

D EVELOP

The Premier Conference for Developers



ORACLE®



Coherence—An Introduction

Shaun Smith
Principal Product Manager



About Me

- Product Manager for **Oracle TopLink**
 - Involved with object-relational and object-XML mapping technology for over 10 years.
- Co-Lead of Eclipse **Dali JPA Tools** Project
- Ecosystem Development Lead on proposed Eclipse Java Persistence Platform Project (**EclipseLink**)



Agenda

- Motivation for Coherence
- What is Coherence?
 - ... and what Coherence isn't!
- How does Coherence work?
 - ... the underlying design objectives
 - ... at the network layer
 - ... how data is managed
- Where to use Coherence?
 - ... features
 - ... architectural integration patterns
- Why Coherence?
- Questions...



Motivation.... Scalability

General Rules for Scalability:

- #1: Make Applications Stateless (statefulness is hard)
- #2: Use Master/Worker Pattern (coordinate tasks)
- #3: Use Messaging (notify of state changes)



Motivation....

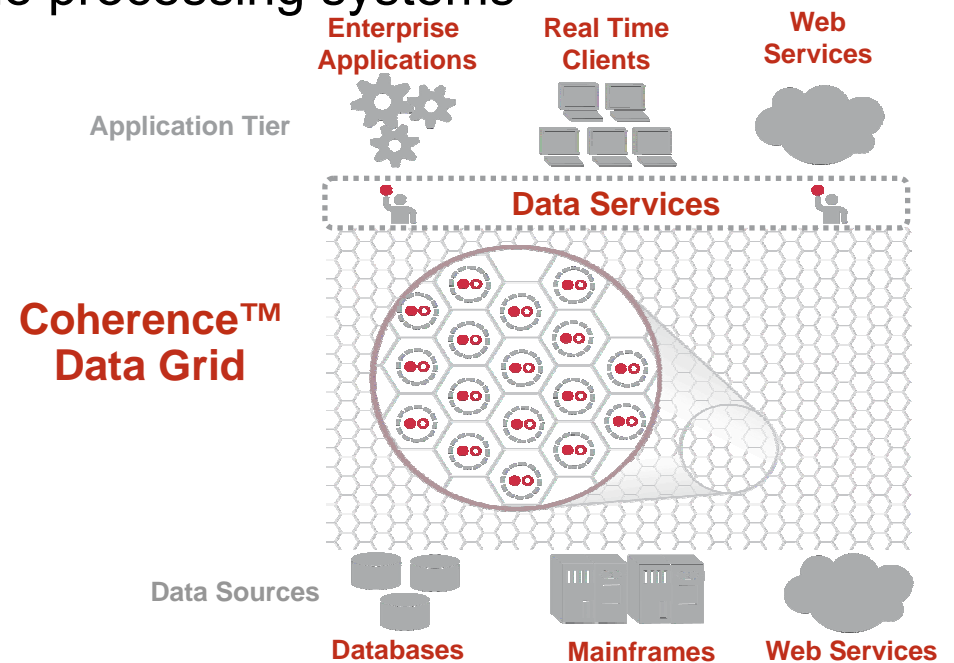
General Observations:

- #1: Where does the data to process come from?
- #2: Imposes “take a copy of the data with you”
- #3: Imposes “process it all in one place”
- #4: Forced to introduce caching to keep data “close”
- #5: Introduces single points-of-failure & bottleneck
- #6: Applications may scale-out (inexpensive), but Data Sources have to scale-up (expensive)
- #7: Manual partitioning Data / Services increases complexity

Motivation....

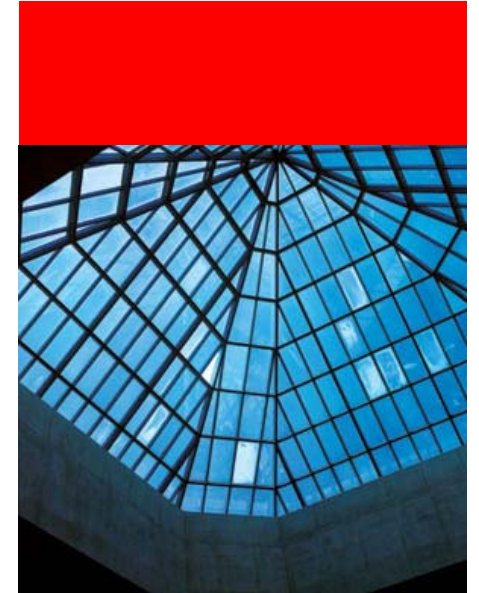
Coherence is designed to...

- Resolve these challenges
- Extend the life of Data Sources & Architectures
- As a foundation for extreme processing systems





What is Coherence? ... and what it isn't!





What is Coherence?

Cluster-based Data Management Solution for Applications



What is Coherence?

Or... Distributed Memory Data
Management Solution
(aka: Data Grid)



What is Coherence? It's Clustering!

Coherence **Clustering** is special... It means...

- Collection of processes (members) that work together
- All members have equal responsibility*
 - for the health of the cluster
- No static/tight coupling of responsibilities to hardware
- No masters and workers/slaves
- No centralized registries of data or services
- No single points of failure
- No single points of bottleneck

*members may be asked to perform individual tasks... eg: Grid Agents



What is Coherence? Data

Data in Coherence...

- Any serializable* Object
- No proprietary classes to extend**
- Fully native Java & .NET interoperability (soon C++)
- No byte-code instruction manipulation or weaving
- Not forced to use Relational Models, Object-Relational-Mapping, SQL etc
- Just real POJOs and PONOs (soon POCOs)
 - Domain objects are fine!

*serialization = writing to binary form

** implementing specialized interfaces improves performance!



What is Coherence? Data

Data in Coherence...


- Different topologies for your Data
- Simple API for all Data, regardless of Topology
- Things you can do...
 - Distributed Objects, Maps & Caching
 - Real-Time Events, Listeners,
 - Parallel Queries & Indexing
 - Data Processing and Service Agents (Grid features),
 - Continuous Views,
 - Aggregation,
 - Persistence, Sessions...



What is Coherence? Management Solution

Management Solution...

- Responsible for Clustering, Data and Service management, including partitioning
- Ideally engineers should not have to...
 - design, specify and code how partitioning occurs in a solution
 - handle Remote Exceptions
 - manage the Cluster, either manually or in code
 - shutdown the system to add new resources or repartition
 - use “consoles” to recover or scale a system.
- These are impediments to scaling cost effectively
- As #members $\rightarrow \infty$, management cost should be C
 - Ideally $\log(\text{\#members})$
- Clustering technology should invisible in your solution!



What is Coherence? For Applications

It's for **Applications!**

- Coherence doesn't require a container / server
- A single library*
- No external / open source dependencies!
- Won't cause JAR / classloader hell (cf: DLL hell)
- Can be embedded or run standalone
- Runs where Java SE / EE, .NET runs
- Won't impose architectural patterns**

* May require addition libraries depending on features required.

EG: spring, web, hibernate, jta integration are bundled as separate libraries.

** Though we do make some suggestions ☺



Clustered Hello World...

```
public static void main(String[] args)
    throws IOException {
    NamedCache nc = CacheFactory.getCache("test");
    nc.put("message", "Hello World");
    System.out.println(nc.get("message"));

    System.in.read(); //may throw exception
}
```

- **Joins / Establishes a cluster**
- **Places an Entry (key, value) into the Cache “test” (notice no configuration)**
- **Retrieves the Entry from the Cache.**
- **Displays it.**
- **“read” at the end to keep the application (and Cluster) from terminating.**



Clustered Hello World...

```
public static void main(String[] args) throws
    IOException {
    NamedCache nc = CacheFactory.getCache("test");
    System.out.println(nc.get("message"));
}
```

- **Joins / Establishes a cluster**
- **Retrieves the Entry from the Cache.**
- **Displays it**

- **Start as many applications as you like... they all cluster the are able to share the values in the cache**



What Coherence isn't!

- It's not an in-memory-database
 - Though it's often used for transactional state and as a transient system-of-record
 - Used for eXtreme Transaction Processing
- You can however:
 - do queries – in a parallel – but not restricted to relational-style (it's not a RDBMS)
 - use SQL-like queries
 - perform indexing (like a DB)
 - do things like Stored Procedures
 - establish real-time views (like Materialized Views).



What Coherence isn't!

- It's not a messaging system
 - You can use Events and Listeners for data inserts, updates, deletes
 - You can use Agents to handle data changes
 - You can use Filters to filter events
- It's not "just" a Cache!
 - Caches expire data!
 - Customers actually turn expiry off!
 - Why do they take that risk? Why do we let them?
 - Data management is based on reliable clustering technology. It's stable, reliable and highly available.



What is Coherence?

Dependable resilient scalable cluster technology

that

enables developers to effortlessly
cluster stateful applications

so they can

dynamically and reliably share data,
provide services and respond to events

to

scale-out solutions to meet business demand.



How does Coherence work?





Clustering is about Consensus!

Coherence Clustering is very different!

Goal:

- Maintain Cluster Membership Consensus all times
- Do it as fast as physically possible
- Do it without a single point of failure or registry of members
- Ensure all members have the same responsibility and work together maintain consensus
- Ensure that no voting occurs to determine membership



Clustering is about Consensus!

Why: If all members are always known...

- We can partition / load balance Data & Services
- We don't need to hold TCP/IP connections open (resource intensive)
- Any member can “talk” directly with any other member (peer-to-peer)
- The cluster can dynamically (while running) scale to any size

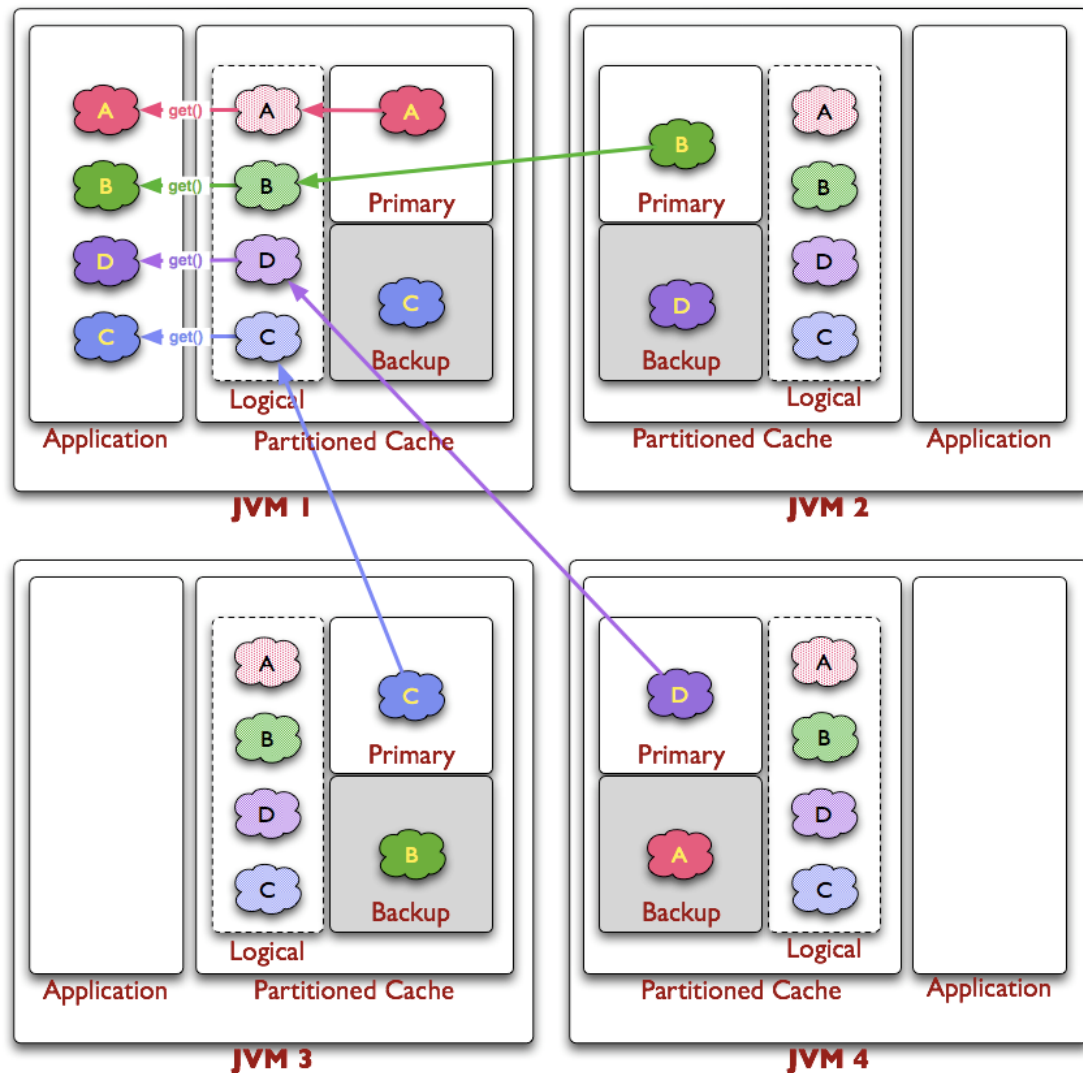
Partitioned Topology : Data Access

Data Access Topologies

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

Partitioned Topology

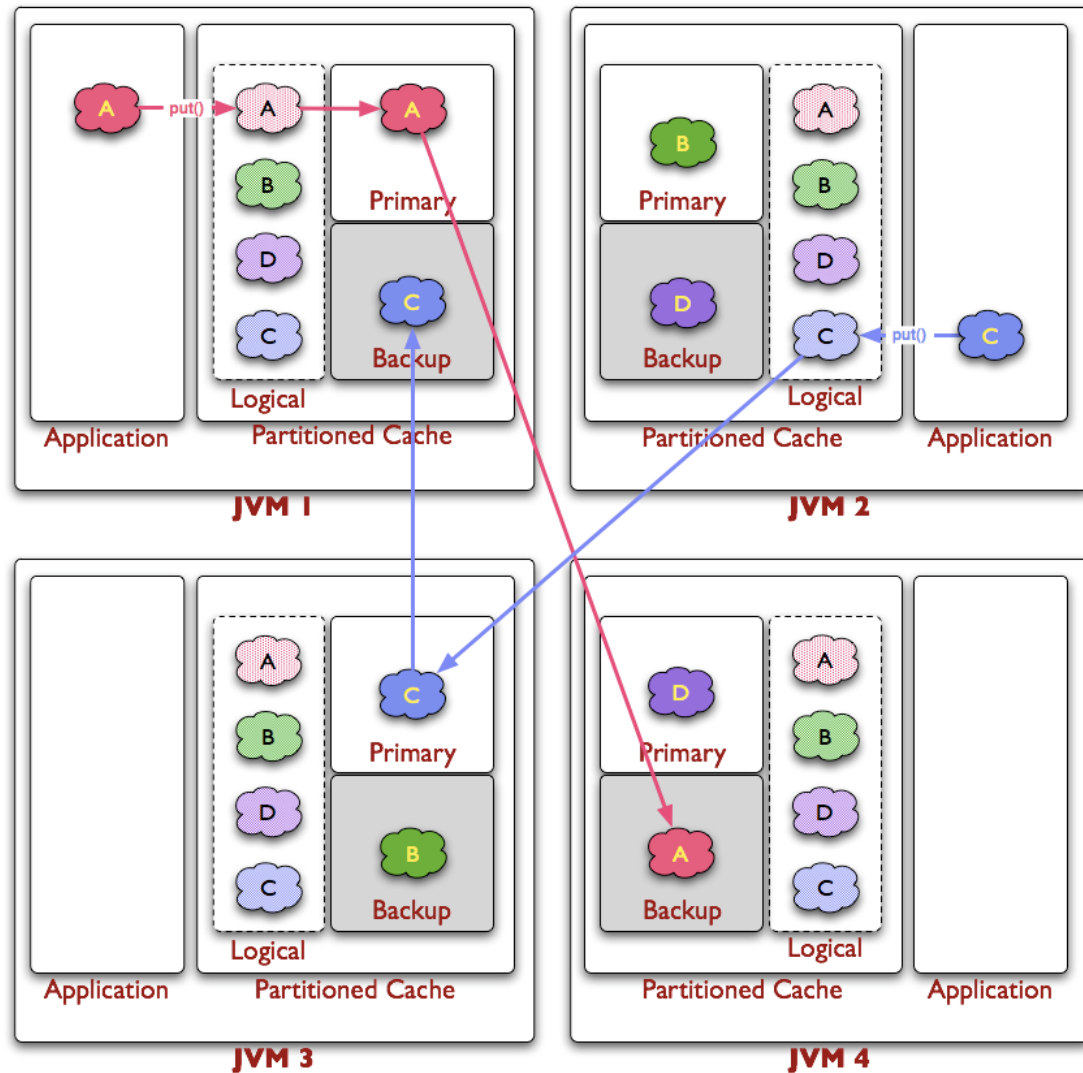
- Data spread and backed up across Members
- Transparent to developer
- Members have access to all Data
- All Data locations are known – no lookup & no registry!



Partitioned Topology : Data Update

Partitioned Topology

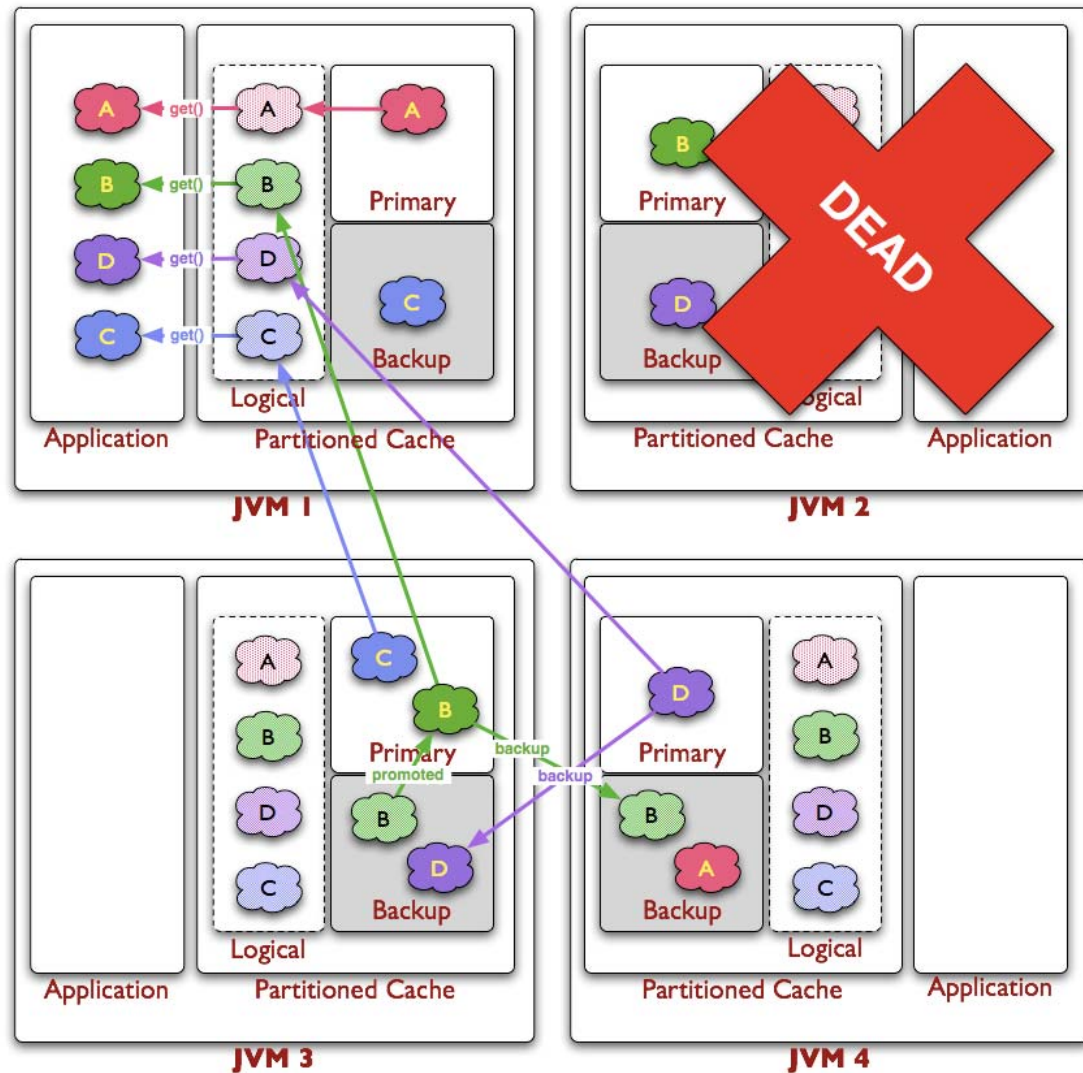
- Synchronous Update
- Avoids potential Data Loss & Corruption
- Predictable Performance
- Backup Partitions are partitioned away from Primaries for resilience
- No engineering requirement to setup Primaries or Backups
- Automatically and Dynamically Managed



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





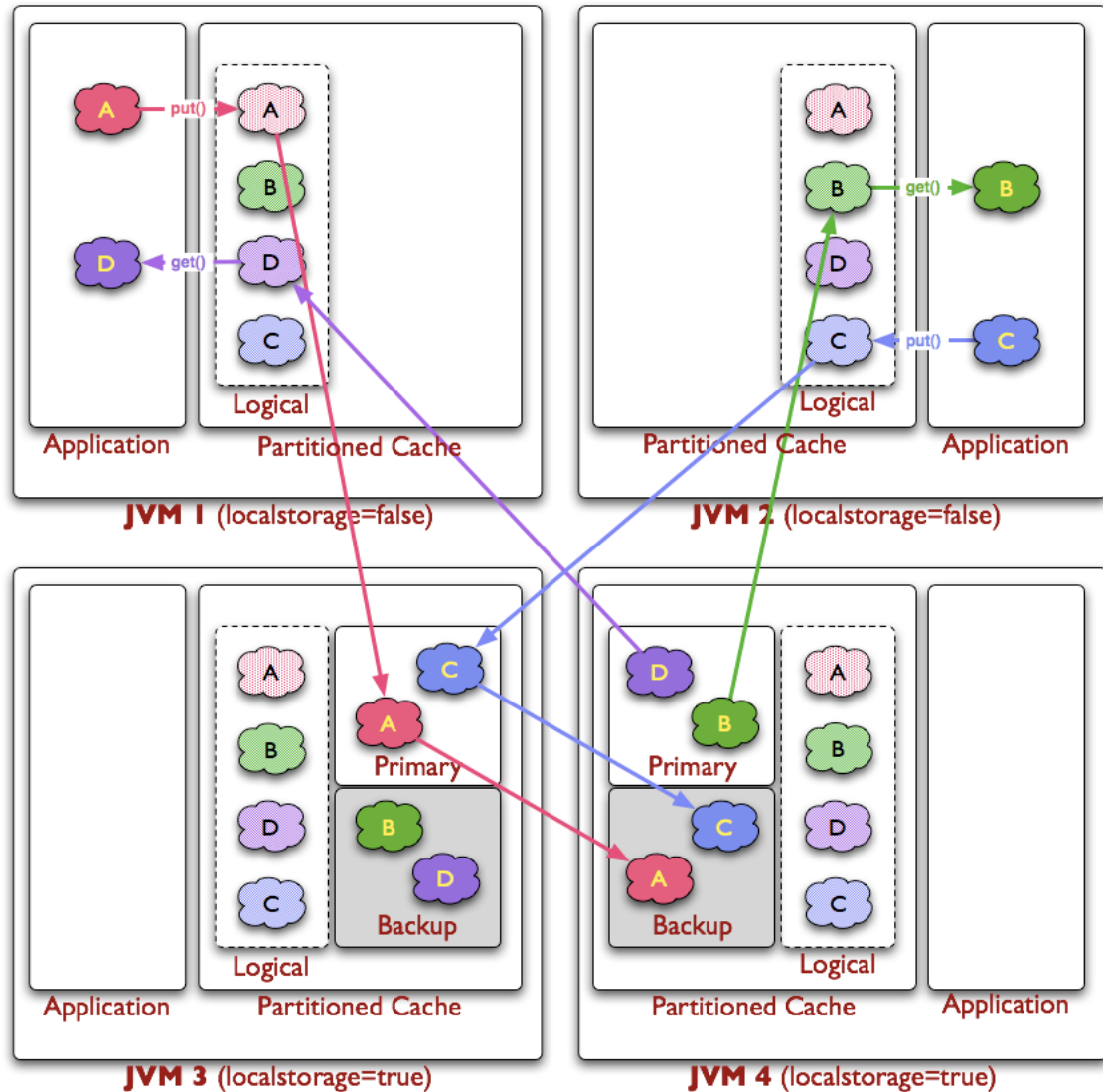
Partitioned Topology

- Deterministic latencies for data access and update
- Linearly scalable by design
 - Including Capacity
- All operations point-to-point (without multi-cast)
- No TCP/IP connections to create / maintain
- No loss of in-flight operations while repartitioning
- No requirement to shutdown cluster to
 - recover from member failure
 - add new members
 - add named caches
- No network exceptions to catch during repartitioning
- Dynamic repartitioning means scale-out on demand

Partitioned Topology : Local Storage

Partitioned Topology

- Some members are temporary in a cluster
- They should not cause repartitioning
- Repartitioning means work for the other members (and network traffic)
- So turn off storage!

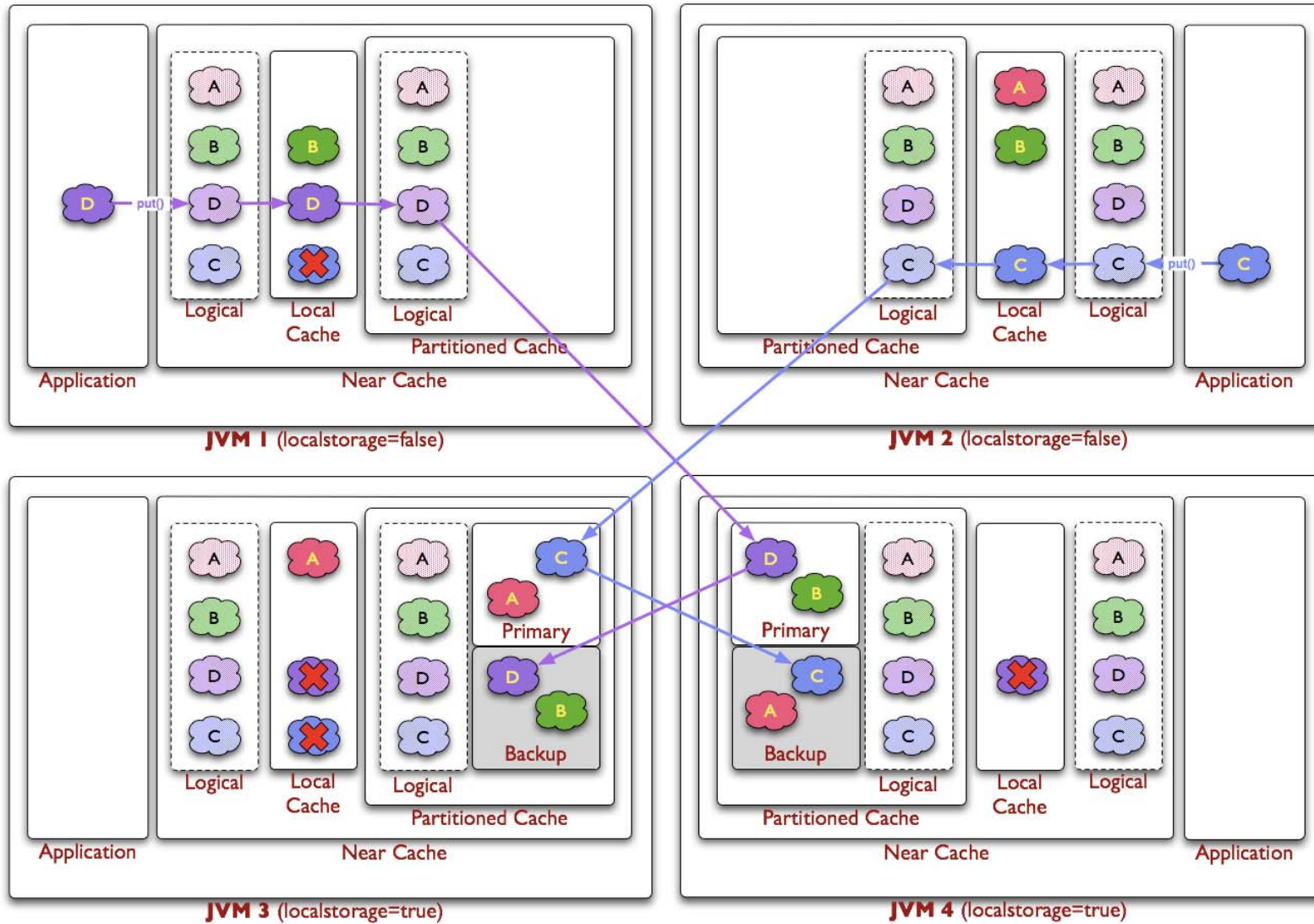




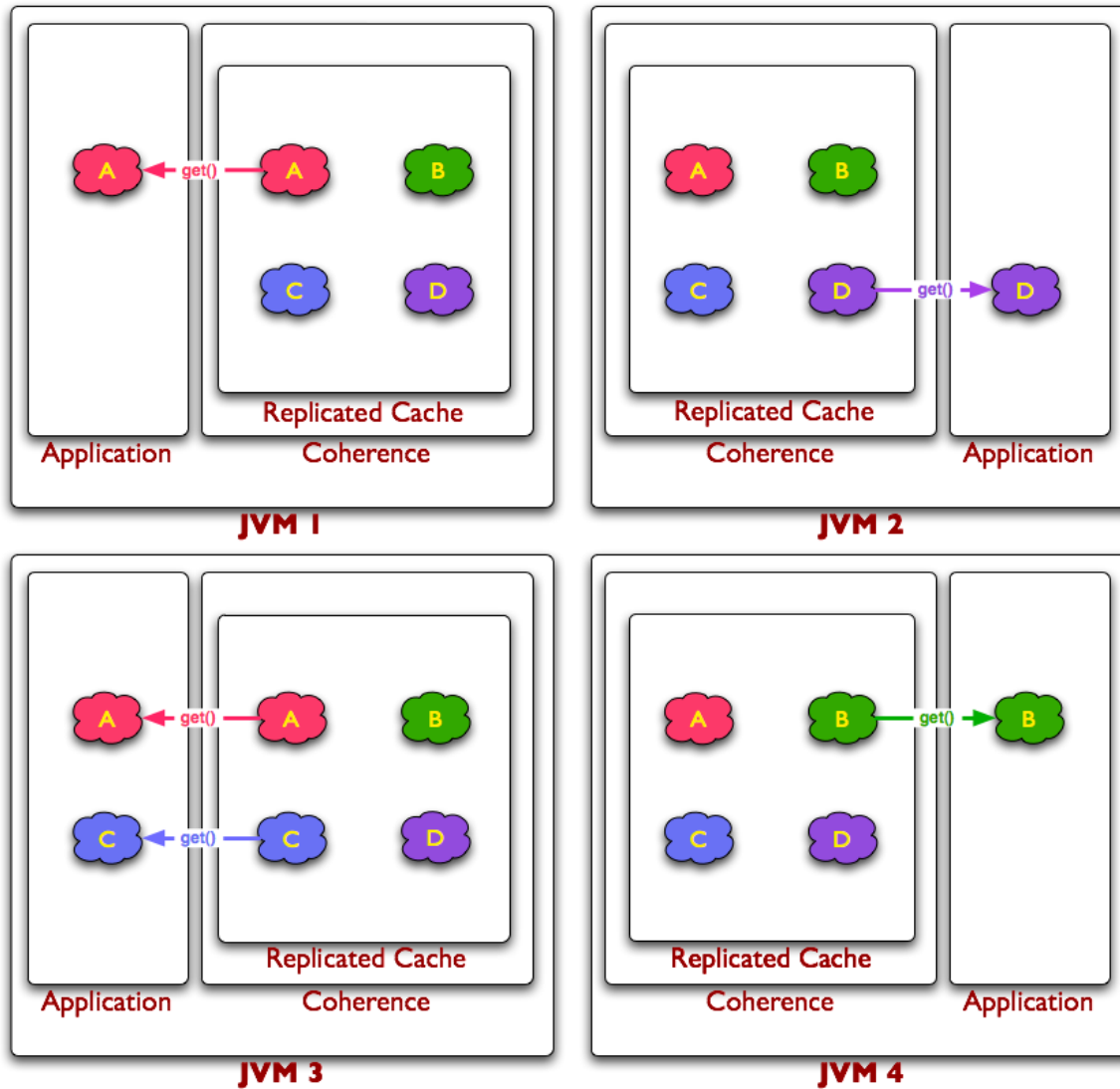
Topology Composition : Near Topology

- Coherence allows Topologies to be Composed
- Base Topologies
 - Local, Replicated, Partitioned / Distributed, Extend
- Composition Topologies
 - Near, Overflow...
- Near Topology
 - Compose a Front and a Back Topology
 - Permit L1 and L2 caching
 - Both Front and Back may be completely different
 - Both may have different Expiry Policies
- Expiry Policies
 - LRU, LFU, Hybrid, Seppuku, Custom...

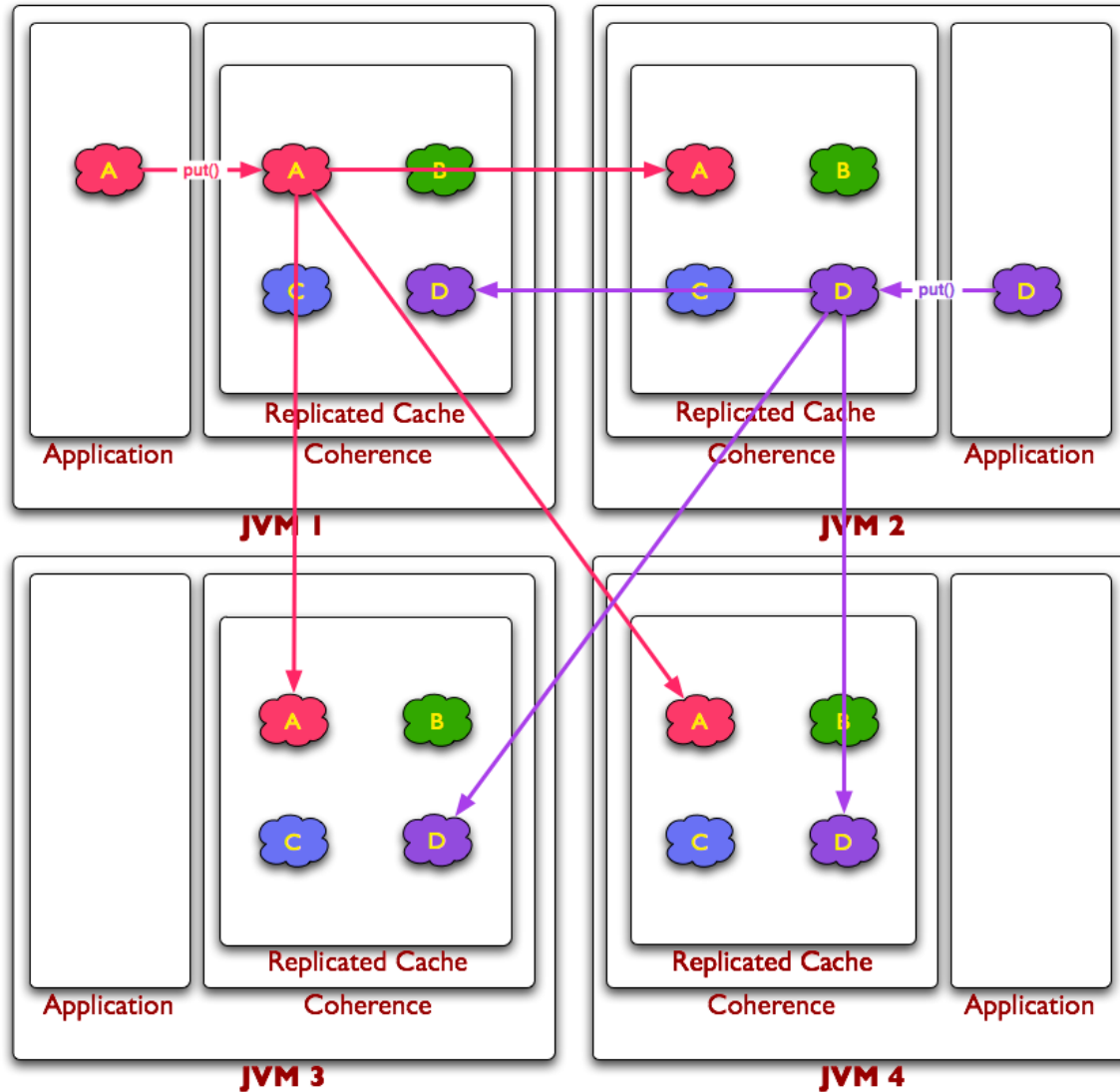
Topology Composition : Near Topology



Replicated Topology : Data Access

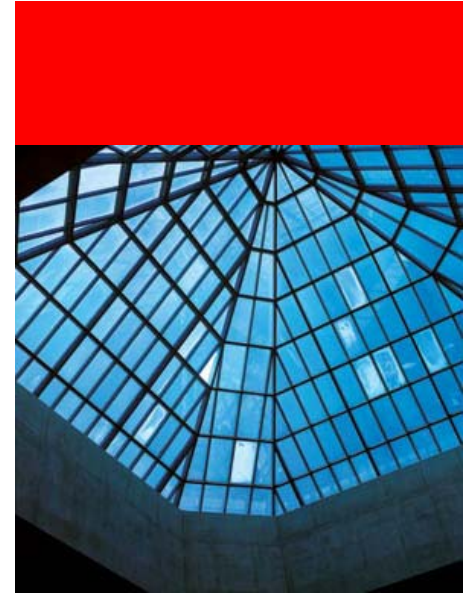


Replicated Topology : Data Update





Using Coherence





Features : Traditional

- Implements Map interface
 - Drop in replacement. Full concurrency control. Multi-threaded. Scalable and resilient!
`get, put, putAll, size, clear, lock, unlock...`
- Implements JCache interface
 - Extensive support for a multitude of expiration policies, including none!
- More than “just a Cache”. More than “just a Map”



Features : Observable Interface

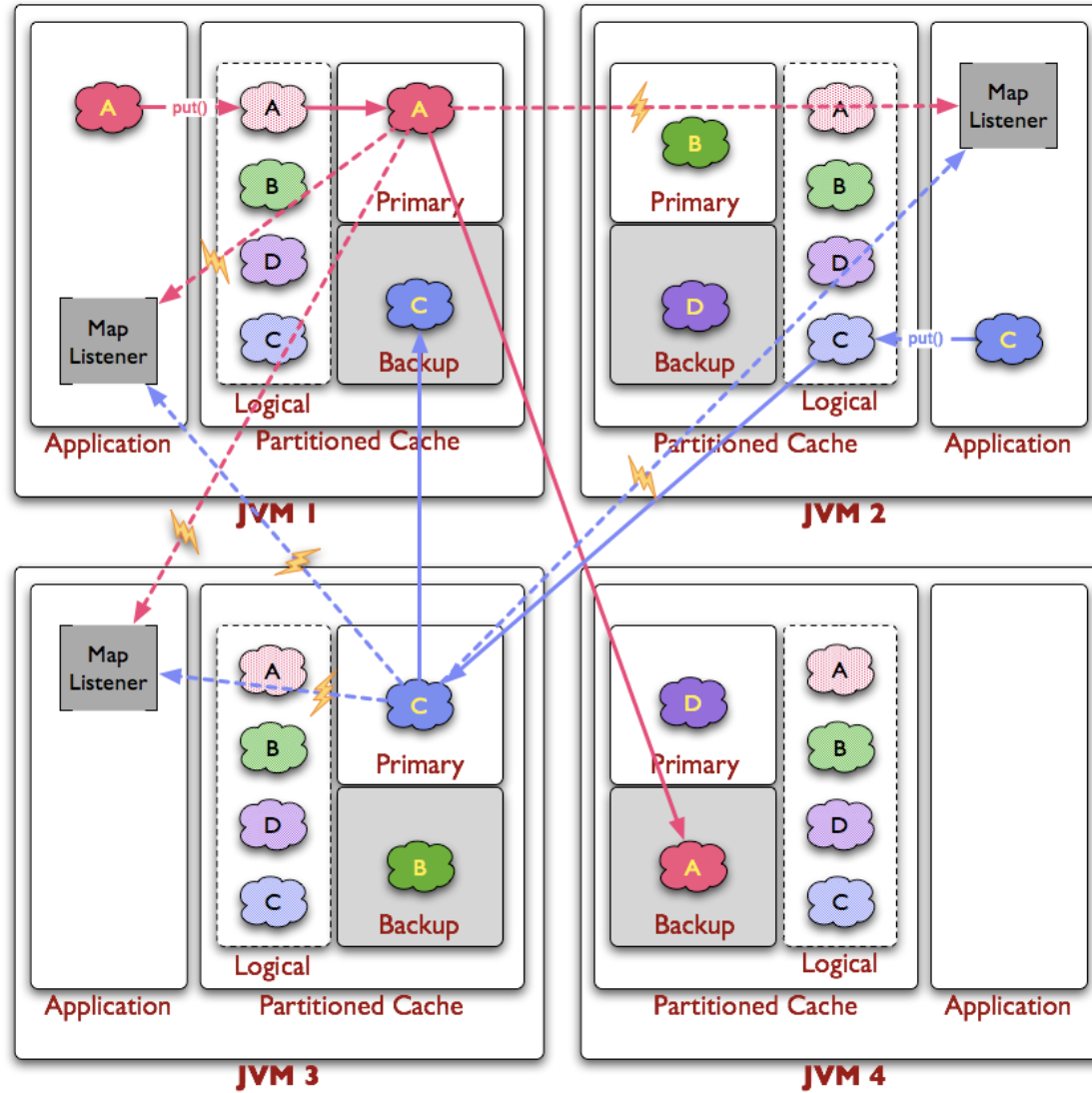
- Real-time filterable (bean) events for entry insert, update, delete
- Filters applied in parallel (in the Grid)
- Filters completely extensible
- A large range of filters out-of-the-box:

All, Always, And, Any, Array, Between, Class, Comparison, ContainsAll, ContainsAny, Contains, Equals, GreaterEquals, Greater, In, InKeySet, IsNotNull, IsNull, LessEquals, Less, Like, Limit, Never, NotEquals, Not, Or, Present, Xor...

- Events may be synchronous

```
trades.addMapListener(  
    new StockEventFilter("ORCL"),  
    new MyMapListener(...));
```

Features : Observable Interface

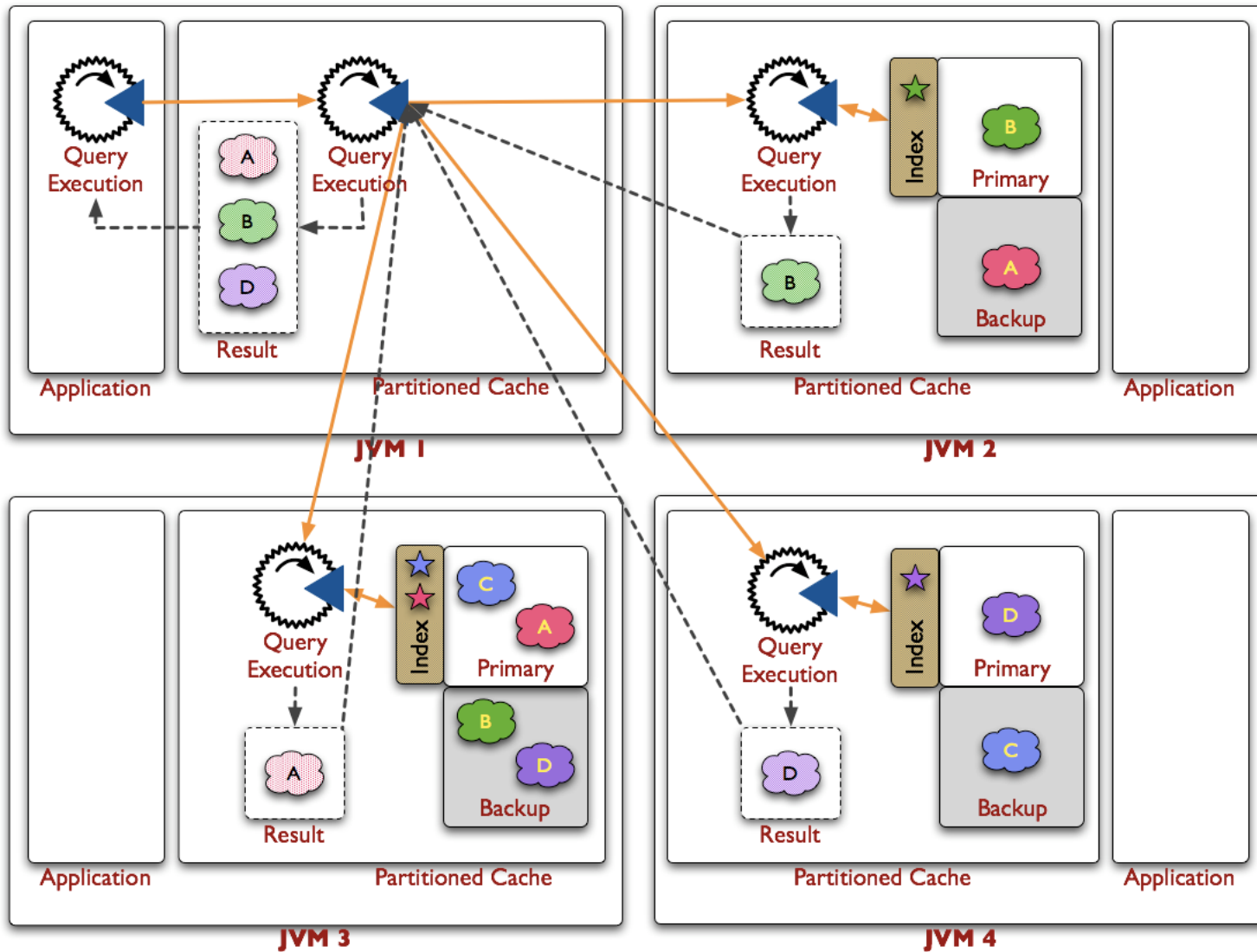




Features : QueryMap Interface

- Find Keys or Values that satisfy a Filter.
`entrySet(...)` , `keySet(...)`
- Define indexes (on-the-fly) to extract and index any part of an Entry
- Executed in Parallel
- Create Continuous View of entries based on a Filter with real-time events dispatch
 - Perfect for client applications “watching” data

Features : QueryMap Interface





Features : InvocableMap Interface

- Execute processors against an Entry, a Collection or a Filter
- Executions occur in parallel (aka: Grid-style)
- No “workers” to manage!

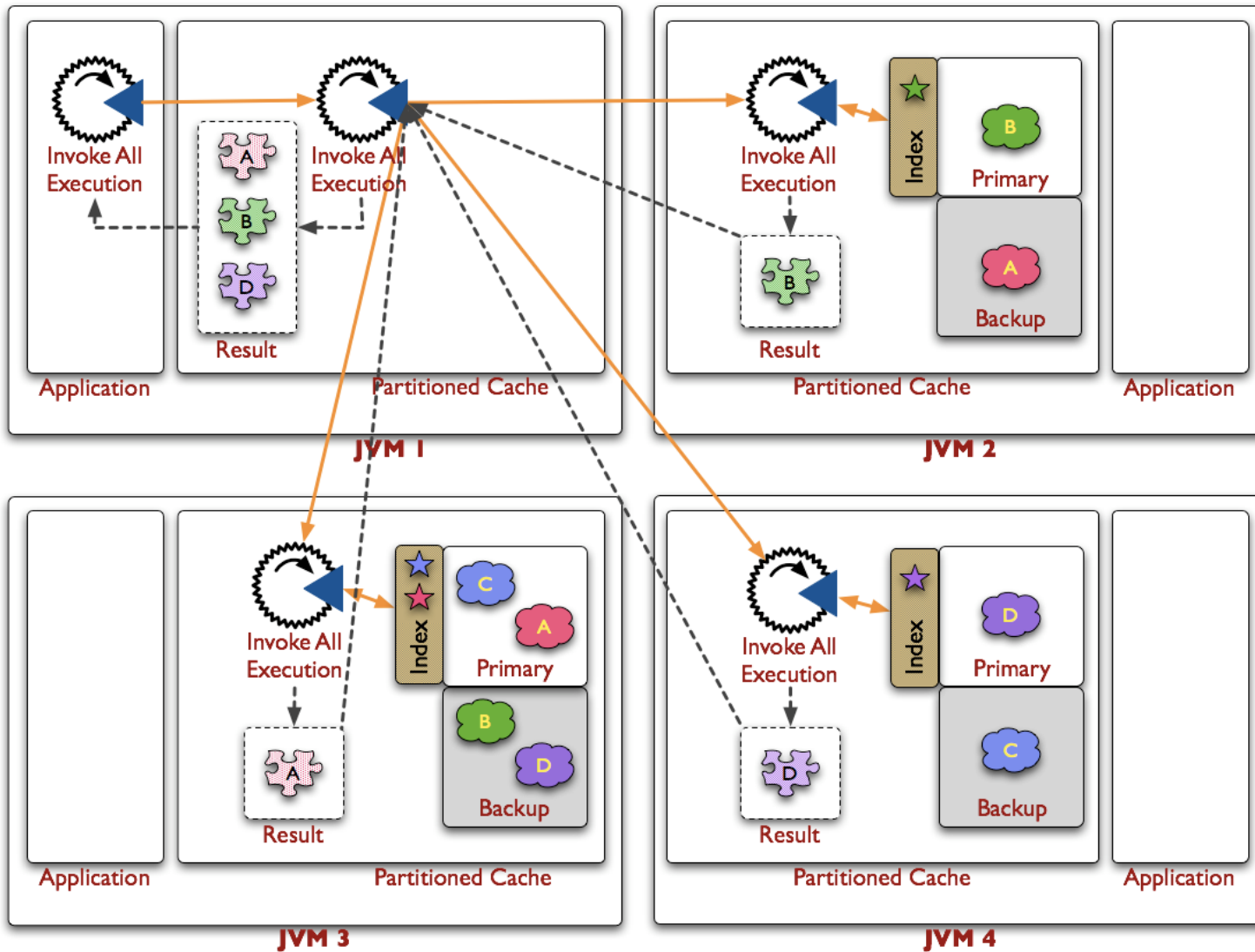
- Processors may return any value

```
trades.invokeAll(  
    new EqualsFilter("getSecurity", "ORCL"),  
    new StockSplit(2.0));
```

- Aggregate Entries based on a Filter

```
positions.aggregate(  
    new EqualsFilter("getSecurity", "ORCL"),  
    new SumFilter("amount"));
```

Features : InvocableMap Interface





Architectural Integration Possibilities!

Data Source Integration (customizable per cache)

Read Through: Read from CacheLoader when data not in grid

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

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

Session Management

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



Architectural Integration Possibilities!

Spring Integration:

Data Grid For Spring:

Data Grid Spring Beans managed by Coherence. Beans become shared singletons and offer both Data and Services to Spring Applications

Spring Applications naturally become clustered, including cluster events to determine cluster membership changes

Expose Coherence Caches as Beans in Spring Configuration



Architectural Integration Possibilities!

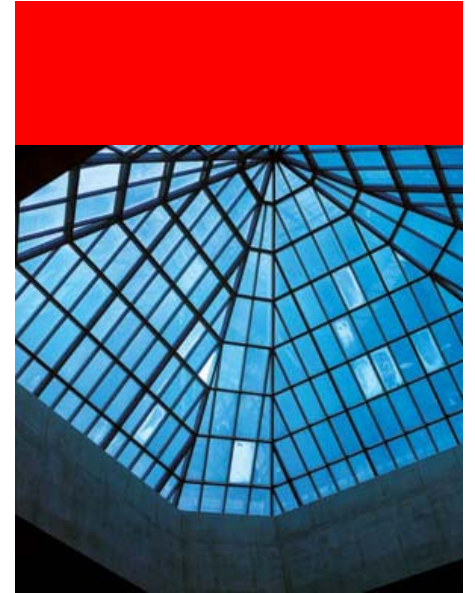
Provide:

Push / Pull data model based on subscription and event notification

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



Why Coherence?

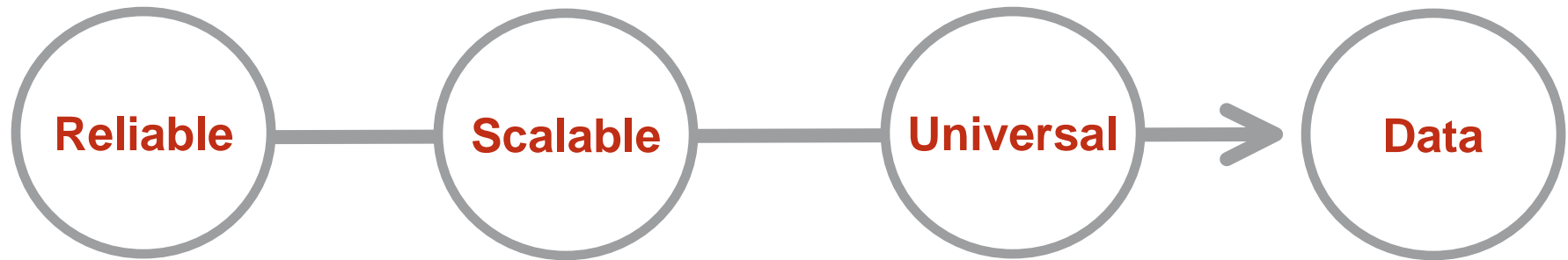




Why Coherence?

- Scale-out stateful applications
- “If you need business agility!”
- Save resources!
 - Avoid managing clusters
 - Avoid designing systems around specialized “cluster masters”
 - Avoid manually “coding in” data and service partitions
- If you want predictable scale-out costs, without re-coding or reconfiguring!
- If you want truly native language support! No wrappers or embedded third-party libraries.

Why Coherence?



- Built for continuous operation
- Data Fault Tolerance
- Self-Diagnosis and Healing
- “Once and Only Once” Processing

- Dynamically Expandable
- No data loss at any volume
- No interruption of service
- Leverage Commodity Hardware
- Cost Effective

- Single view of data
- Single management view
- Simple programming model
- Any Application
- Any Data Source

- Data Caching
- Analytics
- Transaction Processing
- Event Processing



ORACLE IS THE INFORMATION COMPANY