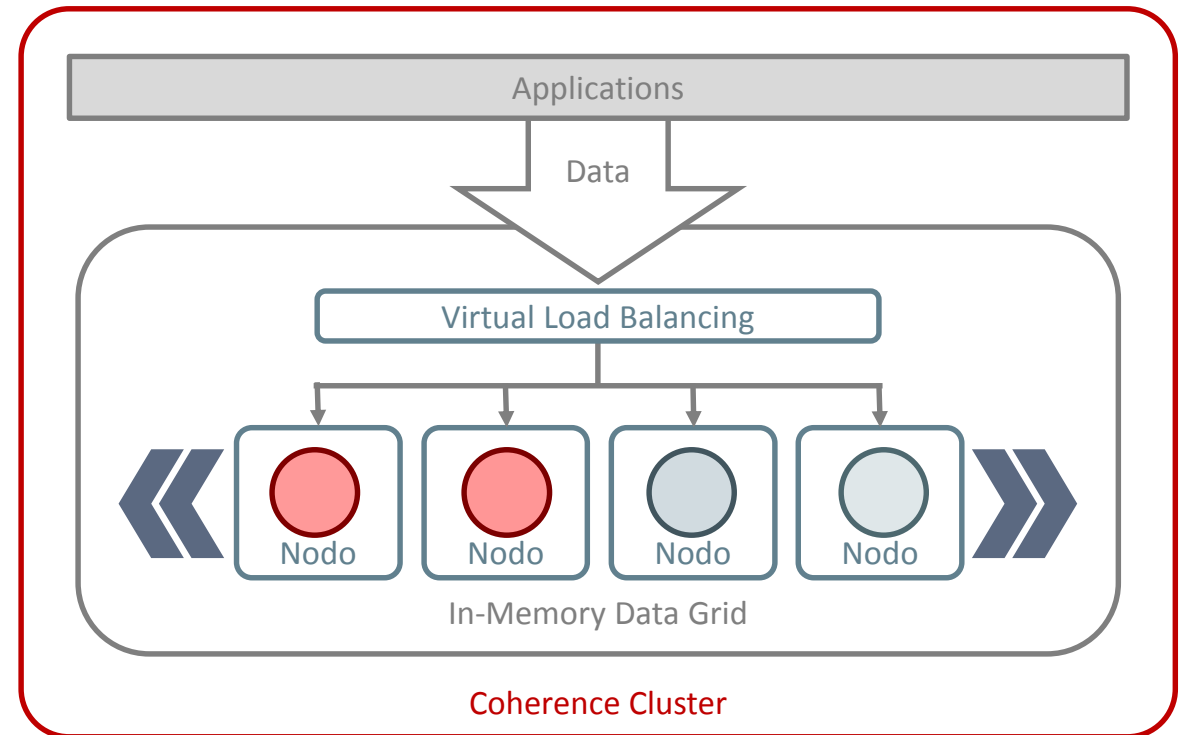
- Single System Image
- Escalable tanto e nivel de lecturas como escrituras
- Escalabilidad linear en procesamiento y capacidad
- No Single-Points-Failure (SPOFs)
- Failover y failback automáticos, transparentes y rápidos



Características

In-Memory Data Grid @ **Oracle** Coherence

- Clustered In-Memory **Key-Value Store**
- **Dynamically Scalable**
- Java, .NET, C++, REST, Memcached, Jcache
- **Async Programming Models**
- In-place Distributed Processing
- Queries & Continuous Queries
- **MapReduce Aggregation**
- **Rich Live Event Programming model**
- Data source Integration
- App Server Integration



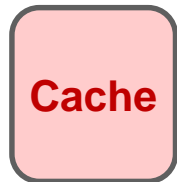
Conceptos Clave

In-Memory Data Grid @ **Oracle** Coherence



Nodo

- Conjunto de nodos/instancias de Coherence que conforman un clúster de Coherence.
- Las instancias son procesos Java.
- Existen dos tipos de nodos: Storage-enabled (Almacena datos) y Storage-disabled (No almacena datos)



Cache

- Estructura de datos <clave, valor> que se despliega sobre un clúster de Coherence



POF

- Portable Object Format es un tipo de serialización utilizado por Coherence para optimizar el rendimiento

Conceptos Clave

In-Memory Data Grid @ **Oracle** Coherence

TCMP



- Tangosol Cluster Management Protocol (IP-Based) de comunicación utilizado para tareas de “Server Discovery”, provisión de servicios, transferencia de datos, descubrimiento de servicios y gestión del clúster
- Protocolo asíncrono orientado a rendimiento y no generar bloqueos

Coherence
Extend

- Tipo de integración nativa de Coherence a través de “Proxy Services” y clientes específicos para Java, C++ y .Net

GAR

- Grid ARchive es el tipo de empaquetado para la aplicaciones Coherence
- Las cachés se definen en

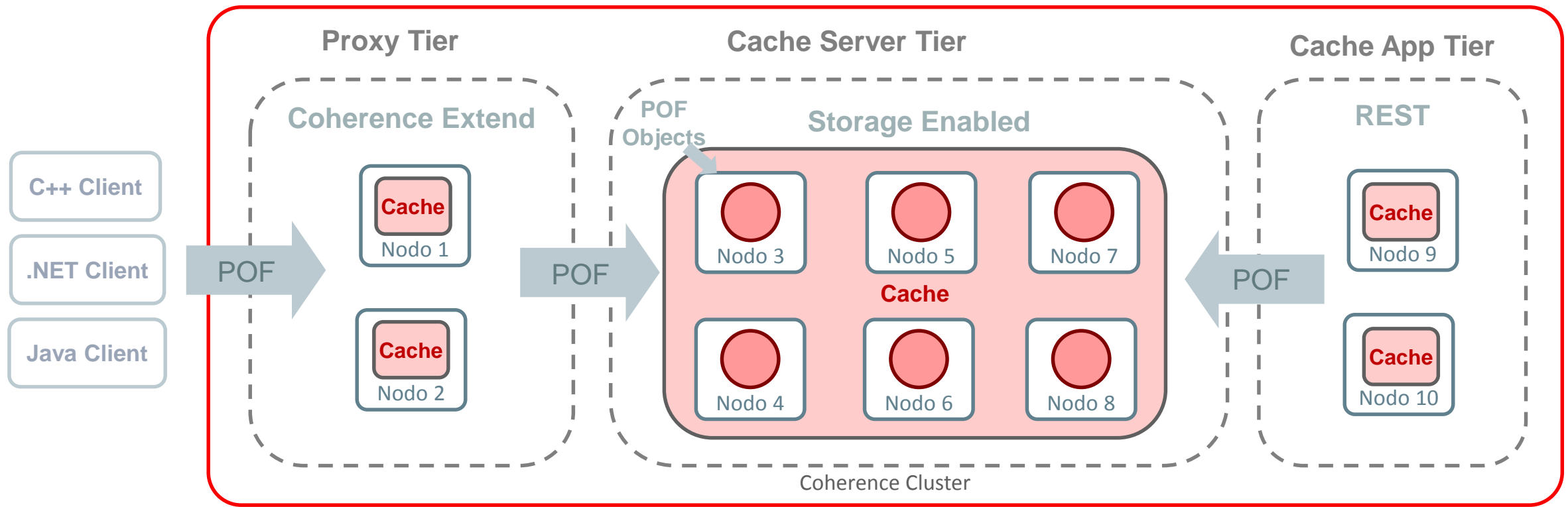
Coherence
REST

- Servicios de Datos proporcionados por el cliente out-of-the-box

Topología

In-Memory Data Grid @ **Oracle** Coherence

Topología de referencia para arquitecturas Fast Data



Tipos de Caches

In-Memory Data Grid @ **Oracle** Coherence

Local

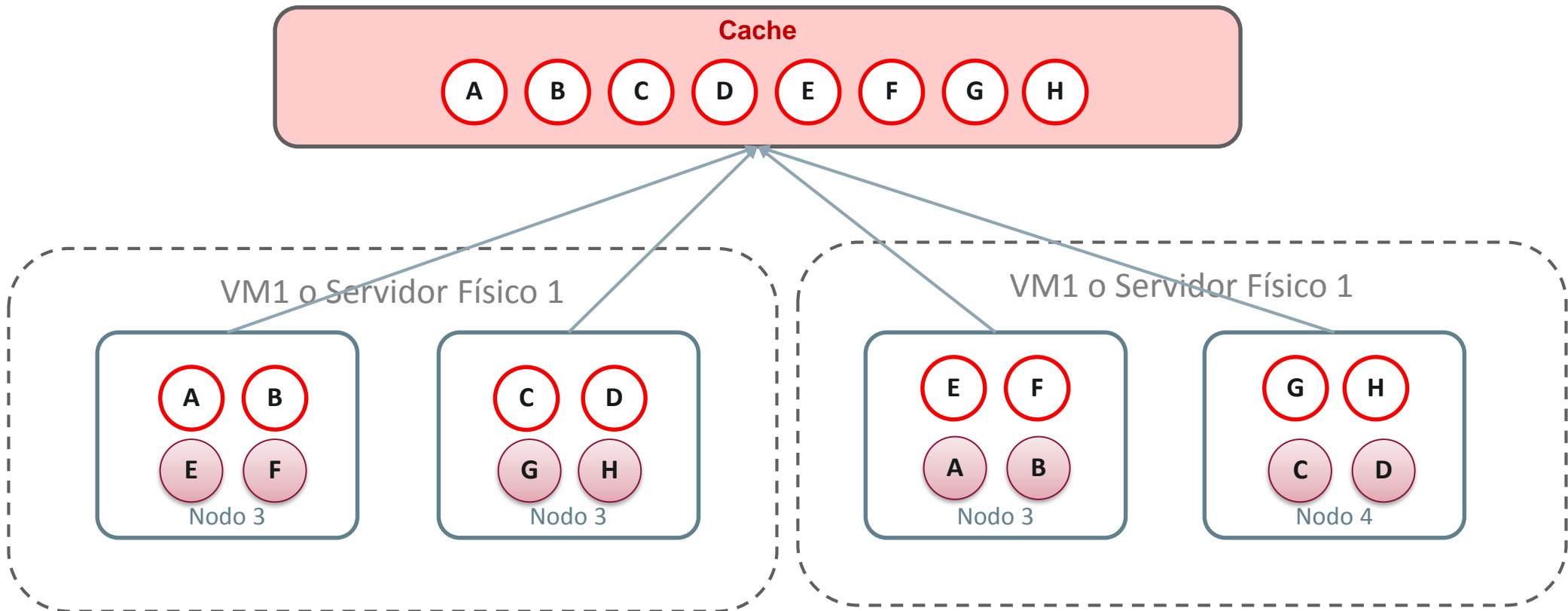
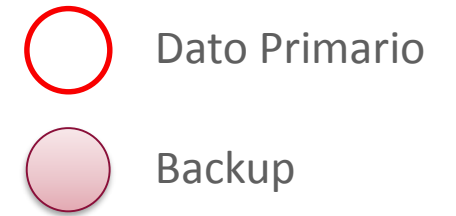
Replicada

Distribuida

Near

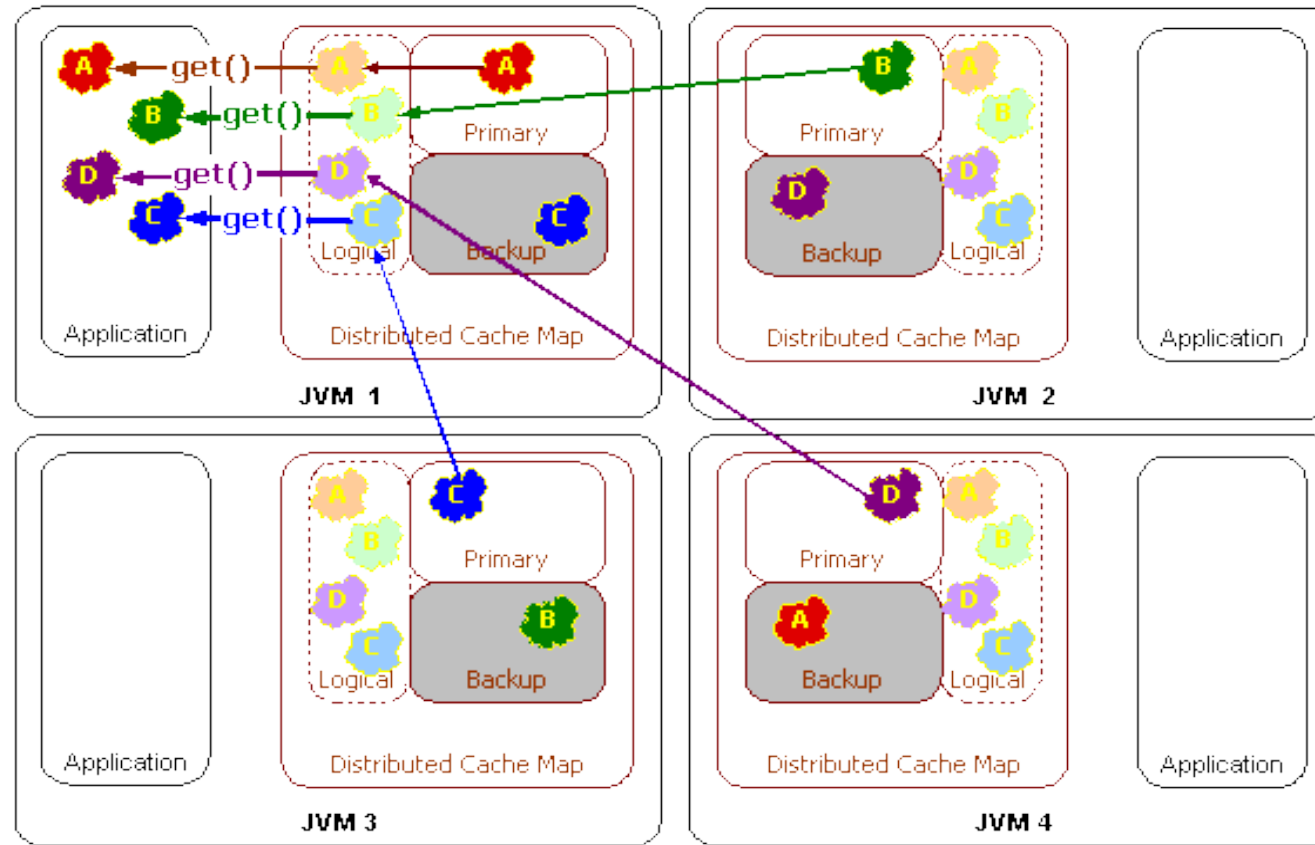
Tipos de Caches - Distribuida

In-Memory Data Grid @ **Oracle** Coherence



Tipos de Caches - Distribuida

In-Memory Data Grid @ **Oracle** Coherence

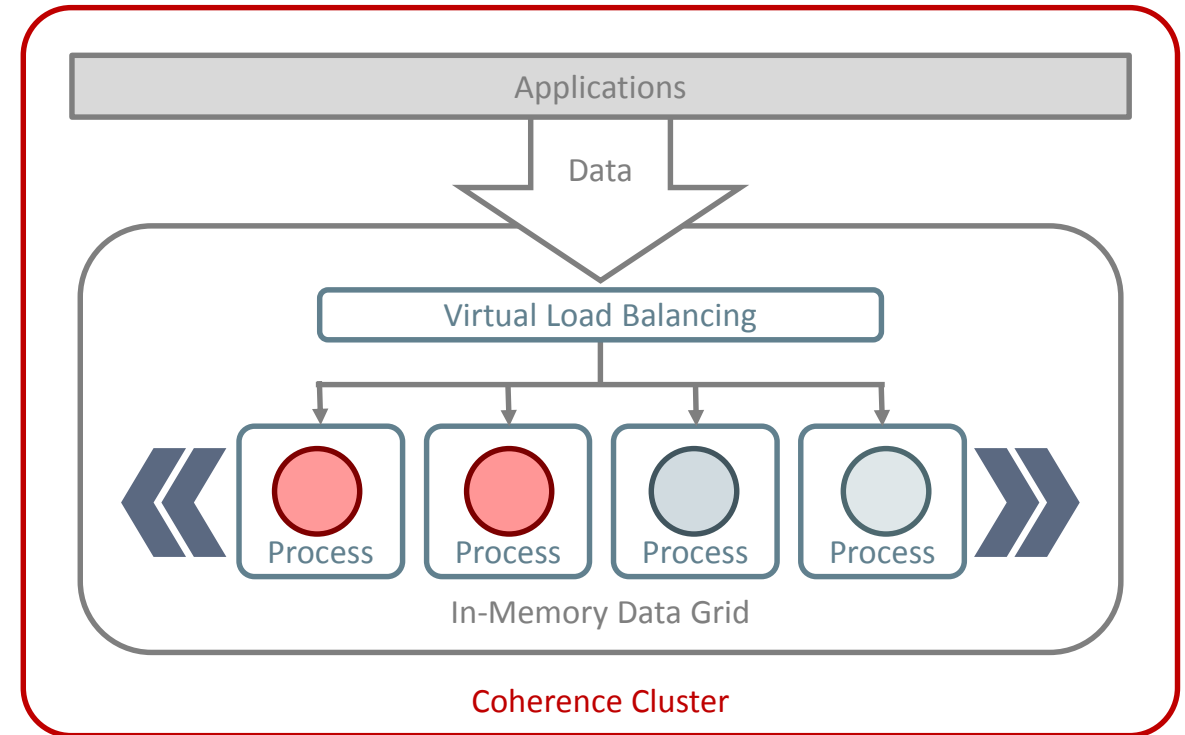


Tipos de Caches - Distribuida

In-Memory Data Grid @ **Oracle** Coherence

Unlimited Data and Processing Capacity:

- Data load balanced across data grid.
- Data/processing scales linearly.
- Ownership responsibilities partitioned.
- Access/update latency are constant.
- Best for large sets of frequently updated data.

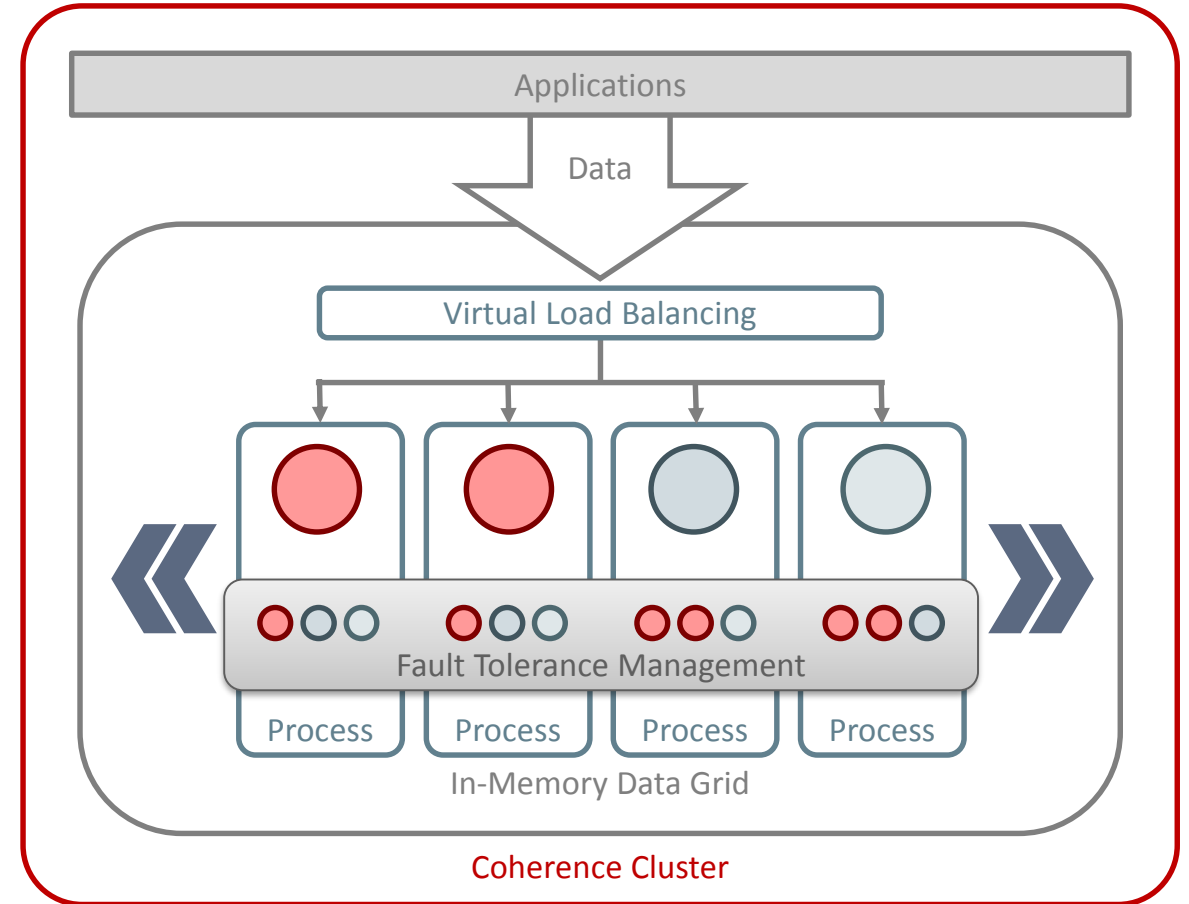


Tipos de Caches - Distribuida

In-Memory Data Grid @ **Oracle** Coherence

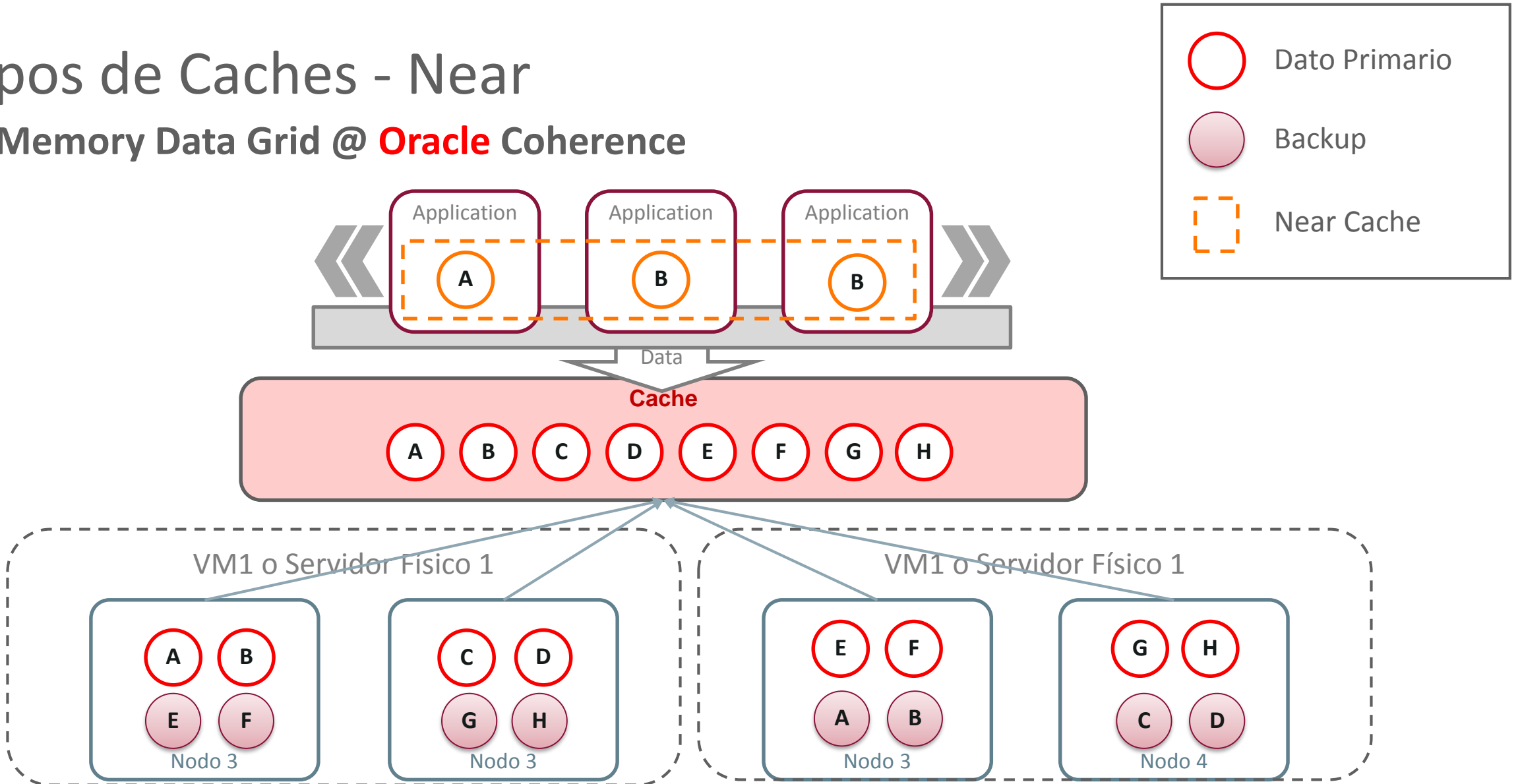
Automatic fault tolerance management:

- Backups stored on separate machine.
- Even distribution of backup responsibilities.
- Configurable number of backup copies.
- Once-and-only-once processing guarantees.



Tipos de Caches - Near

In-Memory Data Grid @ **Oracle** Coherence

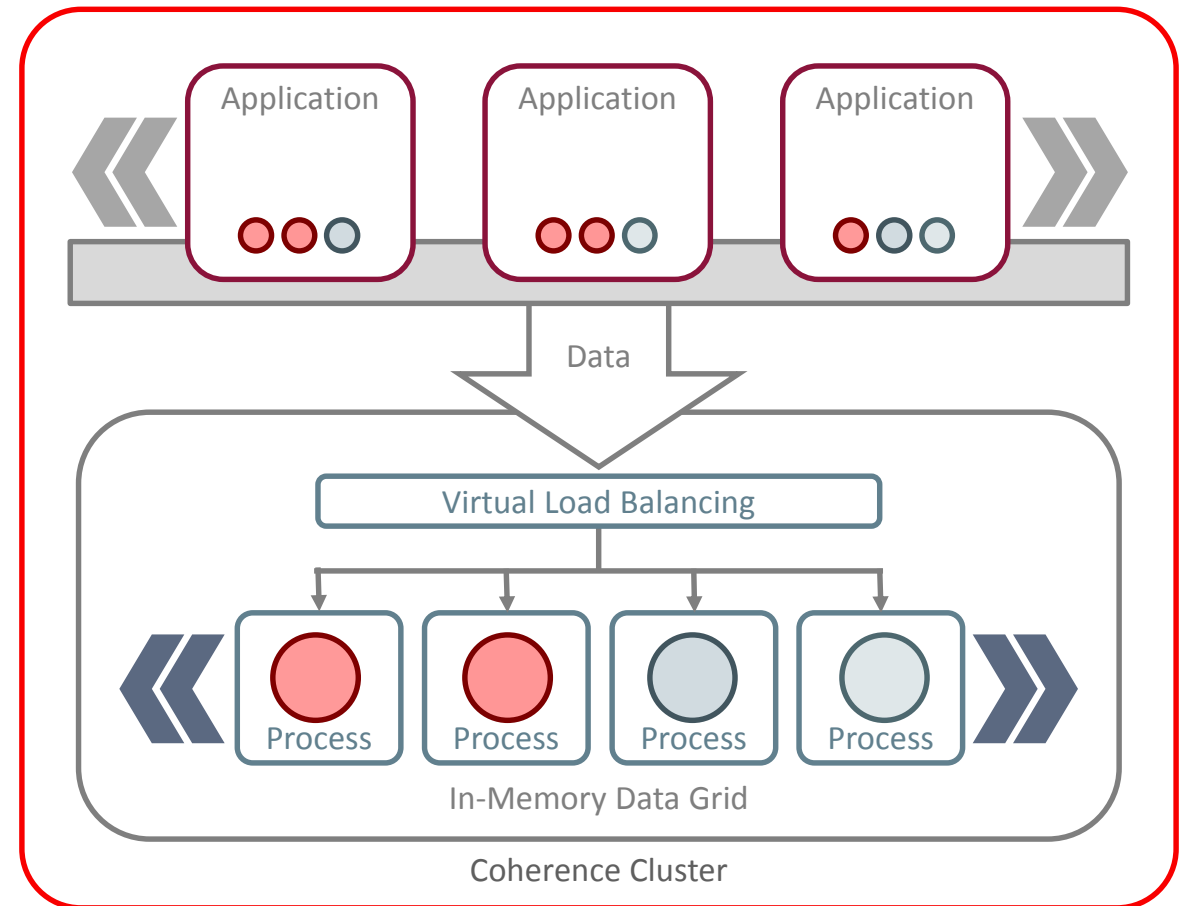


Tipos de Caches - Near

In-Memory Data Grid @ **Oracle** Coherence

Rapid Data Access From Clients

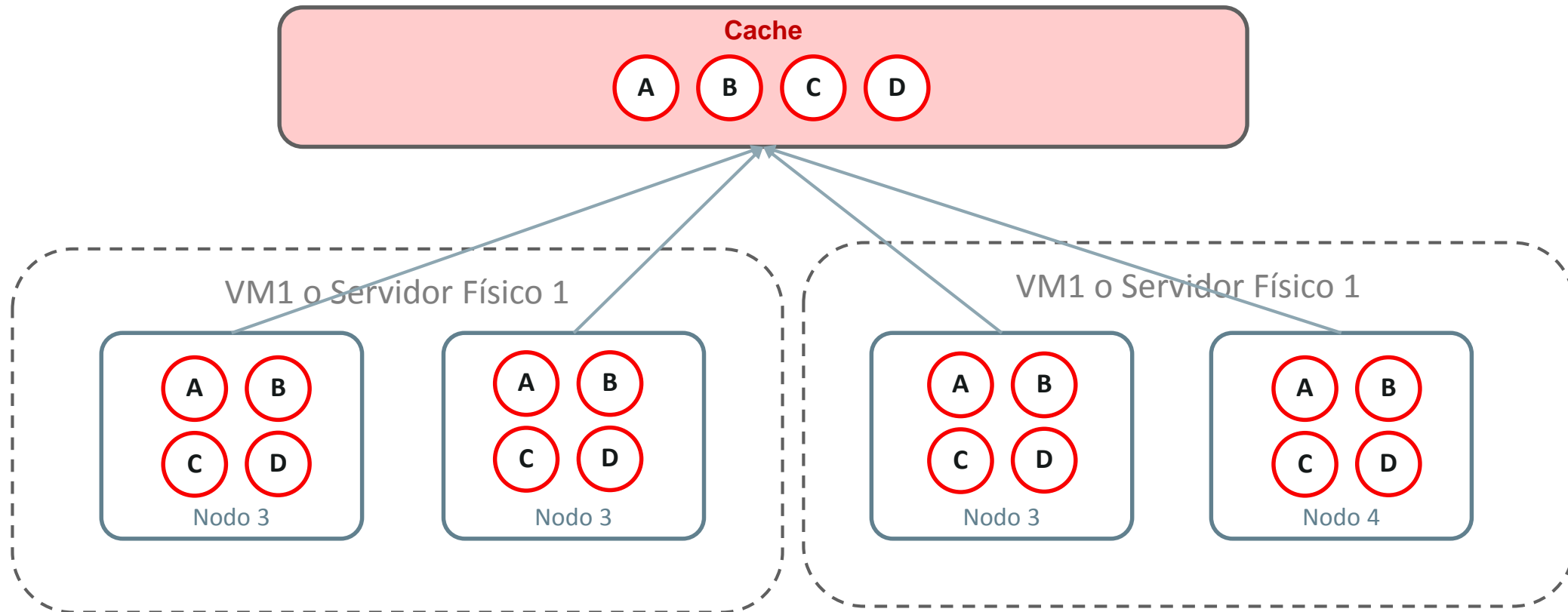
- Recently used data is stored locally.
- Repeated access local and immediate.
- Automatically populated upon data access.
- Automatic invalidation of updated data.
- Scale tiers independently.



Tipos de Caches - Replicada

In-Memory Data Grid @ **Oracle** Coherence

 Dato Primario

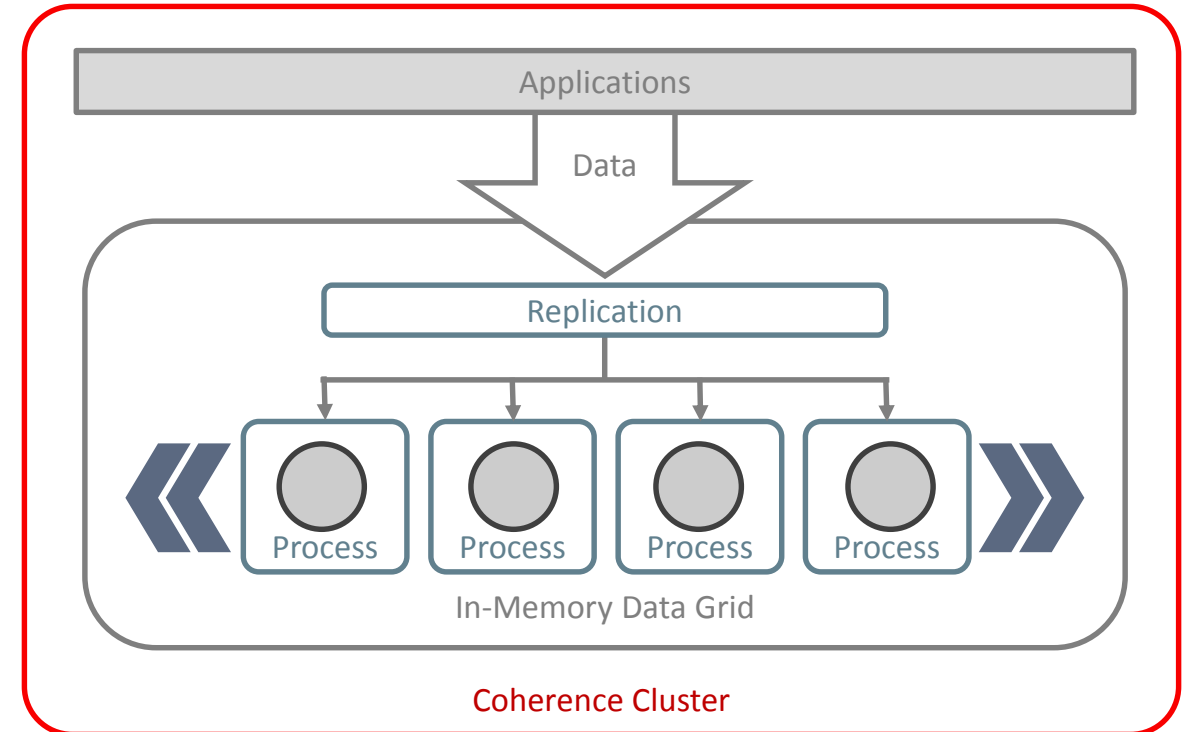


Tipos de Caches - Replicada

In-Memory Data Grid @ **Oracle** Coherence

Rapid Access to Reference Data

- Entire data set is replicated.
- Data is stored as native Java object
- Data access is immediate.
- Updates are replicated across grid.
- Best for small sets of static data.

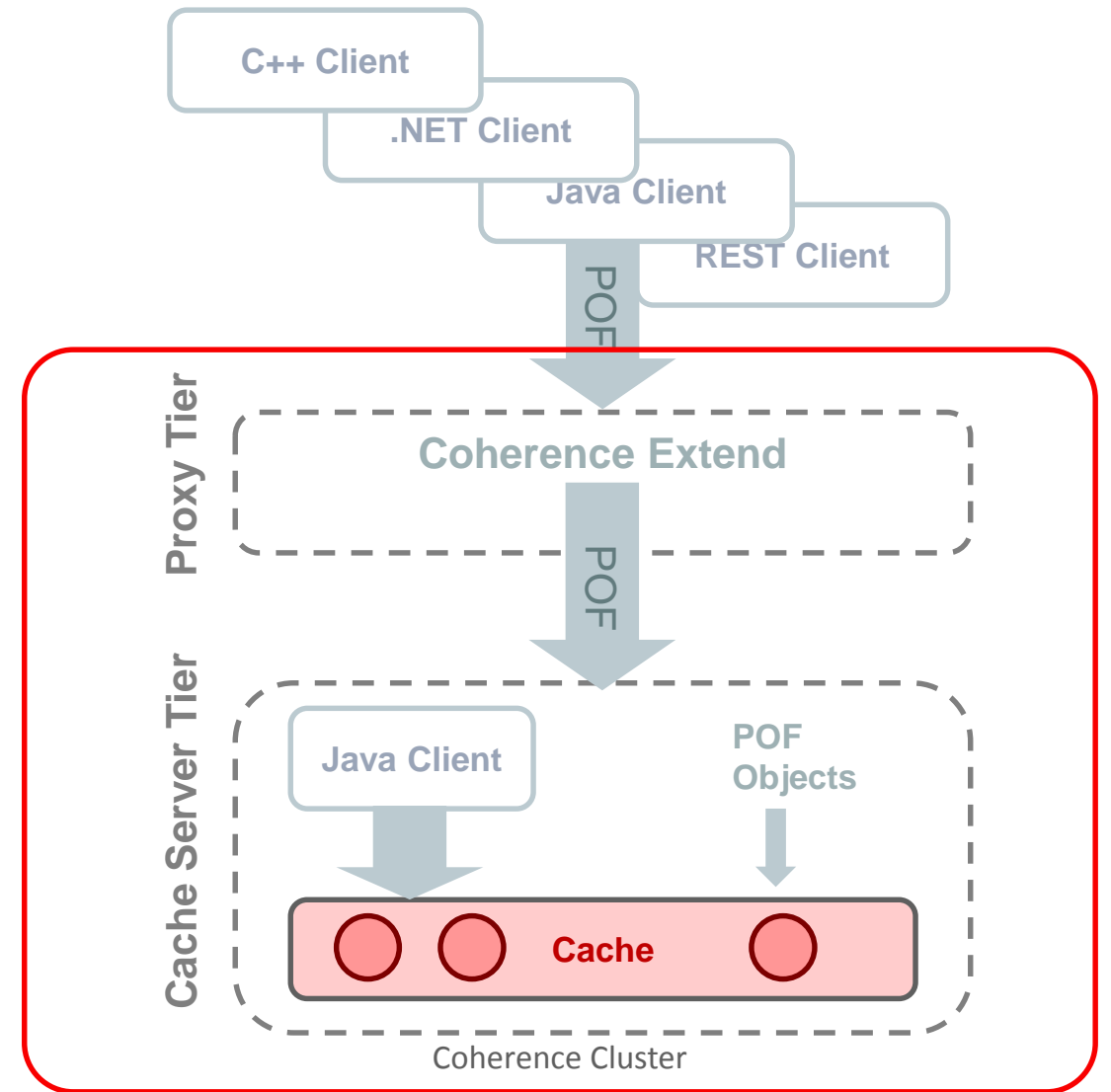


Clients, Proxies, and POF

In-Memory Data Grid @ **Oracle** Coherence

Optimized **Native** and **REST** Support

- Java, .NET, C++, and REST Clients
- Client types
 - Coherence*Extend (external to cluster)
 - Compute Clients (Java cluster members)
 - REST Clients
- Portable Object Format (POF)
 - Highly compressed object format
 - ❑ Stores data more efficiently on data grid
 - ❑ Optimizes network usage
 - Quick indexing to access individual fields

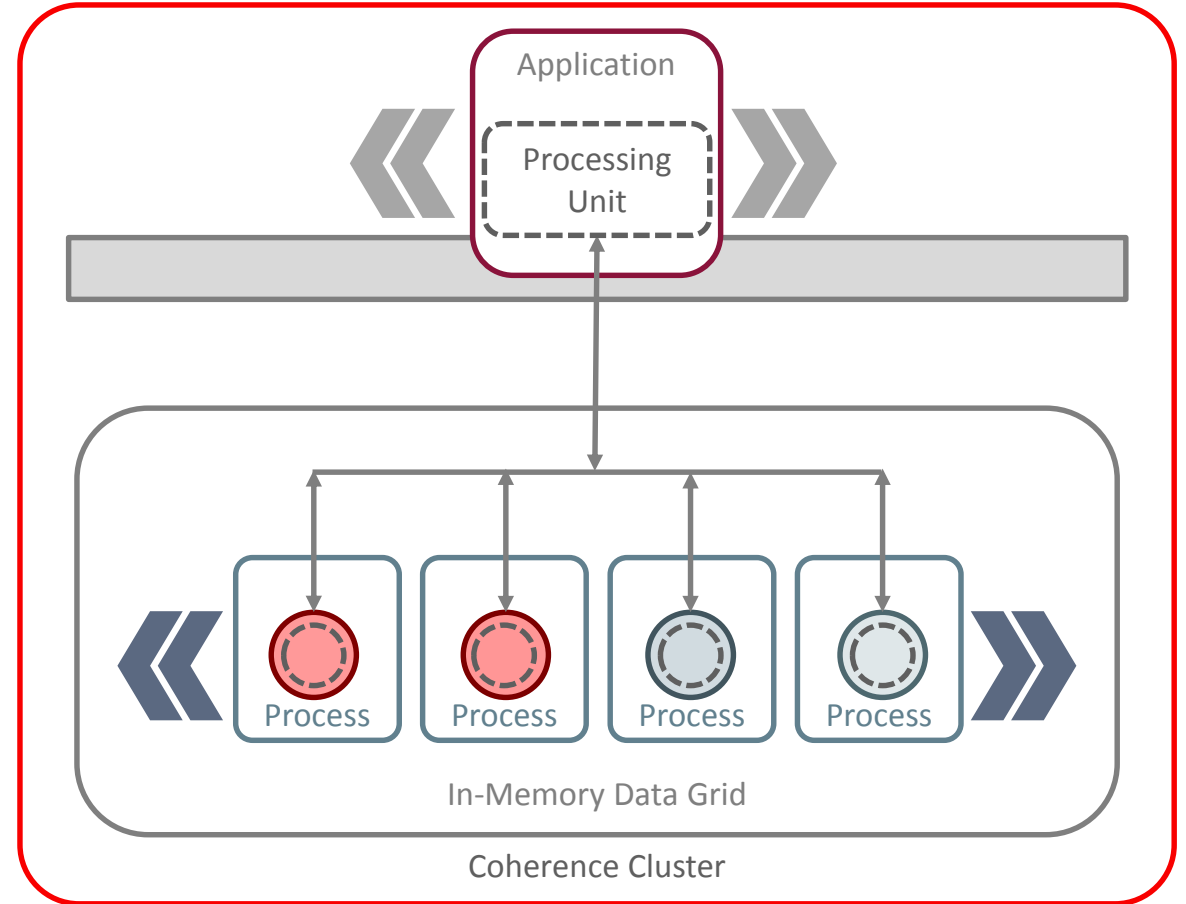


Parallel Processing

In-Memory Data Grid @ **Oracle** Coherence

Querying, Processing, Aggregating in the Data Grid

- Send processing to where the data lives.
- Processing in parallel across grid.
 - Query the Data Grid
 - Continuous Query Cache
 - Parallel Processing on the Data Grid
 - Map/Reduce Aggregation
- Once-and-only-once guarantees.
- Processing scales with the grid.

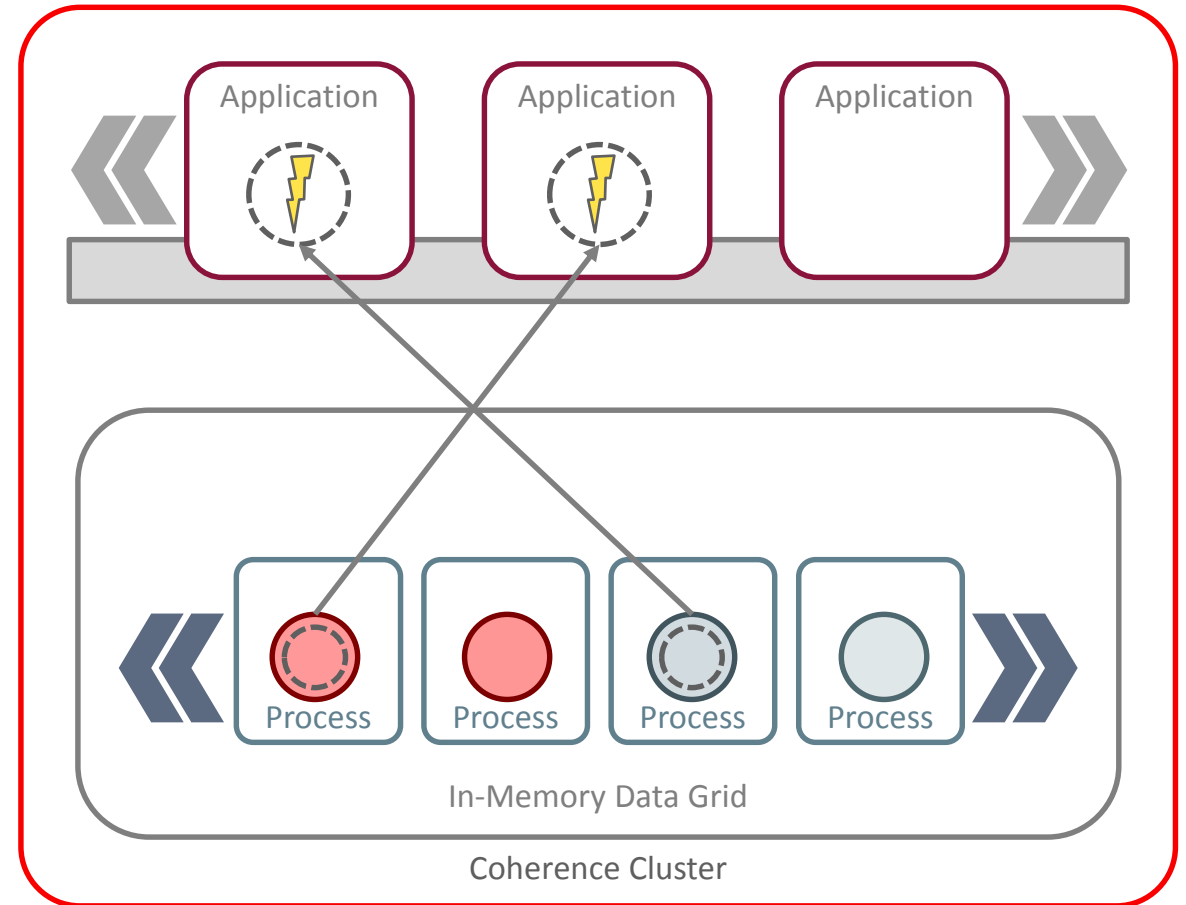


Event Notifications

In-Memory Data Grid @ **Oracle** Coherence

Support for **Event Based** Applications

- Grid based event notification
 - Java Bean Model, key and filter based events
- Complements scalable grid processing
 - Real-time data
- “Live Objects”
 - Objects can respond to own state changes
 - State always recoverable

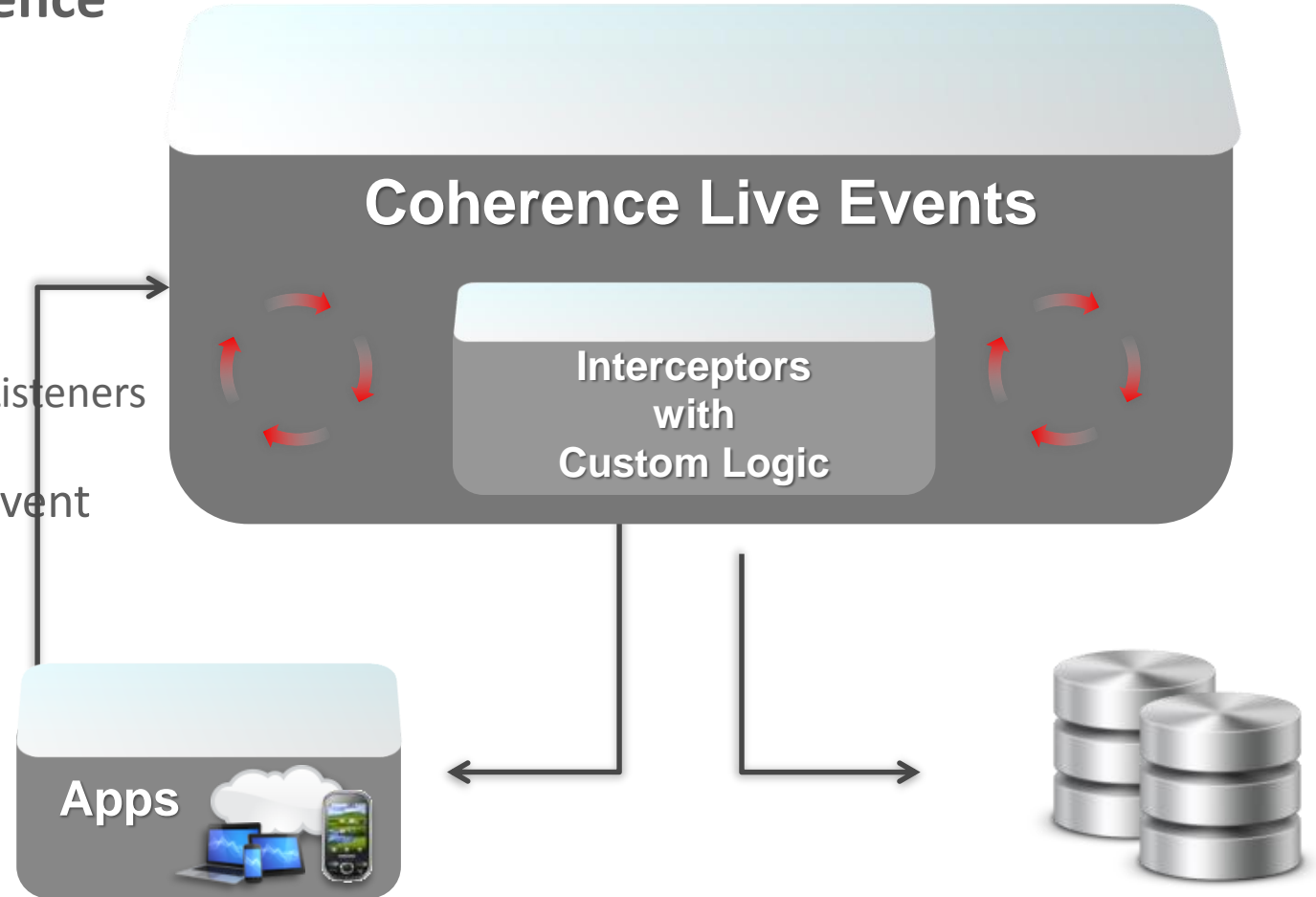


Event Notifications

In-Memory Data Grid @ **Oracle** Coherence

Event-Driven Architecture

- One programming model for all events
 - Triggers, Backing Map Listeners, Partition Listeners
- Formalizes programming semantics for event driven architectures
- Declarative configuration

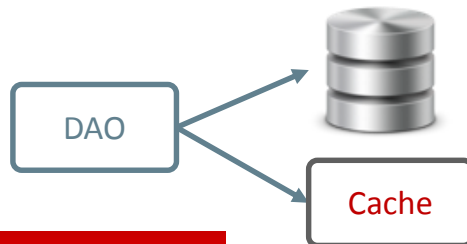


Caching Patterns

In-Memory Data Grid @ Oracle Coherence

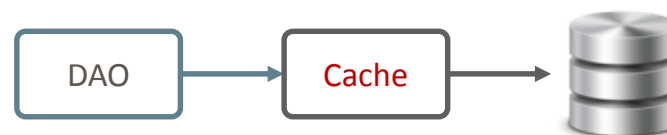
Cache Aside

- **Developer manages cache**
- Check the cache before reading from data source
- Put data into cache after reading from data source
- Evict or update cache when updating data source



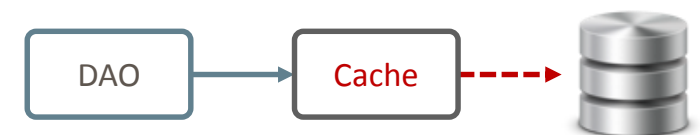
Read Through/Write Through

- **All data reads/writes occur through cache**
- Cache miss causes load from data source
- Cache updates written synchronously to data source



Write Behind

- All data writes occur through cache
- **Updates to cache written asynchronously to the data source**



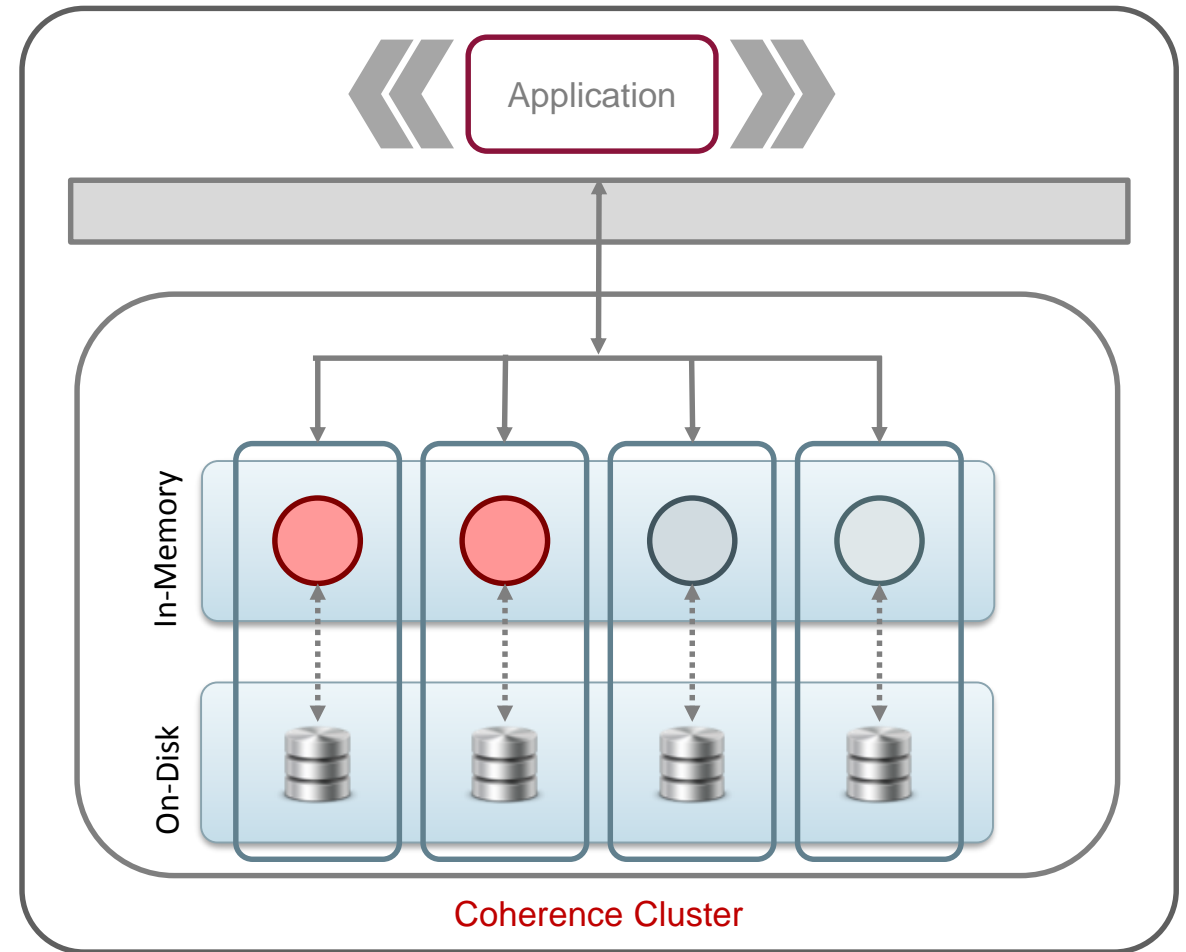
*Data Access Object: Objeto que proporciona el acceso a los datos.

Recoverable Caching – Persistencia

In-Memory Data Grid @ **Oracle** Coherence

Enabling Coherence as Store of Record

- Recoverable storage of cached data
- Automatic recovery from cluster failure
- Transactional or on-demand durability
- Multiple storage topologies
 - Maximum Scalability with distributed local disks
 - Maximum Availability with shared storage (e.g. SAN)

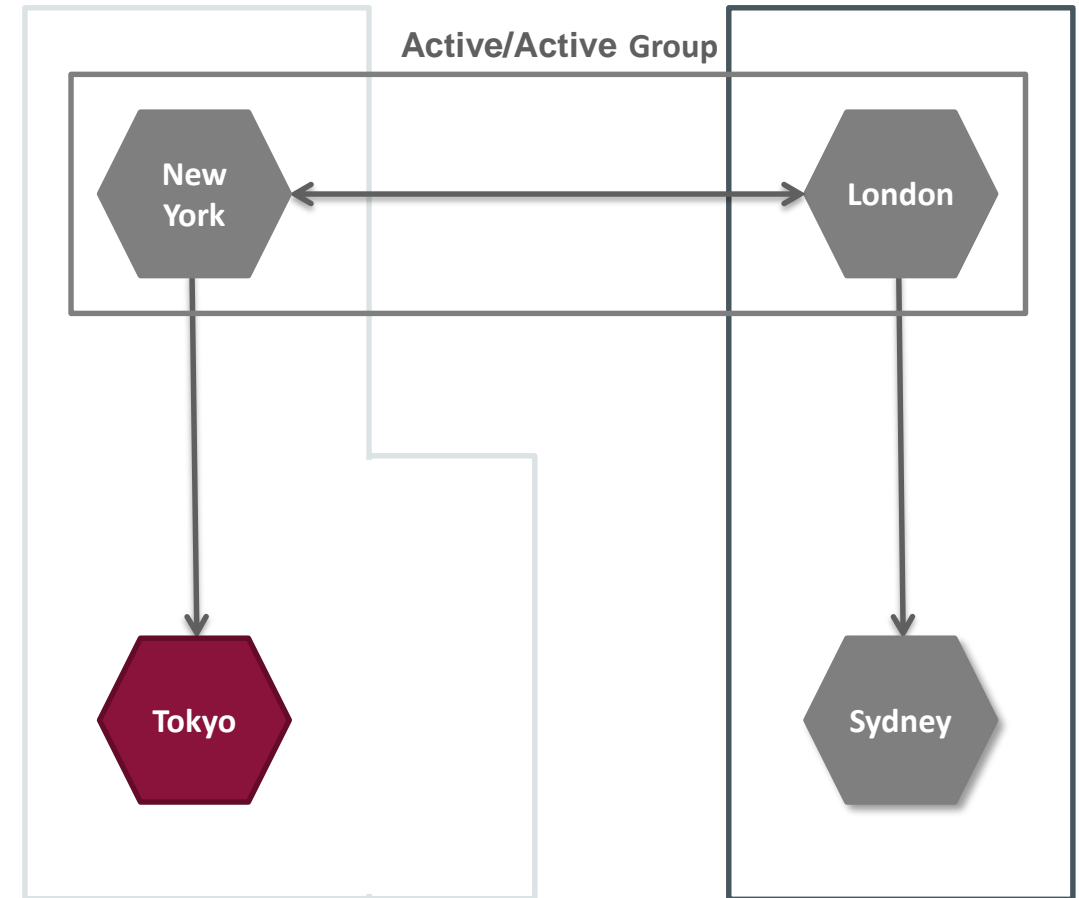


Multi-Datacenter Solutions

In-Memory Data Grid @ Oracle Coherence

Federated Caching

- Distribute data grid updates
- Span on-premise and cloud cluster
- Multiple distribution strategies
 - Active/Passive
 - Active/Active
 - Hub & Spoke
- Overlay distribution strategies across locations
- Pluggable Conflict Resolution

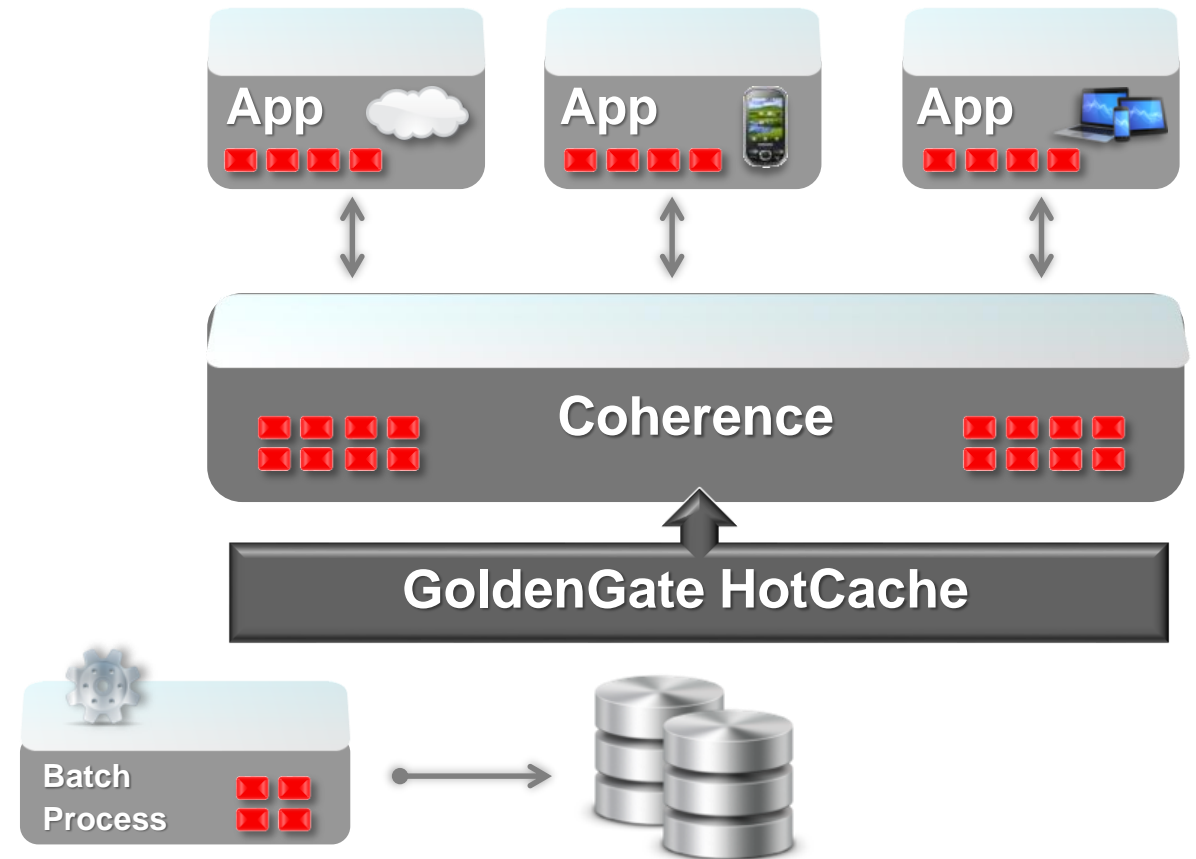


Coherence GoldenGate HotCache

In-Memory Data Grid @ **Oracle** Coherence

Real Time Database Updates for Your Apps

- Detect and reflect database changes in cache in real time
- Leverage existing technologies
 - GoldenGate, TopLink Grid
- Broaden applicability/usability of Coherence
- No code change



Java 8

In-Memory Data Grid @ **Oracle** Coherence

Lambdas

- **Functional Programming on Grid**
 - Serializable, Deployable, Remotable
 - Standard Java as EntryProcessors
- **Zero Deployment**
 - No deployed EntryProcessor classes
 - No restart on app change

Stream API

- **Streams: parallel query/aggregation API**
 - Sources are collections, files, sockets...
 - Pipelines, intermediate/terminal steps
 - Internal iteration
 - Relies on lambdas
- **Distributed (+ parallel) on Coherence**
 - Standard Java API extended to IMDG
 - Reduced custom coding
 - Faster query, mapReduce
- **Streaming Asynchronous API**

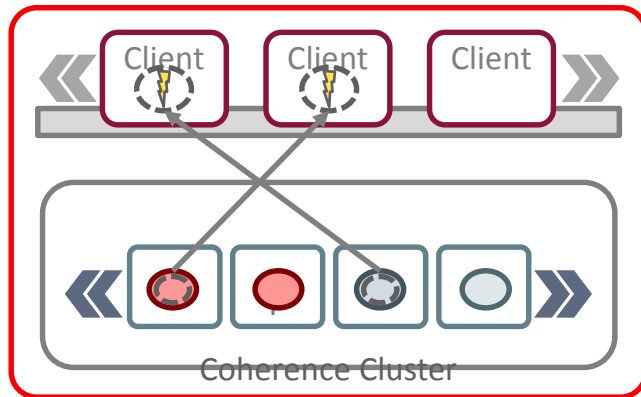
CompletableFuture

- **Improve latency and responsiveness**
 - Perform parallel operations
 - Reacting better than waiting
- **Using CompletableFuture**
 - Provide completion and/or exception callbacks
 - Chained or parallel asynchronous calls

Paradigmas de Programación

In-Memory Data Grid @ **Oracle** Coherence

Messaging



- Easier **integration with distributed topics/queues**

Coherence*JS



- Native JavaScript Client and Server-Side JavaScript programming
- Bring power of **Coherence to JavaScript and Node.JS** world

Coherence*RX



- **Reactive programming** for more efficient Coherence clients

<https://github.com/coherence-community/coherence-rx>



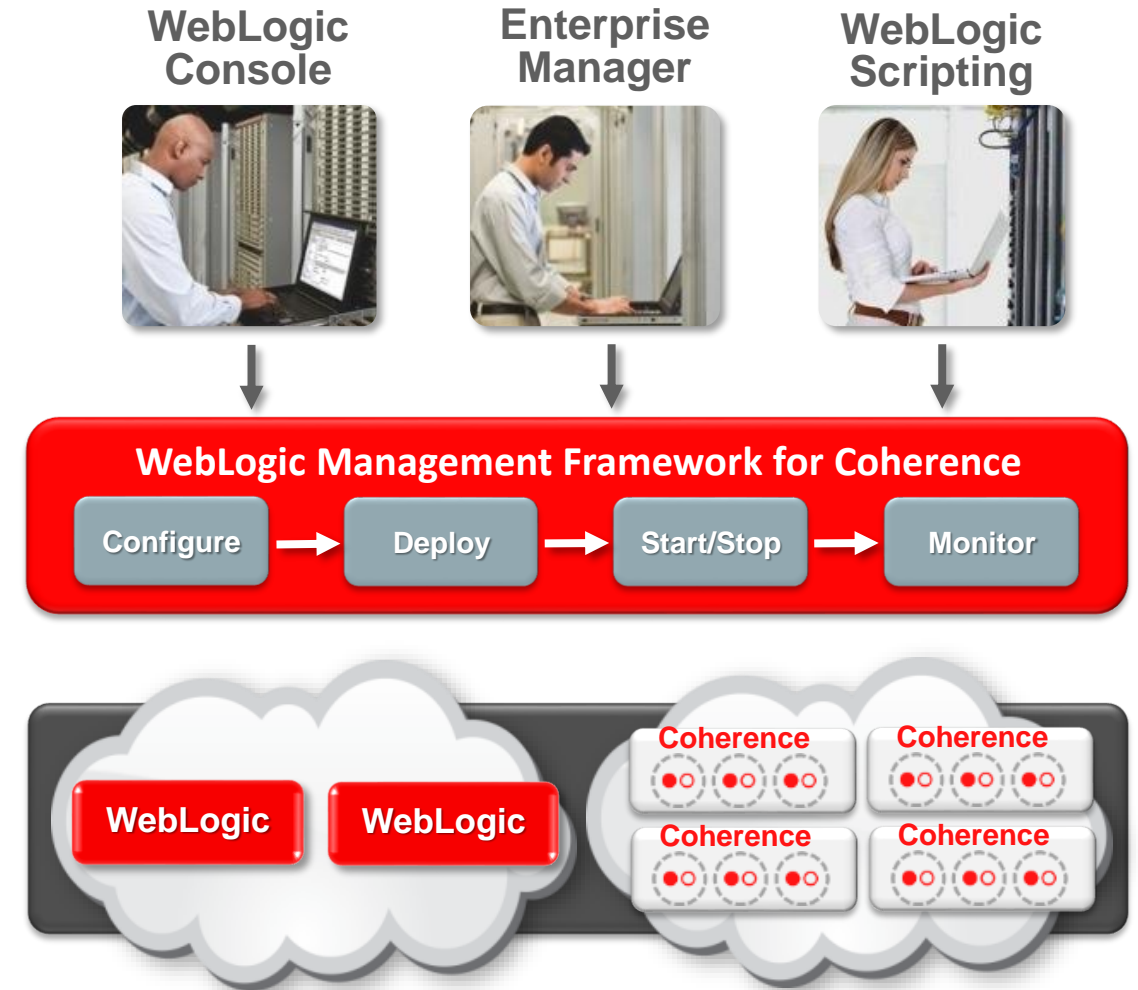
Provisión y Monitorización de la Plataforma

Managed Coherence Servers

In-Memory Data Grid @ **Oracle** Coherence

Administrative and Operational Efficiency

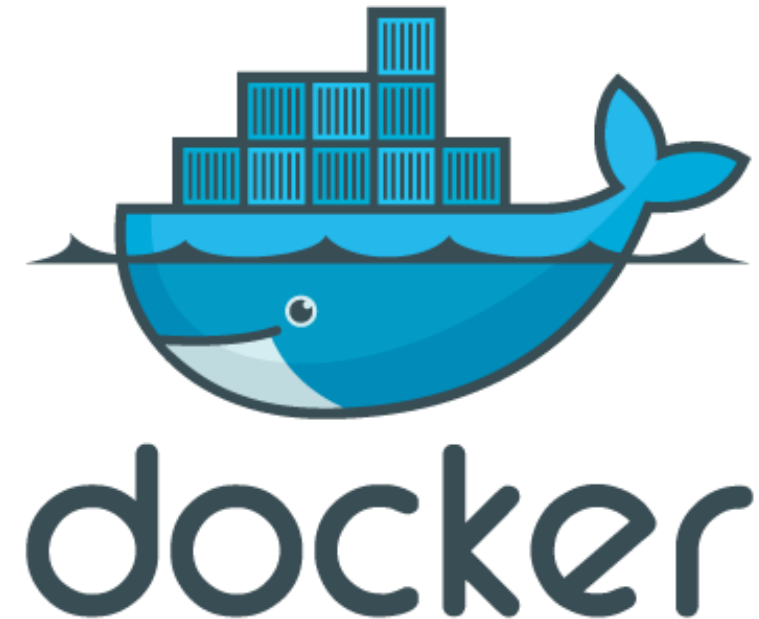
- Combined WebLogic and Coherence Infrastructure
 - WebLogic Management Framework
 - Configuration Wizard, WebLogic admin console, WLST, Node Manager
- Introduces the Grid Archive (GAR)
 - Package and Deploy



Docker

In-Memory Data Grid @ **Oracle** Coherence

- Run Coherence 12.2.1+ with Docker 1.9+
- Multi-host networking to allow for scale-out
- Resources available at <http://coherence.java.net/>
 - Setup, clustering, coherence*extend, federated caching, elastic data, persistence, and jmx
 - Limitations and Issue tracking



Automatización del Provisión

In-Memory Data Grid @ **Oracle** Coherence

Configuration management and orchestration tools:

- ANSIBLE

- <https://github.com/cvezalis/ansible-weblogic-fmw-infra-12c-R2>

- CHEF

- <https://github.com/oracle/fmw-chef-cookbook>

- PUPPET

- <https://github.com/biemondd/biemondd-oracls>



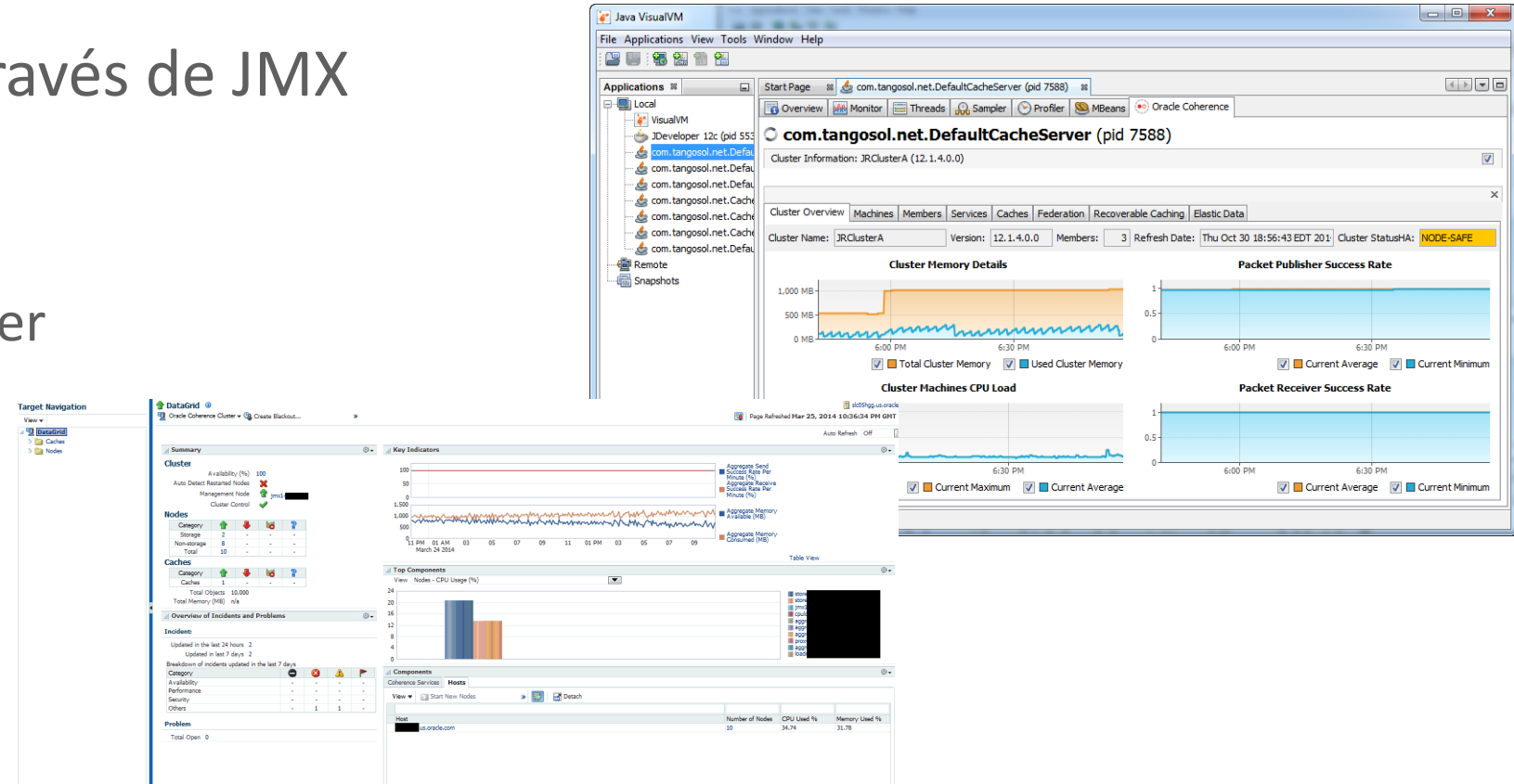
*Sin soporte oficial

Monitorización

In-Memory Data Grid @ Oracle Coherence

Monitorización a través de JMX

- JVisualVM
- Enterprise Manager
- Otros...



*Sin soporte oficial



Desarrollo e Integración Continua

Desarrollo

In-Memory Data Grid @ **Oracle** Coherence



HTML 5, Websockets, JCache
GitHub, REST, Maven...



Oracle Coherence
Incubator



*Sin soporte oficial

Questions?

ORACLE®